Chapter 41

LFG and HPSG

Adam Przepiórkowski
University of Warsaw; Polish Academy of Sciences; University of Oxford

This chapter presents and compares Lexical Functional Grammar and Head-driven Phrase Structure Grammar. It concentrates on their fundamental properties rather than on analyses of particular phenomena. After discussing representations assumed in each theory and the kinds of grammars that lead to such representations, the chapter devotes some attention to models explicitly or implicitly assumed in HPSG and LFG: it identifies some problems and suggests possible solutions.

1 Introduction

The aim of this chapter is to juxtapose two highly formalised grammatical theories: Lexical Functional Grammar (LFG; Kaplan & Bresnan 1982; Bresnan et al. 2016; Dalrymple et al. 2019) and Head-driven Phrase Structure Grammar (HPSG; Pollard & Sag 1987, 1994; Müller et al. 2021).\(^1\) LFG was conceived in the late 1970s, HPSG – in the mid-1980s, so both theories have been around for decades. Within both theories, diverse phenomena have been analysed and then re-analysed, and many will undoubtedly receive new analyses in the future. For this reason, rather than compare particular analyses of some phenomena, this chapter focuses on more fundamental issues: on the general representational architecture of the two theories (in Section 2), on the kinds of grammars that lead to these representations (in Section 3), and on models assumed in both theories (in Section 4). Wechsler & Asudeh (2021) offers a comparison of the treatment of various phenomena in the two theories and, hence, complements the current chapter.

\(^1\)This chapter does not presuppose substantial prior exposure to either LFG or HPSG.
2 Representations

Outside of their respective communities, both theories are best known as theories of syntax, although already at their conception both were envisaged as theories of multiple linguistic levels, including semantics. Current versions of both theories have well-developed approaches to semantics, as well as proposals for the representation of other linguistic levels: morphological and information-structural in the case of both theories, phonological in the case of HPSG, and prosodic in the case of LFG.

However, the two theories adopt rather different approaches to the representation of the various linguistic levels.

2.1 LFG

Let us have a look at possible representations of the simple sentence (1) in the two theories, starting with the LFG representation in Figure 1.

(1) She loves you.

![Figure 1: LFG representation of (1)](image)

A prominent feature of LFG representations are the multiple levels. Figure 1 features three such levels: constituent structure (c-structure; the tree on the left), functional structure (f-structure; the attribute–value matrix, AVM for short, in the middle), and semantic structure (s-structure; the AVM on the right). The first two of these levels are syntactic in nature and they are the core of any LFG analysis. The repertoire and exact properties of other levels, including s-structure, is
a matter of some debate. Among other prominent levels widely assumed in LFG are prosodic structure (for overviews see Dalrymple et al. 2019: Chapter 11 and Bögel forthcoming [this volume]) and information structure (see Dalrymple et al. 2019: Chapter 10 and Zaenen forthcoming [this volume]). Also argument structure used to be assumed as a separate level (see, e.g., Butt et al. 1997), but given an appropriately spelled-out approach to semantics, a separate a-structure does not seem to be needed (see, e.g., Asudeh & Giorgolo 2012 and Findlay 2016).

As shown in Figure 1, levels of representation are connected via mapping functions (rendered in the figure with arrows between levels). One such function, usually called $\phi$, maps c-structures to f-structures, another, $\sigma$, maps f-structures to s-structures. These functions are not necessarily total. In particular, it is often assumed that $\phi$ maps to f-structures only non-terminal nodes of c-structures. For example, in Figure 1, the leftmost nodes NP and Pron, but not the terminal node *she* that they dominate, map to the f-structure representing the subject (the value of the *subj* attribute), the rightmost nodes NP and Pron, but not the terminal *you*, map to the f-structure representing the object, etc. Similarly, the domain of $\sigma$ consists of three f-structures (the ones containing the *pred* attribute), with the exclusion of the f-structures which are the values of *index*. These functions are also not surjective (not onto), for example the values of *index* in the f-structure are not in the range of $\phi$.

Let us take a brief look at particular levels. The c-structure in Figure 1 should be self-explanatory. Unlike derivational theories (see Sells forthcoming [this volume]), but like Simpler Syntax (see Varaschin forthcoming [this volume]) and HPSG, LFG assumes very simple constituency trees, usually without empty categories – but see Bresnan et al. (2016: Chapter 9) for exceptions – and without an abundance of functional nodes. Constituency structures are assumed to vary considerably between languages, even though their grammars are required to follow some – appropriately relaxed – version of the $X'$-theory.  

