HPSG and Lexical Functional Grammar

Stephen Wechsler and Ash Asudeh

May 12, 2020

1 Introduction

Head-Driven Phrase Structure Grammar is similar in many respects to its cousin framework, Lexical Functional Grammar or LFG (**??**). Both HPSG and LFG are lexicalist frameworks in the sense that they distinguish between the morphological system that creates words, and the syntax proper that combines those fully inflected words into phrases and sentences. Both frameworks assume a lexical theory of argument structure (**?**) in which verbs and other predicators come equipped with valence structures indicating the kinds of complements and other dependents that the word is to be combined with. Both theories treat control (equi) and raising as lexical properties of certain control or raising predicates. Both systems are historically related to unification grammar (**?**), but whereas HPSG adopts unification as a declarative constraint on shared structures, LFG captures the effects of unification with declarative descriptions that describe two feature values as equal. Both frameworks use recursively embedded attribute-value matrices; these structures directly model linguistic expressions in LFG, but are understood in HPSG as grammatical descriptions satisfied by the directed acyclic graphs that model utterances. Both theories allow phonologically empty nodes of the constituent structure, although researchers in both theories tend to avoid them unless they are well-motivated.

At the same time, there are interesting differences. Each theory makes available certain representational resources that the other theory lacks. LFG has output filters in the form of *constraining equations*, HPSG does not. HPSG's feature structures are *typed*, those of LFG are not. The feature descriptions (directed graphs) are fully integrated with the phrase structure grammar in the case of HPSG, while in LFG they are intentionally separated in an autonomous level of representation in the form of a *functional structure* or f-structure. These differences lead some linguists to feel that certain types of generalization are more perspicuously stated in one framework than the other. Because LFG's functional structure is autonomous from the constituent structure whose terminal yield gives the order of words in a sentence, that functional structure can instead serve as a representation of the grammatical functions played by various components of a sentence. This makes LFG more amenable to a view of grammatical structure as motivated by its communicative function, and also provides a standard representation language for

capturing certain of the more functionally invariant properties of syntax.[1] Meanwhile, HPSG is more deeply rooted in phrase structure grammar, and thus provides a clearer representation of locality conditions.

This chapter presents a comparison of the two theories with an emphasis on contrasts between the two that arise due to differing design principles. We start with a look at how those design principles differ (Section 2), followed by point by point comparisons of syntactic issues organized by grammatical topic. Then we turn to the semantic system of LFG, beginning with a brief history by way of explaining its motivations (Section 12.1), and continuing with a presentation of the semantic theory itself. A comparison with HPSG semantics is impractical and will not be attempted here, but see Chapter **?**, Chapter **??** of this volume for an overview of HPSG semantics.

2 Design principles of HPSG and LFG

The HPSG and LFG frameworks are motivated by rather different design considerations. In order to make a meaningful comparison between the frameworks, it is essential to understand those differences.

HPSG grew out of the tradition, established by **?**, of studying the computational properties of natural language syntax as a window onto the capabilities of human cognitive processes. The advent of Generalized Phrase Structure Grammar (**?**) was an important milestone in that tradition, bringing as it did the surprising prospect that the entire range of syntactic phenomena known to exist at the time could be described with a context-free grammar (CFG). If this could be maintained it would mean that natural language is context-free in the sense of the Chomsky Hierarchy (**?**), which is an answer to the question raised by Chomsky that was very different from, and more interesting than, Chomsky's own answer. Then **?** and **?** showed that certain phenomena such as recursive cross-serial dependencies in Swiss German and Bambara exceeded the generative capacity of a context-free grammar.

Despite that development, it was nonetheless clear that languages for the most part hewed rather closely to the context-free design. Thus began the search for more powerful grammatical formalisms that would preserve the insight of a context-free grammar while allowing for certain phenomena that exceed that generative capacity.[2] HPSG grew out of this search. As **?**: 83 observe, HPSG "is still closely related to standard CFG". In fact, an HPSG grammar basically consists of constraints on local sub-trees (i.e., trees consisting of a node and her immediate daughters), which would make it a context-free grammar were it not that the nodes themselves are complex, recursively-defined feature structures. This CFG-like character of HPSG means that the framework itself embodies an interesting theory of locality.

---

[1]For example, researchers in LFG and HPSG alike have been known to compare the behavior of 'subjects' in German and Icelandic, but only LFG has a standard representation for subjects that applies to both languages, namely the SUBJ feature.

[2]There were also some problems with the GPSG theory of the lexicon, in which complement selection was assimilated to the phrase structure grammar. For discussion see **?**: Section 4.1.

The architecture of LFG is motivated by rather different concerns: an interest in typological variation, the search for linguistic universals and a desire to explain the prevailing grammatical patterns found across languages. For this reason two levels of representation are discerned, a functional structure or *f-structure* representing the internal grammatical relations in a sentence that are largely invariant across languages; and a categorial constituent structure or *c-structure* representing the external morphological and syntactic expression of those relations, which vary, often rather dramatically, across various typological varieties of language. For example, probably all (or nearly all) languages have subjects and objects, hence those relations are represented in f-structure. But languages vary as to the mechanisms for signaling subjecthood and objecthood, the three main mechanisms being word order, head-marking, and dependent-marking (**?**), hence those mechanisms are distinguished at c-structure. The word *functional* in the name of the LFG framework is a three-way pun, referring to the grammatical *functions* that play such an important role in the framework, the mathematical *functions* that are the basis for the representational formalism, and the generally *functionalist*-friendly nature of the LFG approach.

Evidence for these different underlying motivations can be found in the later proposals for further theory development by two of the principal architects of the respective theories. In some of his last work before his death, Ivan Sag worked on the development of Sign-Based Construction Grammar, a variant of HPSG in which those locality conditions are even more deeply built into the representational system (**?**). Meanwhile, Joan Bresnan had developed Optimality-Theoretic LFG (**?**), a framework in which the OT *input* is defined as an underspecified f-structure representing the universal aspects of a syntactic structure, the *candidate set* is the set of external structures mapping to that input, and cross-linguistic variation is modeled with constraint re-rankings. So while Sag sought to strengthen the locality conditions of the framework, Bresnan sought a more explicit account of universality and variation.

In the remainder of this chapter we will survey various phenomena, showing how the approaches of the different frameworks reflect those differing design motivations.

3 Phrases and endocentricity

A phrasal node shares certain grammatical features with specific daughters. In HPSG this is accomplished by means of structure-sharing (reentrancies) in the immediate dominance schemata and other constraints on local sub-trees such as the Head Feature Principle. LFG employs essentially the same mechanism for feature sharing in a local sub-tree but implements it slightly differently, so as to better address the design motivations of the theory. Each node in a phrase structure is paired with an f-structure, which is formally a set of attribute-value pairs. It is through the f-structure that the nodes of the phrase structure share features. The phrase structure is referred to as *c-structure*, for categorial or constituent structure, in order to distinguish it from f-structure. Context-free phrase structure rules license c-structures, and the c-structure elements are annotated with *functional equations* which describe the corresponding f-structure. The correspon-

dence function from c-structures to f-structures, ϕ, defines and constrains the f-structure on the basis of the equations collected from the c-structure annotations and from lexical entries of the terminal nodes.[3] For example, the phrase structure grammar in (1) and lexicon in (2) generate the tree in Figure 1.[4,5]

(1) a. S $\rightarrow$ NP VP
            $(\uparrow\text{SUBJ}) = \downarrow$ $\uparrow = \downarrow$

     b. NP $\rightarrow$ $\begin{pmatrix} \text{Det} \\ \uparrow = \downarrow \end{pmatrix}$ $\begin{matrix} \text{N} \\ \uparrow = \downarrow \end{matrix}$

     c. VP $\rightarrow$ $\begin{matrix} \text{V} \\ \uparrow = \downarrow \end{matrix}$ $\begin{pmatrix} \text{NP} \\ (\uparrow\text{OBJ}) = \downarrow \end{pmatrix}$

(2) a. *this*: Det $(\uparrow\text{PROX}) = +$
                     $(\uparrow\text{NUM}) = \text{SG}$

     b. *lion*: N $(\uparrow\text{PRED}) = \text{'lion'}$
                     $(\uparrow\text{NUM}) = \text{SG}$

     c. *roar*: V $(\uparrow\text{PRED}) = \text{'roar}\langle(\uparrow\text{SUBJ})\rangle\text{'}$
        -s: *infl* $(\uparrow\text{TENSE}) = \text{PRES}$
                     $(\uparrow\text{SUBJ}) = \downarrow$
                         $(\downarrow\text{PERS}) = 3$
                         $(\downarrow\text{NUM}) = \text{SG}$

Each node in the c-structure maps to an f-structure, that is, to a set of attribute-value pairs. Within the equations, the up and down arrows are metavariables over f-structure labels, interpreted as follows: the up arrow refers to the f-structure to which the c-structure mother node maps, and the down arrow refers to the f-structure that its own c-structure node maps to. To derive the f-structure from (1), we instantiate the metavariables to specific function names and solve for the f-structure associated with the root node (here, S). In Figure 2, the f-structure labels *f1, f2,* etc. are subscripted to the node labels. The arrows have been replaced with those labels.

Collecting all the equations from this tree and solving for f_1, we arrive at the f-structure in (3):

---

[3]Taken together the set of equations in a c-structure is called the functional description or *f-description*.

[4]In (2c) the verb is broken down into the root *roar* and third person singular suffix *-s* to show the functional equations contributed by each morpheme. However, LFG does not require that words be analyzed into morphemes. It is compatible with morpheme-based or realizational morphology or any other theory of morphology that associates word forms with grammatical features.

[5]For simplicity's sake, in this basic example we assume an NP analysis rather than a DP one (**?**). However, much recent LFG work assumes DP; see **?** and **?** for further discussion.
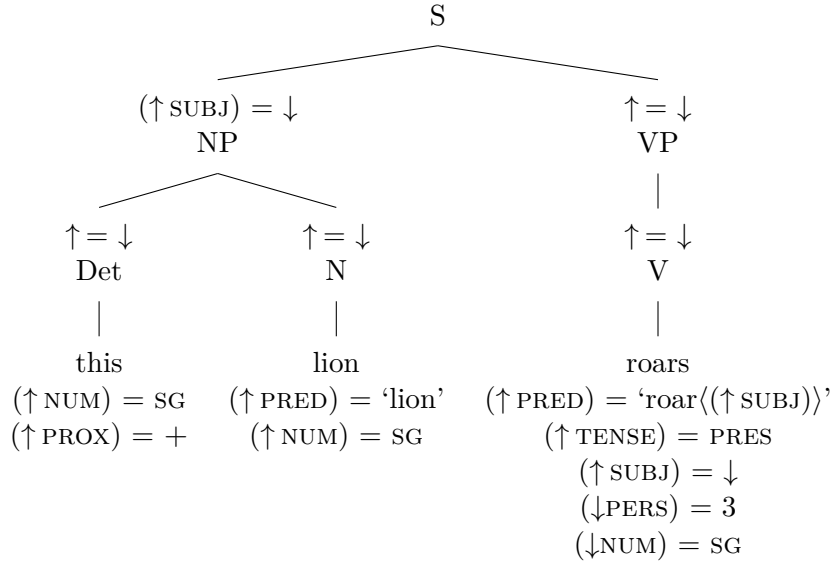
$$S$$