On the other hand, functional structures are cross-linguistically more uniform. While they contain morphosyntactic information, which is quite different for different languages, their main function is to represent grammatical functions such as subject and object, and the repertoire of grammatical functions is supposed to be universal.  

---

2See, e.g., Bresnan et al. 2016: Chapter 6 and Dalrymple et al. 2019: Section 3.2. LFG versions of $X'$-theory are relaxed in various ways. While the standard $X'$-theory assumes at most binary branching, LFG does not make such an assumption. Also, standard derivational versions of $X'$-theory assume the presence of the head (perhaps subsequently moved to a different tree position or realised as a phonetically empty constituent to start with), while in LFG the head may be optional in a rule and completely absent from the resulting tree. In this sense, LFG versions of $X'$-theory may be construed as theories of descriptions rather than structures.

3See Patejuk & Przepiórkowski (2016) for a critical discussion of this assumption and Kaplan (2017) for a reply.
the \texttt{PRED} attribute – originally designed to encode in syntax the information that maps to semantic representations; as repeatedly noted in the literature, this information is largely redundant in contemporary LFG, given the existence of semantic structures.\footnote{See, e.g., Dalrymple et al. (1993: 13–14) and Kuhn (2001: Sections 1.3.3, 1.4.1).} In the case of Figure 1, the main f-structure represents a present-tense utterance with the semantic form ‘\texttt{LOVE\langle SUBJ, OBJ\rangle}’. Both the subject and the object of this utterance have the semantic form ‘\texttt{PRO\rangle’, i.e., they are pronouns, specifically, personal pronouns. Their morphosyntactic information is represented within the values of \texttt{CASE} and \texttt{INDEX}.\footnote{Analyses in both LFG and HPSG often follow Wechsler & Zlatić (2003) and distinguish between \texttt{INDEX} agreement and \texttt{CONCORD} agreement; here I retain \texttt{INDEX} as a separate bundle of features but do not explicitly represent the \texttt{CONCORD} bundle, just the \texttt{CASE} feature within it.}

Finally, s-structures contain purely semantic information. In the case at hand, it is the information that the meaning of this utterance is modelled by the relation \texttt{LOVE} and that there are two arguments of this relation, corresponding to the subject and the object.

\section{2.2 HPSG}

Let us now have a look at the HPSG representation of (1) in Figure 2. HPSG representations are formally more uniform: there is just one contiguous data structure used for the representation of information from all linguistic levels, namely, an attribute–value matrix.\footnote{Figure 2 also contains lists, indicated with angle brackets, but this is a shorthand notation for AVMs with attributes such as \texttt{FIRST} and \texttt{REST} (or \texttt{HEAD} and \texttt{TAIL}), whose values are the first element (head) of the list and the rest (tail) of the list.} In particular, there are no separate levels of representation – all constituency, morphosyntactic, and semantic information is interspersed throughout the structure.

Clearly, the cost of the greater formal uniformity is the diminished perspicuity (or, for an unaccustomed eye, downright unreadability) of representations such as that in Figure 2. For this reason, it is common among HPSG practitioners to use all kinds of abbreviations and representational devices to make representations more readable. For example, the structure of that figure may be presented as in Figure 3, where the constituency structure becomes transparent.

Taking a closer look at the AVM in Figure 2 we may first note that, unlike f-structures (or s-structures) in LFG, feature structures in HPSG are typed. For example, the structure represented by the whole AVM is of type \texttt{hd-subj-ph} (i.e., \textit{head-subject-phrase}), and the value of the attribute \texttt{HD-DTR} is of type \texttt{hd-comp-ph} (i.e., \textit{head-complement-phrase}).
Figure 2: HPSG representation of (1)
Figure 3: Shorthand HPSG representation of (1)
As discussed in more detail in Section 3.2 below, types determine what attributes may and must appear on the objects described by the AVM (not necessarily on the AVM itself, which may be a partial description of such objects; this point will be crucial below) and what their values may and must be. Types are ordered in an inheritance hierarchy, where subtypes inherit conditions imposed by supertypes and may add more such conditions. For example, both `hd-subj-ph` and `hd-comp-ph` are subtypes of `headed-phrase`, which is a subtype of `phrase`, which in turn – along with `word` – is a subtype of `sign`; see Figure 4.

```
[ word  list  
   PHON    synsem ]

word [ phrase  
      DTRS list ]

[ headed-phrase  non-headed-phrase  
   HD-DTR sign ]

hd-subj-ph  hd-comps-ph
```

Figure 4: A small fragment of an HPSG type hierarchy

All objects of type `sign` must have two attributes: `PHON` and `SYNSEM` (I will explain their role shortly). The `word` subtype does not add any conditions, and all the three subsidiary AVMs of type `word` in Figure 2 have exactly these two attributes and no others. On the other hand, the `phrase` subtype of `sign` requires an additional attribute, namely, `DTRS` (i.e., `DAUGHTERS`), whose value is a list of immediate constituents. An important subtype of `phrase` is `headed-phrase`, where one of the immediate constituents is singled out as the syntactic head; this constituent

---

7In the HPSG lingo, this amounts to saying that feature structures are totally well-typed (Carpenter 1992: 94–95; Pollard & Sag 1994: 18).

8It is sometimes argued that LFG templates (which are, essentially, possibly parameterised macros, as in programming languages) “can play the same role in capturing linguistic generalizations as hierarchical type systems in theories like HPSG” (Dalrymple et al. 2004: 207); unfortunately, a discussion of similarities and differences between the two mechanisms – especially, the crucial ontological differences – is outside the scope of this fairly introductory chapter.
is the value of the additional HD-DTR (i.e., HEAD-DAUGHTER) attribute. Hence, any object of type headed-phrase must have four attributes: PHON, SYNSEM, DTRS, and HD-DTR. As hd-subj-ph and hd-comp-ph do not add any attributes, the two AVMs corresponding to the phrases *she loves you* and *loves you* have exactly these four attributes.

Let us take a closer look at the encoding of constituency structure via the attributes DTRS and HD-DTR. In the root AVM of Figure 2, the value of DTRS is a 2-element list, whose first element is a *word* structure of *she* and the second element is a *hd-comp-ph* structure of *loves you*. This second element is only marked as [1] on the DTRS list, but it is fully presented as the value of the HD-DTR attribute; boxed numbers such as [1] should be understood as bound variables signalling multiple occurrences of a structure in different places (here, in the DTRS list and in the value of HD-DTR). The structure [1], being (a subtype of) a headed phrase, also has the attribute DTRS, whose value is a pair of structures of *loves* and *you*, and the HD-DTR attribute, which singles out the structure of *loves* as the head. This configuration of attributes DTRS and HD-DTR and their values encodes the syntactic tree of Figure 3.

The other two attributes of phrase structures, present also on word structures, are PHON and SYNSEM. In work which does not deal with phonology or phonetics the values of PHON are taken to be lists of words, as in Figure 2, but it is clear that in an exhaustive representation values of PHON must be highly structured.\(^9\)

For our purposes, values of SYNSEM are more important – they represent all grammatical information other than constituency structure. Figure 2 presents slightly simplified values of SYNSEM: it omits those parts of synsem structures which are responsible for non-local information, i.e., for book keeping related to unbounded dependencies and relative clauses (see Borsley & Crysmann 2021, Arnold & Godard 2021, Chaves 2021, and references therein).\(^10\) Local information is distributed between the attributes CAT(egory) and CONT(ent), as well as CONTEXT, not represented here either (see Pollard & Sag 1994: 332–337, as well as De Kuthy 2021 and references therein). CONT represents semantic information comparable to that distributed between LFG f-structures and s-structures. For example, the two personal pronouns (see the two CONT values of type ppro) contribute referential indices, referred to as [3] and [5], and the verb contributes the love-rel(ation) with the index [3] of *she* as its ACT(or) and the index [5] of *you* as...

---

\(^9\)See, e.g., Bird & Klein (1994) and Höhle (1999) for two very different proposals.

\(^10\)Normally, synsem structures contain two attributes, local and nonlocal. Since nonlocal and its value is omitted here, also the attribute local is not mentioned in this chapter, and its values of type local are presented as SYNSEM values of type synsem.
its und(ergoer). This verbal semantics is shared along the verbal spine, so the structures of loves, loves you, and she loves you all have the same cont value [7].