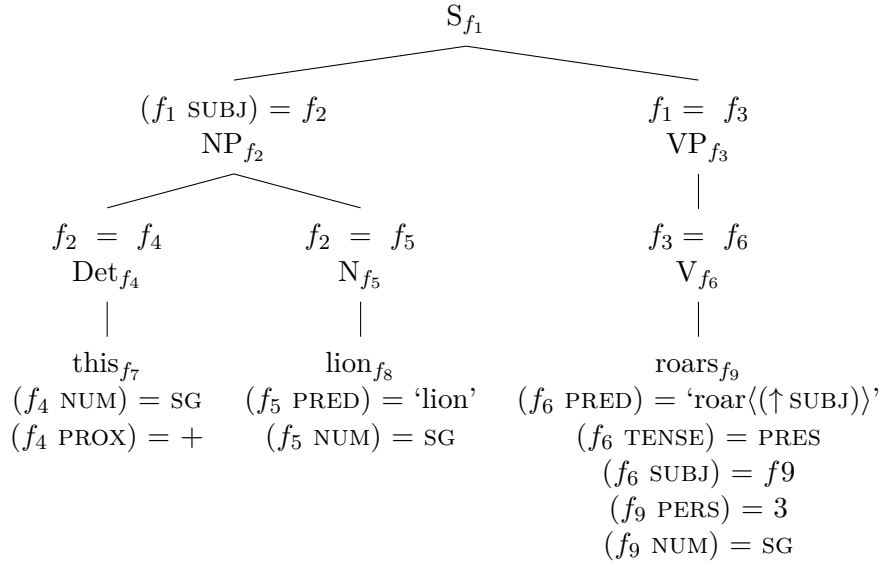$$(\uparrow \text{SUBJ}) = \downarrow \qquad\qquad \uparrow = \downarrow$$
$$\text{NP} \qquad\qquad\qquad \text{VP}$$

$$\uparrow = \downarrow \qquad \uparrow = \downarrow \qquad\qquad \uparrow = \downarrow$$
$$\text{Det} \qquad\quad \text{N} \qquad\qquad\qquad \text{V}$$

$$\text{this} \qquad\quad \text{lion} \qquad\qquad \text{roars}$$
$$(\uparrow \text{NUM}) = \text{SG} \quad (\uparrow \text{PRED}) = \text{`lion'} \quad (\uparrow \text{PRED}) = \text{`roar}\langle(\uparrow \text{SUBJ})\rangle\text{'}$$
$$(\uparrow \text{PROX}) = + \qquad (\uparrow \text{NUM}) = \text{SG} \qquad (\uparrow \text{TENSE}) = \text{PRES}$$
$$(\uparrow \text{SUBJ}) = \downarrow$$
$$(\downarrow \text{PERS}) = 3$$
$$(\downarrow \text{NUM}) = \text{SG}$$

Figure 1: C-structure in LFG

$$S_{f_1}$$

$$(f_1 \text{ SUBJ}) = f_2 \qquad\qquad f_1 = f_3$$
$$\text{NP}_{f_2} \qquad\qquad\qquad \text{VP}_{f_3}$$

$$f_2 = f_4 \qquad f_2 = f_5 \qquad\qquad f_3 = f_6$$
$$\text{Det}_{f_4} \qquad \text{N}_{f_5} \qquad\qquad \text{V}_{f_6}$$

$$\text{this}_{f_7} \qquad\quad \text{lion}_{f_8} \qquad\qquad \text{roars}_{f_9}$$
$$(f_4 \text{ NUM}) = \text{SG} \quad (f_5 \text{ PRED}) = \text{`lion'} \quad (f_6 \text{ PRED}) = \text{`roar}\langle(\uparrow \text{SUBJ})\rangle\text{'}$$
$$(f_4 \text{ PROX}) = + \qquad (f_5 \text{ NUM}) = \text{SG} \qquad (f_6 \text{ TENSE}) = \text{PRES}$$
$$(f_6 \text{ SUBJ}) = f9$$
$$(f_9 \text{ PERS}) = 3$$
$$(f_9 \text{ NUM}) = \text{SG}$$

Figure 2: Deriving the f-description

(3) f_1 $\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'lion'} \\ \text{NUM} & \text{sg} \\ \text{PERS} & 3 \\ \text{PROX} & + \end{bmatrix} \\ \text{PRED} & \text{'roar}\langle (f_1 \text{ SUBJ}) \rangle \text{'} \\ \text{TENSE} & \text{pres} \end{bmatrix}$

Since the up and down arrows refer to nodes of the local subtree, LFG annotated phrase structure rules like those in (1) can often be directly translated into HPSG immediate dominance schemata and principles constraining local subtrees. By way of illustration, let FS (for *f-structure*) be an HPSG attribute corresponding to the f-structure projection function. Then the LFG rule in (4a) (repeated from (1a) above) is equivalent to the HPSG rule in (4b):

(4) a. S $\rightarrow$ NP VP
    ($\uparrow$ SUBJ) $= \downarrow$ $\uparrow = \downarrow$

   b. S[FS ①] $\rightarrow$ NP[FS ②] VP[FS ①[SUBJ ②]]

Let us compare the two representations with respect to heads and dependents.

Taking heads first, the VP node annotated with $\uparrow = \downarrow$ is an *f-structure head*, meaning that the features of the VP are identified with those of the mother S. This effect is equivalent to the tag ① in (4b). Hence $\uparrow = \downarrow$ has an effect similar to HPSG's Head Feature Principle. However, in LFG the part of speech categories and their projections such as N, V, Det, NP, VP, DP, etc. belong to the c-structure and not the f-structure. As a consequence those features are not subject to sharing, and any principled correlations between such categories, such as the fact that N is the head of NP, V the head of VP, C the head of CP, and so on, are instead captured in an explicit version of (extended) X-bar theory applying exclusively to the c-structure (**?**). The version of extended X-bar theory in **?**: Chapter 6 assumes that all nodes on the right side of the arrow of the phrase structure rule are optional, with many unacceptable partial structures ruled out in the f-structure instead. Also not all structures need to be endocentric (i.e., not all structures have a head daughter in c-structure). The LFG category S shown in (4a) is inherently exocentric, lacking a c-structure head whose c-structure category could influence its external syntax (the f-structure head of S is the daughter with the $\uparrow = \downarrow$ annotation, here the VP). (English is also assumed to have endocentric clauses of category IP, where an auxiliary verb of category I (for Inflection) serves as the c-structure head.) S is used for copulaless clauses and also for the flat structures of non-configurational clauses in languages such as Warlpiri (see Section 10 below).

Functional projections like DP, IP, and CP are typically assumed to form a "shell" over the lexical projections NP, VP, AP, and PP (plus CP can appear over S). While this assumption is widespread in transformational approaches, its origins can be found in nontransformational research, including early LFG: CP was proposed in **?**'s Fassi-Fehri's (**?**: 141ff.) LFG treatment of Arabic; IP (the idea of the sentence as functional projection) is found in **?**'s (**?**) LFG analysis of the English auxiliary system (Falk called
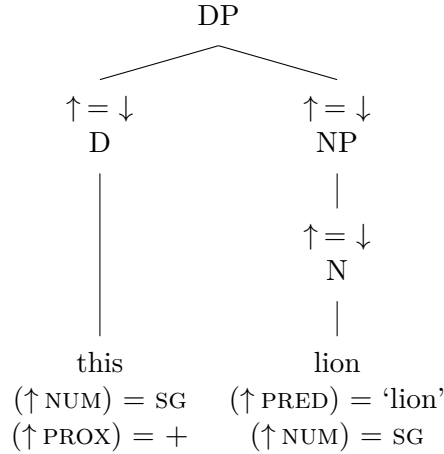
Figure 3: Functional heads in LFG

it 'MP' instead of 'IP'); and the DP originated in the nontransformational work of **?**.[6]

Extended projections are formally implemented in LFG by having the functional head (such as D) and its lexical complement (such as NP) be f-structure co-heads. See for example the DP *this lion* in Figure 3, where D, NP and N are all annotated with ↑ = ↓ hence the DP, D, NP, and N nodes all map to the same f-structure. What makes this identity possible is that function words lack a PRED feature that would otherwise indicate a semantic form.[7] Content words such as *lion* have such a feature ([PRED 'lion']), and so if the D had one as well then they would clash in the f-structure. Note more generally that the f-structure flattens out much of the hierarchical structure of the corresponding c-structure.

Complementation works a little differently in LFG from HPSG. Note that the LFG rule (4a) indicates the SUBJ grammatical function on the subject NP node, while the pseudo-HPSG rule (4b) indicates the SUBJ function on the VP functor selecting the subject. A consequence of the use of functional equations in LFG is that a grammatical relation such as SUBJ can be locally associated with its formal exponents, whether a configurational position in phrase structure (as in (1)), head-marking (agreement, see (2c)), or dependent marking (case). A subject-marking case affix such as a nominative inflection can introduce a so-called 'inside-out' functional designator, (SUBJ ↑), which requires that the f-structure of the NP or DP bearing that case ending be the value of a SUBJ attribute (**?**).[8] This aspect of LFG representations makes it convenient for

---

[6]For more on the origins of extended projections see **?**: 124–5.

[7]The attribute PRED ostensibly stands for 'predicate', but really it means something more like 'semantic content', as there are lexical items, such as proper names, which have PRED features but are not predicates under standard assumptions.

[8]Inside-out designators can be identified by their special syntax: the attribute symbol (here SUBJ) precedes instead of following the function symbol (here the metavariable ↑). They are defined as follows: for function f, attribute a and value v, $(f\,a) = v$ iff $(a\,v) = f$. If f is the f-structure

functionalist and typological work on grammatical relations.

4 Grammatical functions and valence

Grammatical functions (or *grammatical relations*) like subject and object play an important role in LFG ideology, and it is worthwhile to compare HPSG and LFG with respect to their status. Grammatical functions in LFG are best understood in the context of the break with transformational grammar that led to LFG and other alternative frameworks. **?** (*Aspects of the Theory of Syntax*) argued that while grammatical functions clearly play a role in grammar, they need not be explicitly incorporated into the grammar as such. Instead he proposed to define them in phrase structural terms: the 'subject' is the NP immediately dominated by S, the 'object' is the NP immediately dominated by VP, and so on. This theoretical assumption necessitated the use of transformations and a profusion of certain null elements: an affixal subject, for example, would have to be inserted under an NP node that is the daughter of S, and then moved to its surface position as affix; a subject that is phonologically null but anaphorically active would have to be generated in that position as well, hence the need for 'null *pro*'. Early alternatives to transformational grammar such as Relational Grammar (**?**) and LFG also sought to capture the equivalence across different alternative expressions of a single argument, but rejected the transformational model. Instead the grammar licenses an abstract representation of the 'subject', for example, essentially as a set of features. The language-specific grammar maps a predicator's semantic roles to these abstract representations (named 'subject', 'object', and so on), and also maps those representations onto the expressions in the language.

As a non-transformational theory, HPSG breaks down the relation between semantic roles and their grammatical expressions into two distinct mappings, in roughly the same way that LFG does. The intermediate representations of arguments consist of sets of features, just like in LFG. However, while LFG provides a consistent cross-linguistic representation of each grammatical function type, the HPSG framework allows the representation to vary from language to language. For example LFG identifies subjects in every language with the attribute SUBJ, while HPSG identifies the subject, if at all, as a specific element of the ARG-ST list, but which element it is can vary with the language. An English subject is usually assumed to be the first element in the list, or ARG-ST|FIRST, while a German subject is whatever element of the ARG-ST list has the nominative case feature [CASE *nom*]. Note that it would not work to represent German subjects as ARG-ST|FIRST, because in German subjectless sentences with arguments (such as datives), the first item in the ARG-ST list is a non-subject. Also, the HPSG valence feature SUBJ (or SPR 'specifier') does not represent the subject grammatical function in the LFG sense, but is instead closer to Chomsky's (1965) definition of subject as an NP in a particular phrase structural position.

This illustrates once again LFG's orientation towards cross-linguistic grammar com-

---

representing a nominal in nominative case, then (SUBJ *f*) refers to the f-structure of the clause whose subject is that nominal.

parison. HPSG practitioners can still formulate theories of 'subjects', comparing subjects in English, German, Icelandic, and so on, but there is no formal correlate of the notion 'subject' in HPSG formalism comparable to LFG's SUBJ feature. Meanwhile from the perspective of a single grammar the original motivation for adopting grammatical functions, in reaction to the pan-phrase-structural view of the transformational approach, informed both LFG and HPSG in the same way.

In LFG a lexical predicator such as a verb selects its complements via grammatical functions, which are native to f-structure, rather than using c-structure categories. A transitive verb selects a SUBJ and OBJ, which are features of f-structure, but it cannot select for the category DP because such part of speech categories belong only to c-structure. For example the verb stem *roar* in (2c) has a PRED feature whose value contains ($\uparrow$SUBJ), which has the effect of requiring a SUBJ function in the f-structure. The f-structure (shown in (3)) is built using the defining equations, as described above. Then that f-structure is checked against any *existential constraints* such as the expression ($\uparrow$SUBJ), which requires that the f-structure contain a SUBJ feature. That constraint is satisfied, as shown in (3). Moreover, the fact that ($\uparrow$SUBJ) appears in the angled brackets means that it expresses a semantic role of the 'roar' relation, hence the SUBJ value is required to contain a PRED feature, which is satisfied by the feature [PRED 'lion'] in (3).

Selection for grammatical relations instead of formal categories enables LFG to capture the flexibility in the expression of a given grammatical relation described at the end of the previous section. As noted there, in many languages the subject can be expressed either as an independent NP/DP phrase as in English, or as a pronominal affix on the verb. As long as the affix introduces a PRED feature and is designated by the grammar as filling the SUBJ relation, then it satisfies the subcategorization requirements imposed by a verb. A more subtle example of flexible expression of grammatical functions can be seen in English constructions where an argument can in principle take the form of either a DP (as in (5a)) or a clause (as in (5b)) (the examples in (5) are from (**?**: 11–12)).

(5) a. That problem, we talked about for days.
     b. That he was sick, we talked about for days.
     c. We talked about that problem for days.
     d. *We talked about that he was sick for days.

The variant of *talk* taking an *about*-PP selects neither a DP nor a CP *per se*, but rather selects the attribute string OBL$_{about}$ OBJ, describing the path to the f-structure for the object of the preposition *about*.[9]

(6) *talk*: V   ($\uparrow$PRED) = 'talk-about$\langle$($\uparrow$SUBJ)($\uparrow$OBL$_{about}$OBJ)$\rangle$'

It is not the verb but the local c-structure environment that conditions the category of that argument: the canonical prepositional object position right-adjacent to *about* can

---

[9]Following **?**: 300, this analysis assumes that the preposition *about* does not introduce a PRED feature, so it lacks its own argument structure and instead serves as a marker of an argument of the verb, functioning much like an oblique case marker.

only house a DP (compare (5c) and (5d)), while the topic position allows either DP or CP (compare (5a) and (5b)). In LFG the grammatical functions such as SUBJ and OBJ represent equivalence classes across various modes of c-structure expression.

Some HPSG approaches to filler-gap mismatches are presented in **?**, Chapter **??** of this volume, Section 9. They are essentially similar to the LFG account just presented, in that they work by allowing the preposed clause and the gap to differ in certain features. The main difference between research on this problem in the two frameworks is that in LFG the bifurcation between matching and non-matching features of filler/gap is built into the framework in the separation between f-structure and c-structure.

5 Head mobility

The lexical head of a phrase can sometimes appear in an alternative position apparently outside of what would normally be its phrasal projection. Assuming that an English finite auxiliary verb is the (category I) head of its (IP) clause, then that auxiliary appears outside its clause in a yes/no question:

(7) a. [$_{IP}$ she is mad].
     b. Is [$_{IP}$ she ___ mad]?

Transformational grammars capture the systematic relation between these two structures with a head-movement transformation that leaves the source IP structure intact, with a trace replacing the moved lexical head of the clause. The landing site of the moved clausal head is often assumed to be C, the complementizer position, as motivated by complementarity between the fronted verb and a lexical complementizer. This complementarity is observed most strikingly in German verb-second versus verb-final alternations, but is also found in other languages, including some English constructions such as the following:

(8) a. I wonder whether [$_{IP}$ she is mad].
     b. I wonder, is [$_{IP}$ she ___ mad]?
     c. *I wonder whether is she mad.

Non-transformational frameworks like HPSG and LFG offer two alternative approaches to head mobility, described in the HPSG context in **?**, Chapter **??** of this volume, Section 5 'Head movement vs. constructional approaches assuming flat structures'. Let us consider these in turn, starting with 'constructional approaches assuming flat structures'.

The sentences in (8) have been treated as displaying two distinct structures licensed by the grammar (**?** and other references in **?**, Chapter **??** of this volume, Section 5). For example, assuming ternary branching in (7b) then the subject DP *she* and predicate AP *mad* would normally be assumed to be sisters of the fronted auxiliary *is*. On that analysis the phrase structure is flattened out so that *she mad* is not a constituent. In fact for English the fronting of *is* can even be seen as a consequence of that flattening: English is a head-initial language so the two dependents *she* and *mad* are expected to

CP
┌──────────┴──────────┐
↑ = ↓   ↑ = ↓
C   IP
                ┌────────┴────────┐
          (↑ SUBJ) = ↓   ↑ = ↓
              DP   I′
                           ┌──────┴──────┐
                         ↑ = ↓   ↑ = ↓
                          I   AP
                          | |
whether   she   is   mad
(↑ FIN) = +   (↑ PRED) = 'pro' (↑ FIN) = +   (↑ PRED) = 'mad⟨(↑ SUBJ)⟩'
               (↑ PERS) = 3   (↑ SUBJ) = ↓
               (↑ NUM) = SG   (↓PERS) = 3
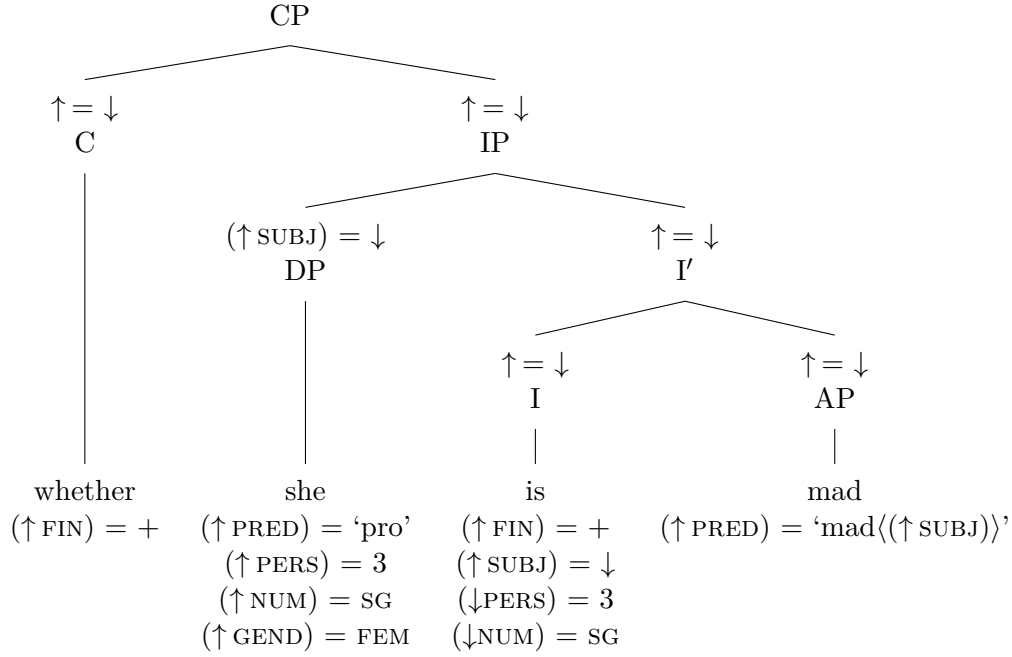   (↑ GEND) = FEM   (↓NUM) = SG

Figure 4: Functional projections in LFG

follow their selecting head *is*. This analysis is common in HPSG, and it could be cast within the LFG framework as well.

The second approach is closer in spirit to the head movement posited in transformational grammar. It is found in both HPSG and LFG but the formal implementation is rather different in the two frameworks. The HPSG 'head movement' account due to **?** posits a phonologically empty element that can function as the verb in its canonical position (such as 7a), and that empty element is structure-shared with the verb that is fronted. This treats the variation in head position similarly to non-local dependencies involving phrases. See **?**, Chapter **??** of this volume, Section 5.1 'Head movement approaches'.

The LFG version of 'head movement' takes advantage of the separation of autonomous c- and f-structures. Recall from the above discussion of the DP in Figure 3 that functional heads such as determiners, auxiliaries and complementizers do not introduce new f-structures, but rather map to the same f-structure as their complement phrases. The finite auxiliary can therefore appear in either the I or C position without this difference in position affecting the f-structure, as we will see presently. Recall also that c-structure nodes are optional and can be omitted as long as a well-formed f-structure is generated. Comparing the non-terminal structures of Figure 4 and Figure 5, the I preterminal node is omitted from the latter structure but otherwise they are identical.

The lexical equations in Figure 5 are the same as the ones in Figure 4 but are mostly omitted for clarity. Given the many ↑ = ↓ annotations, the C, I, and AP nodes (as well
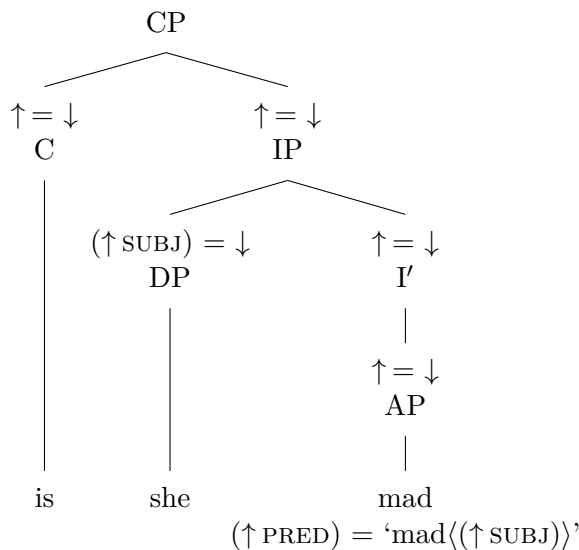
Figure 5: Head mobility in LFG

as IP and CP) all map to the same f-structure, namely the one shown in (9).

$$
(9) \quad f \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{'pro'} \\ \text{NUM} & \text{sg} \\ \text{PERS} & 3 \\ \text{GEND} & \text{fem} \end{bmatrix} \\ \text{PRED} & \text{'mad}\langle(f\,\text{SUBJ})\rangle\text{'} \\ \text{FIN} & + \end{bmatrix}
$$

The C and I positions are appropriate for markers of clausal grammatical features such as finiteness ([FIN ±]), encoded either by auxiliary verbs like finite *is* or complementizers like finite *that* and infinitival *for*: *I said that/*for she is present* vs. *I asked for/*that her to be present.* English has a specialized class of auxiliary verbs for marking finiteness from the C position, while in languages like German all finite verbs, including main verbs, can appear in a C position that is unoccupied by a lexical complementizer. Summarizing, the LFG framework enables a theory of head mobility based on the intuition that a clause has multiple head positions where inflectional features of the clause are encoded.

6 Agreement, case, and constraining equations

The basic theory of agreement is the same in LFG and HPSG (see **?**, Chapter **??** of this volume): agreement occurs when multiple feature sets arising from distinct elements of a sentence specify information about a single abstract object, so that the information must be mutually consistent (**?**). The two forms are said to agree when the values imposed by the two constraints are compatible, while ungrammaticality results when they are

incompatible. An LFG example is seen in (1) above, where the noun, determiner and verbal suffix each specify person and/or number features of the same SUBJ value.

The basic mechanism for case marking works in essentially the same way as agreement, in both frameworks: in case marking, distinct elements of a sentence specify case information about a single abstract object, hence that information must be compatible. To account for the contrast in (10a), nominative CASE equations are associated with the pronoun *she* and added to the entry for the verbal agreement suffix *-s*:

(10) a. She/*Her/*You rules.

    b. *she*: D $(\uparrow \text{PRED}) = $ 'pro'
                        $(\uparrow \text{CASE}) = \text{NOM}$
                        $(\uparrow \text{PERS}) = 3$
                        $(\uparrow \text{NUM}) = \text{SG}$
                        $(\uparrow \text{GEND}) = \text{FEM}$

    c. *her*: D $(\uparrow \text{PRED}) = $ 'pro'
                        $(\uparrow \text{CASE}) = \text{ACC}$
                        $(\uparrow \text{PERS}) = 3$
                        $(\uparrow \text{NUM}) = \text{SG}$
                        $(\uparrow \text{GEND}) = \text{FEM}$

    d. *-s*: *infl* $(\uparrow \text{TENSE}) = \text{PRES}$
                         $(\uparrow \text{SUBJ}) = \downarrow$
                             $(\downarrow \text{PERS}) = 3$
                             $(\downarrow \text{NUM}) = \text{SG}$
                             $(\downarrow \text{CASE}) = \text{NOM}$

The variant of (10a) with *her* as subject is ruled out due to a clash of CASE within the SUBJ f-structure. The variant with *you* as subject is ruled out due to a clash of PERS features. This mechanism is essentially the same as in HPSG, where it operates via the subject valence feature.

This account allows for underspecification of both the case assigner and the case-bearing element, and of both the controller and target of agreement. In English, for example, gender is marked on some pronouns but not on a verbal affix; and (nominative) case is marked on the verbal affix but not on nominals, with the exception of the pronouns. But certain case and agreement phenomena do not tolerate underspecification, and for those phenomena LFG offers an account using a *constraining equation*, a mechanism absent from HPSG and indeed ruled out by current foundational assumptions of HPSG theory. (Some early precursors to HPSG included a special feature value called ANY that functioned much like an LFG constraining equation (**?**: 36-37), but that device has been eliminated from HPSG.) The functional equations described so far in this chapter function by building the f-structure, as illustrated in (1) and (3) above; such equations are called *defining equations*. A constraining equation has the same syntax as a defining equation, but it functions by checking the completed f-structure for the presence of a feature. An f-structure lacking the feature designated by the constraining equation is ill-formed.

The following lexical entry for *she* is identical to the one in (10b) above, except that the CASE equation has been replaced with a constraining equation, notated with $=_c$.

(11) *she*: D   ($\uparrow$ PRED) = 'pro'
($\uparrow$ CASE) $=_c$ NOM
($\uparrow$ PERS) = 3
($\uparrow$ NUM) = SG
($\uparrow$ GEND) = FEM

The f-structure is built from the defining equations, after which the SUBJ field is checked for the presence of the [CASE NOM] feature, as indicated by the constraining equation. If this feature has been contributed by the finite verb, as in (10), then the sentence is predicted to be grammatical; if there is no finite verb (and there is no other source of nominative case) then it is ruled out. This predicts the following grammaticality pattern:

(12) Who won the popular vote in the 2016 election?
    a.   She did! / *Her did!
    b.   *She! / Her!

English nominative pronouns require the presence of a finite verb, here the finite auxiliary *did*. Constraining equations operate as output filters on f-structures and are the primary way to grammatically specify the obligatoriness of a form, especially under the assumption that all daughter nodes are optional in the phrase structure. As described in Section 4 above, obligatory dependents are specified in the lexical form of a predicator using existential constraints like ($\uparrow$ SUBJ) or ($\uparrow$ OBJ). These are equivalent to constraining equations in which the particular value is unspecified, but some value must appear in order for the f-structure to be well-formed.

A constraining equation for case introduced by the case-*assigner* rather than the case-bearing element, predicts that the appropriate case-bearing element must appear. A striking example from Serbo-Croatian is described by **?**: 134, who give this descriptive generalization:

(13) Serbo-Croatian Dative/Instrumental Case Realization Condition.

      If a verb or noun assigns dative or instrumental case to an NP, then that case must be morphologically realized by some element within the NP.

In Serbo-Croatian most common nouns, proper nouns, adjectives, and determiners are inflected for case. An NP in a dative position must contain at least one such item morphologically inflected for dative case, and similarly for instrumental case. The verb *pokloniti* 'give' governs a dative object, such as *ovom studentu* in (14a). But a quantified NP like *ovih pet studenata* 'these five students' has invariant case, namely genitive on the determiner and noun, and an undeclinable numeral *pet* 'five'. Such a quantified NP can appear in any case position, except when it fails to satisfy the condition in (13), such as this dative position (**?**: 125):

(14) a. pokloniti knjige   ovom   studentu
          give.INF   books.ACC this.DAT.SG student.DAT.SG
          'to give books to this student'

      b. * pokloniti knjige   [ovih   pet studenata]
          give.INF   books.ACC this.GEN.PL five student.GEN.PL
          intended: 'to give books to these five students'

Similarly, certain foreign names such as *Miki* and loanwords such as *braon* 'brown',
'brunette' are undeclinable, and can appear in any case position, except those ruled out
by (13). Thus the dative example in (15a) is unacceptable unless the inflected possessive
adjective *mojoj/mojom* 'my' appears. When the possessive adjective realizes the case
feature, it is acceptable. In (15b) we contrast the undeclined loan word *braon* 'brown'
with the inflected form *lepoj* 'beautiful'. The example is acceptable only with the inflected
adjective (**?**: 134).

(15) a. Divim   se   *(mojoj)   Miki.
          admire.1SG REFL   my.DAT.SG Miki
          'I admire (my) Miki.'

      b. Divim   se   *braon / lepoj   Miki.
          admire.1sg REFL   brown / beautiful.DAT.SG Miki
          intended: 'I admire brunette/ beautiful Miki.'

This complex distribution is captured simply by positing that the dative (and instru-
mental) case assigning equations on verbs and nouns, such as the verbs *pokloniti* and
divim in the above examples, are constraining equations:

(16) $(\uparrow \text{OBL}_{dat} \text{ CASE}) =_c \text{DAT}$

Any item in dative form within the NP, such as *ovom* or *studentu* in (14a) or *mojoj* or
lepoj in (15), could introduce the [CASE DAT] feature that satisfies this equation, but if
none appears then the sentence fails. In contrast, other case-assigning equations (e.g.,
for nominative, accusative, or genitive case, or for cases assigned by prepositions) are
defining equations, which therefore allow the undeclined NPs to appear. This sort of
phenomenon is easy to capture using an output filter such as a constraining equation,
but rather difficult otherwise. See **?** for further examples and discussion.

7 Agreement and affixal pronouns

Agreement inflections that include the person feature derive historically from incor-
porated pronominal affixes. Distinguishing between agreement markers and affixal pro-
nouns can be a subtle and controversial matter. LFG provides a particular formal device
for representing this distinction within the f-structure: a true pronoun, whether affixal
or free, introduces a semantic form (formally a PRED feature) with the value 'pro', while

an agreement inflection does not. For example, **?** argue that the Chicheŵa (Bantu) object marker (OM) is an incorporated pronoun, while the subject marker (SM) alternates between agreement and incorporated pronoun, as in this example:

(17) Njûchi zi-ná-wá-lum-a   a-lenje.
      10.bee 10.SM-PST-2.OM-bite-FV 2-hunter
      'The bees bit them, the hunters.'

According to **?**, the class 2 object marker *wá-* is a pronoun, so the phrase *alenje* 'the hunters' is not the true object but rather a postposed topic cataphorically linked to the object marker, with which it agrees in noun class (a case of *anaphoric agreement*). Meanwhile, the class 10 subject marker *zi-* alternates: when an associated subject NP (*njûchi* 'bees' in (17)) appears then it is a grammatical agreement marker, but when no subject NP appears then it functions as a pronoun. This is captured in LFG with the simplified lexical entries in (18):

(18) a. *lum*: V   ($\uparrow$ PRED) = 'bite$\langle(\uparrow$ SUBJ$)(\uparrow$ OBJ$)\rangle$'
      b. *wá-*: *Aff*   ($\uparrow$ OBJ GEND) = 2
                       ($\uparrow$ OBJ PRED) = 'pro'
      c. *zi-*: *Aff*   ($\uparrow$ SUBJ GEND) = 10
                       (($\uparrow$ SUBJ PRED) = 'pro')

The PRED feature in (18b) is obligatory while that of (18c) is optional, as indicated by the parentheses around the latter. These entries interact with the grammar in the following manner. The two grammatical functions governed by the verb in (18a) are SUBJ and OBJ (the *governed* functions are the ones designated in the predicate argument structure of a predicator). According to the *Principle of Completeness*, a PRED feature must appear in the f-structure for each governed grammatical function that appears within the angle brackets of a predicator (indicating assignment of a semantic role). By the uniqueness condition it follows that there must be *exactly one* PRED feature, since a second such feature would cause a clash of values.[10]

The OM *wá-* introduces the [PRED 'pro'] into the object field of the f-structure of this sentence; the word *alenje* 'hunters' introduces its own PRED feature with value 'HUNTERS', so it cannot be the true object, and instead is assumed to be in a topic position. **?** note that the OM can be omitted from the sentence, in which case the phrasal object (here, *alenje*) is fixed in the immediately post-verbal position, while that phrase can alternatively be preposed when the OM appears. This is explained by assuming that the post-verbal position is an OBJ position, while the adjoined TOPIC position is more flexible.

The subject *njûchi* can be omitted from (17), yielding a grammatical sentence meaning 'They (some plural entity named with a class 10 noun) bit them, the hunters.' The optional PRED feature equation in (18c) captures this pro-drop property: when the equation appears then a phrase such as *njûchi* cannot appear in the subject position,

---

[10]Each PRED value is assumed to be unique, so that two 'pro' values cannot unify.

since this would lead to a clash of PRED values ('pro' versus 'bees'); but when the equation is not selected then *njûchi* must appear in the subject position in order for the f-structure to be complete.

The diachronic process in which a pronominal affix is reanalyzed as agreement has been modeled in LFG as the historic loss of the PRED feature, along with the retention of the pronoun's person, number, and gender features (**?**). The anaphoric agreement of the older pronoun with its antecedent then becomes reanalyzed as grammatical agreement of the inflected verb with an external nominal. Finer transition states can also be modeled in terms of selective feature loss. Clitic doubling can be modeled as optional loss of the PRED feature with retention of some semantic vestiges of the pronominal, such as specificity of reference.

The LFG analysis of affixal pronouns and agreement inflections can be approximated in HPSG by associating the former but not the latter with pronominal semantics in the CONTENT field. However, because HPSG lacks output filters like the LFG Principle of Completeness, it is more difficult for the HPSG framework to directly capture the complementarity between the appearance of a pronominal affix and an analytic phrase filling the same grammatical relation.

8 Lexical mapping

LFG and HPSG both adopt *lexical approaches to argument structure* in the sense of **?**: a verb or other predicator is equipped with a valence structure indicating the grammatical expression of its semantic arguments as syntactic dependents. Both frameworks have complex systems for mapping semantic arguments to syntactic dependents that are designed to capture prevailing semantic regularities within a language and across languages. The respective systems differ greatly in their notation and formal properties but it is unclear whether there are any theoretically interesting differences, such as types of analysis that are available in one but not the other. This section identifies some of the most important analogues across the two systems, namely LFG's Lexical Mapping Theory (LMT; (**?**: Chapter 14)) and the theory of macro-roles proposed by Davis and Koenig for HPSG (**??**; see **?**, Chapter **??** of this volume).[11]

In Lexical Mapping Theory, the argument structure is a list of a verb's argument slots, each labeled with a thematic role type such as Agent, Instrument, Recipient, Patient, Location, and so on, in the tradition of Charles Fillmore's *Deep Cases* (**??**) and Pāṇini's *kārakas* (**?**). The ordering is determined by a thematic hierarchy that reflects priority for subject selection.[12] The thematic role type influences a further classification by the features $[\pm o]$ and $[\pm r]$ that conditions the mapping to syntactic functions (this version is from **?**: 331):

---

[11]A recent alternative to LMT based on Glue Semantics has been developed by **?** and **?**, based on the mapping theory of **?**. It preserves the monotonicity of LMT, discussed below, but uses Glue Semantics to model argument realization and valence alternations.

[12]The particular ordering proposed in (**?**: 23) and (**?**: 329) is the following: *agent* ≻ *beneficiary* ≻ *experiencer/goal* ≻ *instrument* ≻ *patient/theme* ≻ *locative*

(19) **Semantic classification of argument structure roles for function**:

patientlike roles:
$$\begin{array}{c}\theta\\{[-r]}\end{array}$$

secondary patientlike roles:
$$\begin{array}{c}\theta\\{[+o]}\end{array}$$

other semantic roles:
$$\begin{array}{c}\theta\\{[-o]}\end{array}$$

The features $[\pm r]$ (thematically *restricted*) and $[\pm o]$ (*objective*) cross-classify grammatical functions: subject is $[-r, -o]$, object is $[-r, +o]$, obliques are $[+r, -o]$ and restricted objects are $[+r, +o]$. A monotonic derivation (where feature values cannot be changed) starts from the argument list with the Intrinsic Classification (*I.C.* in example (20) below), then morpholexical operations such as passivization can suppress a role (not shown), then the thematically highest role (such as the Agent), if $[-o]$, is selected as Subject (*Sbj.* in example (20)) and then any remaining features receive positive values by default (*Def.* in example (20)).

(20) Derivation of *eat* as in *Pam ate a yam.*

a-structure:	*eat*	<	*ag*	*th*	>
	I.C.		$[-o]$	$[-r]$	
	Sbj.		$[-r]$		
	Def.			$[+o]$	
			|	|	
f-structure:			SUBJ	OBJ	

In the macro-role theory formulated for HPSG, the analogues of $[-o]$ and $[-r]$ are the macro-roles Actor (ACT) and Undergoer (UND), respectively. The names of these features reflect large general groupings of semantic role types, but there is not a unique semantic entailment such as "agency" or "affectedness" associated with each of them. Actor (ACT) and Undergoer (UND), name whatever semantic roles map to the subject and object, respectively, of a transitive verb.[13] On the semantic side they are disjunctively defined: x is the Actor and y is the Undergoer iff 'x causes a change in y, or x has a notion of y, or ...' (quoted from **?**, Chapter **??** of this volume, example (**??**)). Such disjunctive definitions are the HPSG analogues of the Lexical Mapping Theory 'semantic classifications' shown in (19) above. In the HPSG macro-role system, linking constraints dictate that the ACT argument maps to the first element of ARG-ST, and that the UND argument maps to some nominal element of ARG-ST (21) and (22) are from **?**, Chapter **??** of this volume, examples (**??**) and (**??**); each set of dots in the list value of ARG-ST represents zero or more list items):

---

[13]Note, for example, that within this system the 'Undergoer' argument of the English verb *undergo*, as in *John underwent an operation*, is the object, and not the subject as one might expect if being an Undergoer involved actually undergoing something.

(21) $\begin{bmatrix} \text{CONTENT|KEY} & \begin{bmatrix} \text{ACT} & \boxed{1} \end{bmatrix} \\ \text{ARG-ST} & \left\langle \text{NP}_{\boxed{1}}, \dots \right\rangle \end{bmatrix}$

(22) $\begin{bmatrix} \text{CONTENT|KEY} & \begin{bmatrix} \text{UND} & \boxed{2} \end{bmatrix} \\ \text{ARG-ST} & \left\langle \dots, \text{NP}_{\boxed{2}}, \dots \right\rangle \end{bmatrix}$

The first element of ARG-ST maps to the subject of an active voice verb, so (21)–(22) imply that the subject is the ACT if there is one, and otherwise it is the UND. Similarly, in Lexical Mapping Theory as described above, the subject is the $[-o]$ highest argument, if there is one, and otherwise it is the $[-r]$ argument. In this simple example we can see how the two systems accomplish exactly the same thing. A careful examination of more complex examples might point up theoretical differences, but it seems more likely that the two systems can express virtually the same set of mappings.

In LFG the *argument structure* (or *a-structure*) contains the predicator and its argument roles classified and ordered by thematic role type and further classified by Intrinsic Classification. It is considered a distinct level of representation, along with c-structure and f-structure. As a consequence the grammar can make reference to the initial item in a-structure, such as the agent (*ag*) in (20), which is considered the 'most prominent' role and often called the *a-subject* ('argument structure subject') in LFG parlance. To derive the passive voice mapping, the a-subject is suppressed in a morpholexical operation that crucially takes place before the subject is selected:

(23)   Derivation of *eaten* as in *A yam was eaten (by Pam)*.

a-structure:	*eat*	<	*ag*	*th*	>
	I.C.		$[-o]$	$[-r]$	
	passive		Ø		
	Sbj.			$[-o]$	
				$\mid$	
f-structure:				SUBJ	

(The optional *by*-phrase is considered to be an adjunct referring to the passivized a-subject.) Note that the passive alternation is *not* captured by a procedural rule that replaces one grammatical relation (such as OBJ) with another (such as SUBJ). The mapping from word strings to f-structures in LFG is monotonic, in the sense that information cannot be destroyed or changed. As a result the mapping between internal and external structures is said to be transparent in the sense that the grammatical relations of parts of the sentence are preserved in the whole (for discussion of this point, see **?**: Chapter 5). In early versions of LFG, monotonicity was assumed for the syntax proper, while destructive procedures were permitted in the lexicon. This was canonized in the *Principle of Direct Syntactic Encoding*, according to which all grammatical relation changes are lexical (**?**). At that time an LFG passive rule operated on fully specified predicate argument structures, replacing OBJ with SUBJ, and SUBJ with an OBL$_{by}$ or an existentially bound variable. The advent of LMT brought monotonicity to the lexicon as well.
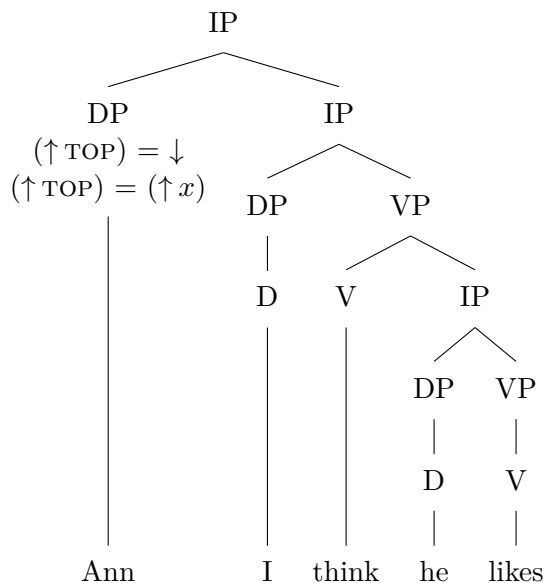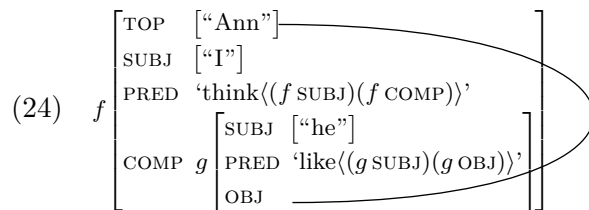
IP

DP   IP

$(\uparrow \text{TOP}) = \downarrow$

$(\uparrow \text{TOP}) = (\uparrow x)$

DP   VP

D   V   IP

DP   VP

D   V

Ann   I   think   he   likes

Figure 6: Long distance dependencies in LFG

9 Long distance dependencies

In LFG a long distance dependency is modeled as a reentrancy in the f-structure. The HPSG theory of long distance dependencies is based on that of GPSG (**?**) and uses the percolation of a 'slash' feature through the constituent structure. But LFG and HPSG accounts are essentially very similar, both working by decomposing a long distance dependency into a series of local dependencies. As we will see, there are nevertheless some minor differences with respect to what hypothetical extraction patterns can be expressed.

Both frameworks allow accounts either with or without gaps: regarding LFG see **?** for gaps and **?** for gapless; regarding HPSG see **?** for gaps and **?** for gapless. Gaps have been motivated by the (controversial) claim that the linear position of an empty category matters for the purpose of weak crossover and other binding phenomena (**?**: Chapter 9). In this section we compare gapless accounts.

LFG has two grammaticalized discourse functions, TOP (topic) and FOC (focus). A sentence with a left-adjoined topic position is depicted in Figure 6. The topic phrase *Ann* serves as the object of the verb *like* within the clausal complement of *think*. This dependency is encoded in the second equation annotating the topic node, where the variable x ranges over strings of attributes representing grammatical functions such as SUBJ, OBJ, OBL, or COMP. These strings describe paths through the f-structure. In this example x is resolved to the string COMP OBJ, so this equation has the effect of adding to the f-structure in (24) the curved line representing token identity.

(24) f $\begin{bmatrix} \text{TOP} & [\text{``Ann''}] \\ \text{SUBJ} & [\text{``I''}] \\ \text{PRED} & \text{`think}\langle(f\,\text{SUBJ})(f\,\text{COMP})\rangle\text{'} \\ \text{COMP} & g \begin{bmatrix} \text{SUBJ} & [\text{``he''}] \\ \text{PRED} & \text{`like}\langle(g\,\text{SUBJ})(g\,\text{OBJ})\rangle\text{'} \\ \text{OBJ} \end{bmatrix} \end{bmatrix}$

HPSG accounts are broadly similar. One HPSG version relaxes the requirement that the arguments specified in the lexical entry of a verb or other predicator must all appear in its valence lists. Arguments are represented by elements of the ARG-ST list, so the list for the verb *like* contains two NPs, one each for the subject and object. In a sentence with no extraction, those ARG-ST list items map to the valence lists, the first item appearing in SUBJ and any remaining ones in COMPS. To allow for extraction, one of those ARG-ST list items is permitted to appear on the SLASH list instead. The SLASH list item is then passed up the tree by means of strictly local constraints, until it is bound by the topicalized phrase (see **?**, Chapter **??** of this volume and **?**).

The LFG dependency is expressed in the f-structure, not the c-structure. **?**: Chapter 2 note that this allows for category mismatches between the phrases serving as filler and those in the canonical, unextracted position. This was discussed in Section 4 and illustrated with example (5) above.

Constraints on extraction such as accessibility conditions and island constraints can be captured in LFG by placing constraints on the attribute string x (**?**).[14] If subjects are not accessible to extraction then we stipulate that SUBJ cannot be the final attribute in x; if subjects are islands then we stipulate that SUBJ cannot be a non-final attribute in x. If the f-structure is the only place such constraints are stated then this makes the interesting (but unfortunately false; see presently) prediction that the theory of extraction cannot distinguish between constituents that map to the same f-structure. For example, as noted in Section 5, function words like determiners and their contentful sisters like NP are usually assumed to be f-structure co-heads, so the DP *the lion* maps to the same f-structure as its daughter *lion* (see diagram Figure 3). This predicts that if the DP can be extracted then so can the NP, but of course that is not true:

(25)   a.   The lion, I think she saw.
       b.   *Lion, I think she saw the.

These two sentences have exactly the same f-structures, so any explanation for the contrast in acceptability must involve some other level. For example, one could posit that the phrase structure rules can introduce some items obligatorily (see (**?**: 239)), such as an obligatory sister of the determiner *the*.[15]

---

[14] **?** shows that LFG's *off-path constraints* (**?**: 225–230) can even capture quite strict locality conditions on extraction, in the spirit of *successive cyclicity* in movement-based accounts (**??**), but without movement.

[15] This would depart from the assumption that all nodes are optional, adopted in **?**.

10 Nonconfigurationality

Some languages make heavy use of case and agreement morphology to indicate the grammatical relations, while allowing very free ordering of words within the clause. Such radically nonconfigurational syntax receives a straightforward analysis in LFG, due to the autonomy of functional structure from constituent structure. Indeed the notion that phrasal position and word-internal morphology can be functionally equivalent is a foundational motivation for that separation of structure and function, as noted in Section 2 above. As **?**: 5 observe, "The idea that words and phrases are alternative means of expressing the same grammatical relations underlies the design of LFG, and distinguishes it from other formal syntactic frameworks."

The LFG treatment of nonconfigurationality will be illustrated with the simplified analysis of the clausal syntax of Warlpiri, a Pama-Nyungan language of northern Australia (from **?**: 352-3). The following example gives three of the many possible grammatical permutations of words, all expressing the same truth-conditional content.

(26) a. Kurdu-jarra-rlu wita-jarra-rlu   ka-pala   maliki   wajilipi-nyi.
       child-DUAL-ERG small-DUAL-ERG PRES-DUAL dog-ABS chase-NONPAST
       'The two small children are chasing the dog.'

   b. Kurdu-jarra-rlu ka-pala   maliki   wajilipi-nyi   wita-jarra-rlu.
       child-DUAL-ERG PRES-DUAL dog-ABS chase-NONPAST small-DUAL-ERG
       'The two small children are chasing the dog.'

   c. Maliki   ka-pala   kurdu-jarra-rlu wajilipi-nyi   wita-jarra-rlu.
       dog-ABS PRES-DUAL child-DUAL-ERG chase-NONPAST small-DUAL-ERG
       'The two small children are chasing the dog.'

The main constraint on word order is that the auxiliary (here, the word *kapala*) must immediately follow the first daughter of S, where that first daughter can be any other word in the sentence, or else a multi-word NP as in (26a). Apart from that constraint, all word orders are possible. Any word or phrase in the clause can precede the auxiliary, and the words following the auxiliary can appear in any order.

The LFG analysis of these sentences works by directly specifying the auxiliary-second constraint within the c-structure rule in (27a).[16] Then the lexical entries directly specify the case, number, and other grammatical features of the word forms, including case-assignment properties of the verb (see (29)). The framework does the rest, licensing all and only grammatical word orderings and generating an appropriate f-structure (see (30)).

(27) a. S   →   X   (Aux)   X*   where X = NP or V

---

[16]The c-structure is slightly simplified for illustrative purposes. In the actual c-structure proposed for Warlpiri the second position auxiliary is the c-structure head (of IP) taking a flat S as its right sister (see **?**: 225). Because the IP functional projection and its complement S map to the same f-structure (as discussed in Section 5 above), the analysis presented here works in exactly the same way regardless of whether this extra structure appears.

b. NP → N*

(28) a. Assign (↑SUBJ) = ↓ or (↑OBJ) = ↓ freely to NP.

b. Assign ↑ = ↓ to N, V and Aux.

(29) a. *kurdu-jarra-rlu*: N   (↑PRED) = 'child'
                                  (↑NUM) = DUAL
                                  (↑CASE) = ERG

b. *maliki*: N   (↑PRED) = 'dog'
                                  (↑NUM) = SG
                                  (↑CASE) = ABS

c. *wita-jarra-rlu*: N   (↑ADJ PRED) = 'small'
                                  (↑NUM) = DUAL
                                  (↑CASE) = ERG

d. *wajilipi-nyi*: V   (↑PRED) = 'chase⟨(↑SUBJ)(↑OBJ)⟩'
                                  (↑TENSE) = NONPAST
                                  (↑SUBJ CASE) = ERG
                                  (↑OBJ CASE) = ABS

e. *ka-pala*: Aux   (↑ASPECT) = PRESENT.IMPERFECT
                                  (↑SUBJ NUM) = DUAL

(30) f
$$
\begin{bmatrix}
\text{PRED} & \text{'chase } \langle (f\,\text{SUBJ})(f\,\text{COMP})\rangle\text{'} \\
\text{SUBJ} & \begin{bmatrix} \text{PRED 'child'} \\ \text{NUM dual} \\ \text{CASE erg} \\ \text{ADJ } [\text{PRED 'small'}] \end{bmatrix} \\
\text{OBJ} & \begin{bmatrix} \text{PRED 'dog'} \\ \text{NUM sg} \\ \text{CASE abs} \end{bmatrix} \\
\text{TENSE} & \text{nonpast} \\
\text{ASPECT} & \text{present.imperfect}
\end{bmatrix}
$$

The functional annotations on the NP nodes (see (28)) can vary as long as they secure Completeness and Coherence, given the governing predicate. In this case the main verb is transitive, so SUBJ and OBJ must be realized, each with exactly one PRED value.[17] The noun *wita* 'small' is of category N, as Warlpiri does not distinguish between nouns and adjectives; but it differs functionally from the nouns for 'child' and 'dog' in that it

---

[17] By the Principle of Completeness, both SUBJ and OBJ appear in the f-structure; by the Principle of Coherence no other governable grammatical function appears in the f-structure.

modifies another noun. This is indicated by embedding its PRED feature under the ADJ ('adjunct') attribute (see the first equation in (29c)).[18]

Comparing HPSG accounts of nonconfigurationality is instructive. Two HPSG approaches are described in **?**, Chapter **??** of this volume: the *domain order* approach (**?**, described in **?**, Chapter **??** of this volume, Section 6.2 'Absolutely free'); and the noncancellation approach (**?**; described in **?**, Chapter **??** of this volume, Section 7 'Free constituent order languages without order domains'). Interestingly, they both involve the adoption of a separation of levels analogous to LFG's c-structure/f-structure divide.

In the *domain order* sbj]domain order approach the words of Warlpiri are combined into constituent structures resembling those of a configurational language like English. For example, in a domain order analysis of the Warlpiri sentences in (26), as well as all other acceptable permutations of those word forms, the words for 'two small children' together form an NP constituent. However, a domain feature DOM lists the phonological forms of the words in each constituent, and allows that list order to vary freely relative to the order of the daughters. This effective shuffling of the DOM list applies recursively on the nodes of the tree, up to the clausal node. It is the DOM list order, not the order of the terminal nodes, that determines the order of words for pronunciation of the sentence. That function of the DOM feature is carried out in LFG by the c-structure. Meanwhile, the phrase structure tree in the domain order account is reinterpreted as similar to LFG's f-structure, displaying the logical relations in the sentence rather than the word order (compare the tree diagram in **?**, Chapter **??** of this volume, Fig. 11, with the f-structure in (30) above).

The *non-cancellation* sbj]non-cancellation approach also effects a separation into correlates of c-structure and f-structure, and is even more LFG-like. In essence an f-structure is added to HPSG in the form of a feature, much like the FS feature in the pseudo-HPSG rule in (4b) above. Instead of FS, that feature is called COMPS and has a list value. Unlike the valence feature normally called COMPS, items of this COMPS feature are not canceled from the list (and unlike ARG-ST, this feature is shared between a phrase and its head daughter, so it appears on non-terminal nodes). The items of that list are referenced by their order in the list. Special phrasal types define grammatical relations between a non-head daughter and an item in the COMPS list of the head daughter. These phrasal types are equivalent to LFG annotated phrase structure rules. For example, suppose the second item in COMPS ('*2nd-comp*') is the object. Then *head-2nd-comp-phrase* in **?**: 12 is equivalent to an LFG rule where the non-head daughter has the OBJ annotation (see (28a)). Since the list item is not canceled from the list, it remains available for other items to combine and to modify the object, using a different phrasal type.

These analyses are not equivalent, and an HPSG incorporating the mechanisms needed for the domain order or the non-cancellation approach to non-configurationality is not thereby equivalent to an LFG. But such mechanisms do bring HPSG closer to LFG by relying on a level of structure that is autonomous from the constituent structure

---

[18]The value of LFG's ADJ feature is normally a set of f-structures, as there can be multiple adjuncts, in fact indefinitely many. We simplify things here by assuming that the value is just an f-structure, rather than a set of f-structures. The standard approach would require the relevant equation to instead be ($\uparrow$ ADJ $\in$ PRED) = 'small'.

responsible for the grouping and ordering of words for pronunciation.

11 Control and raising

Raising and control (equi) words are treated in virtually the same way in LFG and HPSG: a subject control word (such as *hope* in (31)) specifies that its subject is (also) the subject of its predicate complement.

(31) Pam hopes to visit Fred.
    a. *hope*: ($\uparrow$ PRED) = 'hope$\langle$($\uparrow$ SUBJ)($\uparrow$ XCOMP)$\rangle$'
                      ($\uparrow$ SUBJ) = ($\uparrow$ XCOMP SUBJ)
    b. *hope*: [ARG-ST $\langle$ $\boxed{1}$NP, VP[*inf*, SUBJ $\langle$ $\boxed{1}$ $\rangle$]$\rangle$]

The LFG entry for *hope* in (31a) contains the grammatical function XCOMP ('open complement'), the function reserved for predicate complements such as the infinitival phrase *to visit Fred*. The control equation specifies that its subject is identical to the subject of the verb *hope*; the tag $\boxed{1}$ plays the same role in the simplified HPSG entry in (31b).

    The f-structure for (31a) is shown here:

(32) f $\begin{bmatrix} \text{SUBJ} & [\text{"Pam"}] \\ \text{PRED} & \text{'hope}\langle(f\,\text{SUBJ})(f\,\text{XCOMP})\rangle\text{'} \\ \text{XCOMP} & g\begin{bmatrix} \text{PRED} & \text{'visit}\langle(g\,\text{SUBJ})(g\,\text{OBJ})\rangle\text{'} \\ \text{SUBJ} & \underline{\hspace{1cm}} \\ \text{OBJ} & [\text{"Fred"}] \end{bmatrix} \end{bmatrix}$

One interesting difference between the two frameworks is that the HPSG representation allows only for control (or raising) of subjects and not complements. More precisely, it allows for control of the outermost or final dependent to be combined with the verbal projection. This is because of the list cancellation regime that operates with valence lists (on non-cancellation theories see Section 10). The expression VP in (31b) represents an item with an empty COMPS list. In a simple English clause the verb combines with its complement phrases to form a VP constituent, with which the subject is then combined to form a clause. Assuming the same order of combination in the control structure, it is not possible to control the complement of a structure that contains a structural subject, as in (33a):

(33) a. * Fred hopes Pam to visit.
    b. Fred hopes to be visited by Pam.
    c. ?! ($\uparrow$ SUBJ) = ($\uparrow$ XCOMP OBJ)

The intended meaning would be that of (33b). The passive voice is needed in order to make the intended controllee (*Fred*) the subject of *visit* and thus available to be controlled. This restriction of controllees to subjects follows from the HPSG theory, while in LFG it follows only if control equations like the one in (33c) are systematically excluded.

This raises the question of whether the restriction to subject controllees is universal. In fact it appears that some languages allow the control of non-subjects, but it is still unclear whether such languages justify equations such as (33c). For example, **?** shows that Tagalog has two types of control relation. In the more specialized type, which occurs only with a small set of verbs or in a special construction in which the downstairs verb appears in non-volitive mood, both the controller and controllee must be subjects. Kroeger analyzes this type as functional control using a control equation like the one in (31a). In the more common type of Tagalog control, the controllee must be the Actor argument, while the grammatical relations of controllee and controller are not restricted. (Tagalog has a rich voice system, often called a focus marking system, regulating which argument of a verb is selected as its subject.) This latter type of Tagalog control is defined on argument structure (Actors, etc.), so a-structure rather than f-structure is appropriate for representing the control relations.

Instead of functional control, Kroeger analyzes this latter type of control within LFG as *anaphoric control*, in which the controllee has the status of a null pronoun whose antecedent is the controller. In anaphoric control the f-structures of the controller and controllee are not identified, and instead each has its own PRED feature. An anaphoric control analysis of (31) would involve a structure resembling (32) except that the curved line would be eliminated and the SUBJ of *visit* would contain the feature [PRED 'PRO']. In the HPSG analogue of anaphoric control, an element of a valence list represents a null pronoun in the following sense: the rules mapping the valence lists to the phrase structure do not project such an element, and it is accorded the semantic and syntactic properties of a pronoun, such as definiteness and/or specificity, and binding features.

12 Semantics

HPSG was conceived from the start as a theory of the *sign* (**?**), wherein each constituent is a pairing of form and meaning. So semantic representation and composition was built into HPSG (and the subsequent sister framework of Sign-Based Construction Grammar; **?**), as reflected in the title of the first HPSG book (**?**), *Information-Based Syntax and Semantics*. LFG was not founded as a theory that included semantics, but a semantic component was developed for LFG shortly thereafter (**?**). The direction of semantics for LFG changed some ten years later and the dominant tradition is now Glue Semantics (**?????**).

This section presents a basic introduction to Glue Semantics (Glue); this is necessary to fully understand a not insignificant portion of LFG literature of the past fifteen years, which interleaves LFG syntactic analysis with Glue semantic analysis. The section is not meant as a direct comparison of LFG and HPSG semantics, for two reasons. First, as explained in the previous paragraph, HPSG is inherently a theory that integrates syntax and semantics, but LFG is not; the semantic module that Glue provides for LFG can easily be pulled out, leaving the syntactic component exactly the same. Second, as will become clear in the next section, at a suitable level of abstraction, Glue offers an underspecified theory of semantic composition, in particular scope relations, which is

also the goal of what is arguably the most influential recent HPSG semantic approach, Minimal Recursion Semantics (**?**). But beyond observing this big-picture commonality, comparison of Glue and MRS would require a chapter in its own right. Our goal is to present enough of Glue Semantics for readers to grasp the main intuitions behind it, without presupposing much knowledge of formal semantic theory. The references listed at the end of the previous paragraph (especially **?**) are good places to find additional discussion and references.

The rest of this section is organized as follows. In Section 12.1 we present some more historical background on semantics for LFG and HPSG. In Section 12.2, we present Glue Semantics, as a general compositional system in its own right. Then, in Section 12.3 we look at the syntax–semantics interface with specific reference to an LFG syntax. For further details on semantic composition and the syntax–semantics interface in constraint-based theories of syntax, see **?**, Chapter **??** of this volume for semantics for HPSG and **?** for Glue Semantics for LFG.

12.1 Brief history of semantics for LFG and HPSG

Various theories of semantic representation have been adopted by the different non-transformational syntactic frameworks over the years. The precursor to HPSG, GPSG, was paired by its designers with a then fairly standard static Montogovian semantics (**?**), but GPSG itself was subsequently adopted as the syntactic framework used by **?**: 9 for Discourse Representation Theory, a dynamic theory of semantics. Initial work on semantics for LFG also assumed a Montogovian semantics (**??**). But with the increasing interest in Situation Semantics (**?**) in the 1980s at Stanford University and environs (particularly SRI International and Xerox PARC), the sites of the foundational work on both HPSG and LFG, both frameworks also became associated with Situation Semantics. In the case of LFG, this association was not foundational, manifesting primarily in the form of **?**, and soon waned. In contrast, as noted above, HPSG is a theory of signs and therefore had semantics incorporated into the theory from the outset. Moreover, the chosen semantic theory was Situation Semantics, and this also carried over into the second main HPSG book (**?**) and further (**?**).

Beginning in the nineties, the focus subsequently shifted in new directions due to a new interest in computationally tractable theories of the syntax–semantics interface, to support efforts at large-scale grammar development, such as the ParGram project for LFG (**??**) and the LinGO/Grammar Matrix and CoreGram projects for HPSG (**????**).[19] This naturally led to an interest in underspecified semantic representations, so

---

[19]Readers can explore the current incarnations of these projects at the following links (checked May 9, 2020):

ParGram `https://pargram.w.uib.no`

LinGO `http://lingo.stanford.edu`

Grammar Matrix `http://matrix.ling.washington.edu/index.html`

CoreGram `https://hpsg.hu-berlin.de/Projects/CoreGram.html`

that semantic ambiguities such as scope ambiguity could be compactly encoded without the need for full enumeration of all scope possibilities. Two examples for HPSG are *Lexical Resource Semantics* (**??**) and *Minimal Recursion Semantics* (**?**). Similarly, focus in semantics for LFG shifted to ways of encoding semantic ambiguity compactly and efficiently. This led to the development of Glue Semantics.

12.2 General Glue Semantics

In this section, we briefly review Glue Semantics itself, without reference to a particular syntactic framework. Glue Semantics is a general framework for semantic composition that requires *some* independent syntactic framework but does not presuppose anything about syntax except headedness, which is an uncontroversial assumption across frameworks. This makes the system flexible and adaptable, and it has been paired not just with LFG, but also with Lexicalized Tree-Adjoining Grammar (**?**), HPSG (**?**), Minimalism (**?**), and Universal Dependencies (**?**).

In Glue Semantics, meaningful linguistic expressions — including lexical items but possibly also particular syntactic configurations — are associated with *meaning constructors* of the following form:[20]

(34) $\mathcal{M} : G$

$\mathcal{M}$ is an expression from a *meaning language* which can be anything that supports the lambda calculus; G is an expression of *linear logic* (**?**), which specifies the semantic composition (it "glues meanings together"), based on a syntactic parse. By convention a colon separates them. Glue Semantics is related to (Type-Logical) Categorial Grammar (**????**), but it assumes a separate syntactic representation for handling word order, so the terms of the linear logic specify just semantic composition without regard to word order (see **?** for further discussion). Glue Semantics is therefore useful in helping us focus on semantic composition in its own right.

The principal compositional rules for Glue Semantics are those for the linear implication connective, $\multimap$, which are here presented in a natural deduction format, in which each connective has an elimination rule ($\multimap_{\mathcal{E}}$, in this case) and an introduction rule ($\multimap_{\mathcal{I}}$, in this case).:

(35) Functional application : Implication elimination (modus ponens)

$$\frac{f : A \multimap B \quad a : A}{f(a) : B} \; \multimap_{\mathcal{E}}$$

(36) Functional abstraction : Implication introduction (hypothetical reasoning)

---

[20]It is in principle possible for a linguistic expression to have a phonology and syntax but not contribute to interpretation, such as expletive pronouns like *there* and *it* or the *do*-support auxiliary in English; see **?**: 113 for some discussion of expletive pronouns in the context of Glue.

$$\frac{\begin{array}{c} [a : A]^1 \\ \vdots \\ f : B \end{array}}{\lambda a.f : A \multimap B} \; {\multimap}_{\mathcal{I},1}$$

Focusing first on the right-hand, linear logic side, the implication elimination rule is just standard modus ponens. The implication introduction rule is hypothetical reasoning. A hypothesis is made in the first line as an assumption, indicated by presenting it in square brackets with an index that flags the particular hypothesis/assumption. Given this hypothesis, if through some series of proof steps, indicated by the vertical ellipsis, we derive a term, then we are entitled to withdraw the assumption, using its flag to indicate that it is this particular assumption that has been withdrawn, and conclude that the hypothesis implies the term so-derived. In each of these rules, the inference over the linear logic term corresponds to an operation on the meaning term, via the Curry-Howard Isomorphism between formulas and types (**???**). The rule for eliminating the linear implication corresponds to functional application. The rule for introducing the linear implication corresponds to functional abstraction. These rules will be seen in action shortly.

In general, given some head h and some arguments of the head $a_1, \ldots, a_n$, an implicational term like the following models consumption of the arguments to yield the saturated meaning of the head: $a_1 \multimap \ldots \multimap a_n \multimap h$. For example, let us assume the following meaning constructor for the verb *likes* in the sentence *Max likes Sam*:

(37) $\lambda y \lambda x.\mathbf{like}(y)(x) : s \multimap m \multimap l$

Let's also assume that s is mnemonic for the semantic correspondent of the (single word) phrase *Sam*, m similarly mnemonic for *Max*, and l for *likes*. In other words, the meaning constructor for *likes* would be associated with the lexical entry for the verb and specified in some general form such that it can be instantiated by the syntax (we will see an LFG example shortly); here we are assuming that the instantiation has given us the meaning constructor in (37).

Given this separate level of syntax, the glue logic does not have to worry about word order and is permitted to be commutative (unlike the logic of Categorial Grammar). We could therefore freely reorder the arguments for *likes* above such that we instead first compose with the subject and then the object, but still yield the meaning appropriate for the intended sentence *Max likes Sam* (rather than for *Sam likes Max*):

(38) $\lambda x \lambda y.\mathbf{like}(y)(x) : m \multimap s \multimap l$

As we will see below, the commutativity of the glue logic yields a simple and elegant treatment of quantifiers in non-subject positions, which are challenging for other frameworks (see, for example, the careful pedagogical presentation of the issue in **?**: 244–263).

First, though, let us see how this argument reordering, otherwise known as Currying or Schönfinkelization, works in a proof, which also demonstrates the rules of implication elimination and introduction:

(39)

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\lambda y.\lambda x.f(y)(x) : a \multimap b \multimap c \quad [v:a]^1}{\lambda x.f(v)(x) : b \multimap c}\multimap_{\mathcal{E}} \quad [u:b]^2}{f(v)(u) : c}\multimap_{\mathcal{E}}}{\lambda v.f(v)(u) : a \multimap c}\multimap_{\mathcal{I},1}}{\lambda y.f(y)(u) : a \multimap c}\Rightarrow_\alpha}{\lambda u.\lambda y.f(y)(u) : b \multimap a \multimap c}\multimap_{\mathcal{I},2}}{\lambda x.\lambda y.f(y)(x) : b \multimap a \multimap c}\Rightarrow_\alpha$$

The general structure of the proof is as follows. First, an assumption (hypothesis) is formed for each argument, in the order in which they originally occur, corresponding to a variable in the meaning language. Each assumed argument is then allowed to combine with the implicational term by implication elimination. Once the implicational term has been entirely reduced, the assumptions are then discharged in the same order that they were made, through iterations of implication introduction. The result is the original term in curried form, such that the order of arguments has been reversed but without any change in meaning. The two steps of α-equivalence, notated $\Rightarrow_\alpha$, are of course not strictly necessary, but have been added for exposition.

This presentation has been purposefully abstract to highlight what is intrinsic to the glue logic, but we of course need to see how this works with a syntactic framework to see how Glue Semantics actually handles semantic composition and the syntax–semantics interface. So next, in Section 12.3, we will review LFG+Glue, as this is the predominant pairing of syntactic framework and Glue.

12.3 Glue Semantics for LFG

Glue for LFG will be demonstrated by analyses of the following three examples:

(40)   Blake called Alex.

(41)   Blake called everybody.

(42)   Everybody called somebody.

Example (40) is a simple case of a transitive verb with two proper name arguments, but is sufficient to demonstrate the basics of the syntax–semantics interface in LFG+Glue. Example (41) is a case of a quantifier in object position, which is challenging to compositionality because there is a type clash between the simplest type we can assign to the verb, $\langle e, \langle e, t \rangle \rangle$, and the simplest type that would be assigned to the quantifier, $\langle \langle e, t \rangle, t \rangle$. In other theories, this necessitates either a syntactic operation which is syntactically undermotivated, e.g. Quantifier Raising in interpretive theories of composition, such as Logical Form semantics (**?**), or a type shifting operation of some kind in directly compositional approaches, as in categorial or type-logical frameworks; see **?** for further discussion and references. Example (42) also demonstrates this point, but it more importantly demonstrates that quantifier scope ambiguity can be handled in Glue without a)

Expression	Type	Meaning Constructor
Alex	e	$\mathbf{alex} : \uparrow_\sigma$
Blake	e	$\mathbf{blake} : \uparrow_\sigma$
called	$\langle e, \langle e, t \rangle \rangle$	$\lambda y.\lambda x.\mathbf{call}(y)(x) : (\uparrow \text{OBJ})_\sigma \multimap (\uparrow \text{SUBJ})_\sigma \multimap \uparrow_\sigma$
everybody	$\langle \langle e, t \rangle, t \rangle$	$\lambda Q.\mathbf{every}(\mathbf{person}, Q) : \forall S.(\uparrow_\sigma \multimap S) \multimap S$
somebody	$\langle \langle e, t \rangle, t \rangle$	$\lambda Q.\mathbf{some}(\mathbf{person}, Q) : \forall S.(\uparrow_\sigma \multimap S) \multimap S$

Table 1: Relevant lexical details for the three examples in (40–42)

positing an undermotivated syntactic ambiguity and b) while maintaining the simplest types for both quantifiers.

The relevant aspects of the lexical entries involved are shown in Table 1. Other syntactic aspects of the lexical items, such as the fact that *called* has a SUBJECT and an OBJECT, are specified in its meaning constructor. Minimal f-structures are provided below for each example. The subscript σ indicates the semantic structure that corresponds to the annotated f-description. The types for the lexical items are the minimal types that would be expected. Note that in Glue these are normally associated directly with the semantic structures, for example $\uparrow_{\sigma_e}$ and $(\uparrow \text{OBJ})_{\sigma_e} \multimap (\uparrow \text{SUBJ})_{\sigma_e} \multimap \uparrow_{\sigma_t}$, but they have been presented separately for better exposition; see **?**: 299–305 for further discussion. We do not show semantic structures here, as they are not necessary for this simple demonstration.

The generalized quantifier functions associated with *everybody* and *somebody* are, respectively, **every** and **some** in the meaning language. The universal symbol $\forall$ in the glue logic/linear logic terms for the quantifiers ranges over semantic structures of type t. It is unrelated to the generalized quantifiers. Hence even the existential word *somebody* has the universal $\forall$ in its linear logic glue term. The $\forall$-terms thus effectively say that *any* type t semantic structure S that can be found by application of proof rules such that the quantifier's semantic structure implies S can serve as the scope of the quantifier; see **?**: 393–394 for basic discussion of the interpretation of $\forall$ in linear logic. This will become clearer when quantifier scope is demonstrated shortly.

Let us assume the following f-structure for (40):

$$(43) \quad c \begin{bmatrix} \text{PRED} & \text{`call'} \\ \text{SUBJ} & b[\text{PRED} \ \text{`Blake'}] \\ \text{OBJ} & a[\text{PRED} \ \text{`Alex'}] \end{bmatrix}$$

Note that here, unlike in previous sections, the PRED value for the verb does not list its subcategorization information. This is because we've made the standard move in much Glue work to suppress this information.[21] The f-structures are named mnemonically by the first character of their PRED value. All other f-structural information has been

---

[21] Indeed, one could go further and argue that PRED values do not list subcategorization at all, in which case the move is not just notational, and that the Principles of Completeness and Coherence instead follow from the resource-sensitivity of Glue Semantics; for some discussion, see **???**.

suppressed for simplicity. Based on these f-structure labels, the meaning constructors in the lexicon in Table 1 are instantiated as follows (σ subscripts suppressed):

(44) **Instantiated meaning constructors**
   blake : b
   alex : a
   $\lambda y.\lambda x.\mathbf{call}(y)(x) : a \multimap b \multimap c$

These meaning constructors yield the following proof, which is the only available normal form proof for the sentence:[22]

(45) **Proof**

$$\dfrac{\dfrac{\dfrac{\lambda y.\lambda x.\mathbf{call}(y)(x) : a \multimap b \multimap c \quad \mathbf{alex} : a}{(\lambda y.\lambda x.\mathbf{call}(y)(x))(\mathbf{alex}) : b \multimap c} \multimap_{\mathcal{E}}}{\lambda x.\mathbf{call}(\mathbf{alex})(x) : b \multimap c} \Rightarrow_{\beta} \qquad \mathbf{blake} : b}{\dfrac{(\lambda x.\mathbf{call}(\mathbf{alex})(x))(\mathbf{blake}) : c}{\mathbf{call}(\mathbf{alex})(\mathbf{blake}) : c} \Rightarrow_{\beta}} \multimap_{\mathcal{E}}$$

The final meaning language expression, **call**(**alex**)(**blake**), gives the correct truth conditions for *Blake called Alex*, based on a standard model theory.

Let us next assume the following f-structure for (41):

(46) $c\begin{bmatrix} \text{PRED} & \text{`call'} \\ \text{SUBJ} & b[\text{PRED} \ \text{`Blake'}] \\ \text{OBJ} & e[\text{PRED} \ \text{`everybody'}] \end{bmatrix}$

Based on these f-structure labels, the meaning constructors in the lexicon are instantiated as follows (σ subscripts again suppressed):

(47) **Instantiated meaning constructors**
   $\lambda y.\lambda x.\mathbf{call}(y)(x) : e \multimap b \multimap c$
   $\lambda Q.\mathbf{every}(\mathbf{person}, Q) : \forall S.(e \multimap S) \multimap S$
   blake : b

These meaning constructors yield the following proof, which is again the only available normal form proof:[23]

(48) **Proof**

$$\dfrac{\dfrac{\lambda Q.\mathbf{every}(\mathbf{person}, Q) : \forall S.(e \multimap S) \multimap S}{\lambda Q.\mathbf{every}(\mathbf{person}, Q) : (e \multimap c) \multimap c} \forall_{\mathcal{E}}[c/S] \qquad \dfrac{\dfrac{\dfrac{\lambda y.\lambda x.\mathbf{call}(y)(x) : e \multimap b \multimap c \quad [z : e]^1}{\lambda x.\mathbf{call}(z)(x) : b \multimap c} \multimap_{\mathcal{E}}, \Rightarrow_{\beta} \quad \mathbf{blake} : b}{\dfrac{\mathbf{call}(z)(\mathbf{blake}) : c}{\lambda z.\mathbf{call}(z)(\mathbf{blake}) : e \multimap c} \multimap_{\mathcal{I},1}} \multimap_{\mathcal{E}}, \Rightarrow_{\beta}}{\mathbf{every}(\mathbf{person}, \lambda z.\mathbf{call}(z)(\mathbf{blake})) : c} \multimap_{\mathcal{E}}, \Rightarrow_{\beta}$$

---

[22] The reader can think of the normal form proof as the minimal proof that yields the conclusion, without unnecessary steps of introducing and discharging assumptions; see **?** for some basic discussion.

[23] We have not presented the proof rule for Universal Instantiation, but it is trivial; see **?**: 396.

The final meaning language expression, **every**(**person**, $\lambda z.$**call**(z)(**blake**)), again gives the correct truth conditions for *Blake called everybody*, based on a standard model theory with generalized quantifiers.

Notice that the quantifier need not be moved in the syntax — it's just an OBJECT in f-structure — and no special type shifting was necessary. This is because the proof logic allows us to temporarily fill the position of the object quantifier with a hypothetical meaning constructor that consists of a type e variable paired with the linear logic term for the object; this assumption is then discharged to return the scope of the quantifier, $e \multimap c$, and the corresponding variable bound, to yield the function that maps individuals called by Blake to a truth value. In other words, we have demonstrated that this approach scopes the quantifier without positing an *ad hoc* syntactic operation and without complicating the type of the object quantifier or the transitive verb. This is ultimately due to the commutativity of the glue logic, linear logic, since the proof does not have to deal with the elements of composition (words) in their syntactic order, because the syntax is separately represented by c-structure (not shown here) and f-structure.

Lastly, let us assume the following f-structure for (42):

(49) $\quad c \begin{bmatrix} \text{PRED} & \text{'call'} \\ \text{SUBJ} & e\begin{bmatrix} \text{PRED} & \text{'everybody'} \end{bmatrix} \\ \text{OBJ} & s\begin{bmatrix} \text{PRED} & \text{'somebody'} \end{bmatrix} \end{bmatrix}$

Based on these f-structure labels, the meaning constructors in the lexicon are instantiated as follows:

(50) **Instantiated meaning constructors**
$\lambda y.\lambda x.$**call**$(y)(x) : s \multimap e \multimap c$
$\lambda Q.$**some**(**person**, Q) : $\forall S.(s \multimap S) \multimap S$
$\lambda Q.$**every**(**person**, Q) : $\forall S.(e \multimap S) \multimap S$

These meaning constructors yield the following proofs, which are the only available normal form proofs, but there are two distinct proofs, because of the scope ambiguity:[24]

(51) **Proof 1 (subject wide scope)**

$$\cfrac{\lambda Q.\textbf{every}(\textbf{person},Q): \atop \forall S.(e \multimap S) \multimap S \quad \cfrac{\lambda Q.\textbf{some}(\textbf{person},Q): \atop \forall S.(s \multimap S) \multimap S \quad \cfrac{\cfrac{\cfrac{\lambda y.\lambda x.\textbf{call}(y)(x): \atop s \multimap e \multimap c \quad [v:s]^1}{\lambda x.\textbf{call}(v)(x): e \multimap c} {\scriptstyle \multimap_{\mathcal{E}}, \Rightarrow_\beta} \quad [u:e]^2}{\cfrac{\textbf{call}(v)(u):c}{\lambda v.\textbf{call}(v)(u): s \multimap c} {\scriptstyle \multimap_{\mathcal{I},1}}} {\scriptstyle \multimap_{\mathcal{E}}, \Rightarrow_\beta}}{\cfrac{\textbf{some}(\textbf{person},\lambda v.\textbf{call}(v)(u)):c}{\lambda u.\textbf{some}(\textbf{person},\lambda v.\textbf{call}(v)(u)): e \multimap c} {\scriptstyle \multimap_{\mathcal{I},2}}} {\scriptstyle \multimap_{\mathcal{E}}, \forall_{\mathcal{E}}[c/S], \Rightarrow_\beta}}{\cfrac{\cfrac{\cfrac{\textbf{every}(\textbf{person},\lambda u.\textbf{some}(\textbf{person},\lambda v.\textbf{call}(v)(u)))}{\textbf{every}(\textbf{person},\lambda u.\textbf{some}(\textbf{person},\lambda y.\textbf{call}(y)(u)))} {\scriptstyle \Rightarrow_\alpha}}{\textbf{every}(\textbf{person},\lambda x.\textbf{some}(\textbf{person},\lambda y.\textbf{call}(y)(x)))} {\scriptstyle \Rightarrow_\alpha}}{} } {\scriptstyle \multimap_{\mathcal{E}}, \forall_{\mathcal{E}}[c/S], \Rightarrow_\beta}$$

---

[24]We have made the typical move in Glue work of not showing the trivial universal instantiation step this time.

(52) **Proof 2 (object wide scope)**

$$
\cfrac{
\lambda Q.\mathbf{some}(\mathbf{person}, Q):\ \forall S.(s \multimap S) \multimap S
\qquad
\cfrac{
\lambda Q.\mathbf{every}(\mathbf{person}, Q):\ \forall S.(e \multimap S) \multimap S
\qquad
\cfrac{
\cfrac{
\lambda y.\lambda x.\mathbf{call}(y)(x):\ s \multimap e \multimap c \qquad [v:s]^1
}{
\lambda x.\mathbf{call}(v)(x):\ e \multimap c
}\ {\scriptstyle \multimap_{\mathcal{E}},\Rightarrow_\beta}
}{
\cfrac{
\mathbf{every}(\mathbf{person}, \lambda x.\mathbf{call}(v)(x)):\ c
}{
\lambda v.\mathbf{every}(\mathbf{person}, \lambda x.\mathbf{call}(v)(x)):\ s \multimap c
}\ {\scriptstyle \multimap_{\mathcal{I},1}}
}\ {\scriptstyle \multimap_{\mathcal{E}},\forall_{\mathcal{E}}[c/S],\Rightarrow_\beta}
}{
\cfrac{
\mathbf{some}(\mathbf{person}, \lambda v.\mathbf{every}(\mathbf{person}, \lambda x.\mathbf{call}(v)(x)))
}{
\mathbf{some}(\mathbf{person}, \lambda y.\mathbf{every}(\mathbf{person}, \lambda x.\mathbf{call}(y)(x)))
}\ {\scriptstyle \Rightarrow_\alpha}
}\ {\scriptstyle \multimap_{\mathcal{E}},\forall_{\mathcal{E}}[c/S],\Rightarrow_\beta}
$$

The final meaning language expressions in (51) and (52) give the two possible readings for the scope ambiguity, again assuming a standard model theory with generalized quantifiers. Once more, notice that neither quantifier need be moved in the syntax — they are respectively just a SUBJECT and an OBJECT in f-structure. And once more, no special type shifting is necessary. It is a key strength of this approach that even quantifier scope ambiguity can be captured without positing *ad hoc* syntactic operations (and, again, without complicating the type of the object quantifier or the transitive verb). This is once more ultimately due to the commutativity of the linear logic.

13 Conclusion

HPSG and LFG are rather similar syntactic frameworks, both of them important lexicalist alternatives to transformational grammar. They allow for the expression of roughly the same set of substantive analyses, where analyses are individuated in terms of deeper theoretical content rather than superficial properties. The same sort of analytical options can be compared under both systems, answers to questions such as whether a phenomenon is to be captured on a lexical level or in the syntax, whether a given word string is a constituent or not, the proper treatment of complex predicates, and so on. Analyses in one framework can often be translated to the other, preserving the underlying intuition of the account.

Against the backdrop of a general expressive similarity, we have pointed out a few specific places where one framework makes certain modes of analysis available that are not found in the other. The main thesis of this chapter is that the differences between the frameworks stem from different design motivations, reflecting subtly different methodological outlooks. HPSG is rooted in context-free grammars and an interest in the study of locality. LFG is based on the notion of functional similarity or equivalence between what are externally rather different structures (for example, fixed phrasal positions, case markers, and agreement inflections can all function similarly to signal grammatical relations). LFG makes that functional similarity highly explicit, reflecting its functionalist-friendly outlook on grammar.