The other part of synsem values, the category structure, models morphosyntactic and combinatorial properties. The former are the value of head: she is a nominative (pro)noun, you is (here) an accusative (pro)noun, and loves is a finite verb (non-auxiliary, not inverted). The values of head are shared between a mother and its head daughter – see the multiple occurrences of [10]. Finally, combinatorial properties are encoded in values of val(ence): the verb loves requires a subject (8 – the synsem value of she) and a complement (9 – the synsem value of you), loves you has no further complement expectations but still needs a subject, while she loves you is a fully saturated maximal projection – the values of its valency features are empty lists. Such maximal projections are often abbreviated the way illustrated in Figure 3: NP[nom]3 stands for (the synsem value of) a structure with empty subj and comps, with head indicating a nominative noun, and with cont|index value [3].

2.3 Comparison

2.3.1 Levels of representation

The two structures in Figures 1 and 2 look somewhat similar in the sense that they both use complex AVMs, but also very different in the sense that the LFG representation distinguishes multiple levels, each with its own data structure and with a functional mapping between the levels, while the HPSG representation is a monolithic AVM. How important is this difference? My claim is that it is less important than usually assumed. For example, it is possible to define a bijection (a one-to-one correspondence) between LFG representations such as that in Figure 1 and corresponding HPSG-like monolithic AVM representations such as that in Figure 5. In this representation, the c-structure is encoded with the help of attributes label, dtrs, and phon, the mapping from the c-structure to the f-structure is achieved with the attribute synsem, and the mapping from the f-structure to the s-structure – with the attribute cont.12

Conversely, the HPSG representation of Figure 2 might be taken apart and LFGified as in Figure 6. The fact that non-terminal nodes in the c-structure are AVMs is not a problem in itself; in LFG it is often assumed that c-structure node labels

---

11While in LFG the attribute separator in paths is a space, e.g., “subj case”, in HPSG the vertical bar is used, e.g., “synsem|cat|head|case”.
12In fact, this representation makes conspicuous the redundancy – mentioned in Section 2.1 – of pred values with respect to s-structures (i.e., here, cont values).
are really abbreviations of feature matrices (see, e.g., Kaplan 1995, Dalrymple 2017, and Lowe & Lovestrand 2020). What is somewhat unusual is that some of the attributes in these AVMs are list-valued and refer to other AVMs within the same c-structure (rather than to particular values within such AVMs as in, e.g., Lowe & Lovestrand 2020). However, this does not seem to violate any deep LFG principles.

What LFG grammars and the multi-level representations they lead to try to capture is the cognitive modularity and encapsulation of particular linguistic levels; constituency structures, functional structures, semantic structures, etc., each have their own sets of primitives and operations, and the interactions between them are only possible via the mapping functions \( \phi \), \( \sigma \), etc. By contrast, no such encapsulation is attempted in HPSG, so it is easy to state constraints in this theory which may simultaneously refer to arbitrary parts of the structure of a sentence, e.g., the phonetic properties of a verb and the semantics of its subject; such a constraint would be much more cumbersome to state in LFG. On the other hand, actual LFG analyses sometimes make use of the converses of \( \phi \),
Figure 6: LFG-like HPSG representation of (1)

σ, etc., i.e., refer to c-structures from the level of f-structures and to f-structures from the level of s-structures, so, in principle, any level may be referred to from any other level. Hence, the difference between LFG and HPSG when it comes to encapsulation of linguistic levels is one of degree – and relative easiness of stating constraints across grammatical levels – rather than a categorical difference between the complete encapsulation and the total lack thereof.

In summary, while representations with separate linguistic levels such as those in Figures 1 and 6 are certainly more immediately readable than monolithic representations such as those in Figures 2 and 5, it is not clear that there are any fundamental differences in the kinds of linguistic analyses that LFG and HPSG presuppose.

---

13 However, as pointed out by Ash Asudeh (p.c.), correspondence functions in LFG are typically not injective (i.e., they are many-to-one), so their converses are proper relations rather than functions. For example, while φ maps particular c-structure nodes to particular f-structures, the converse of φ will map f-structures to sets of c-structure nodes, making it more difficult to refer to particular c-structure nodes from the level of f-structures. This “blurring” or “fuzziness” of converses of correspondence functions might be claimed to constitute a substantive hypothesis about encapsulation of grammatical levels.

14 But see Section 3.3.3, on the expressiveness of formalisms underlying LFG and HPSG.
2.3.2 Grammatical functions

Perhaps a more important – and certainly linguistically more contentful – difference between HPSG and LFG regards grammatical functions. In LFG each argument bears a different grammatical function drawn from a repertoire that includes subj(ect) and obj(ect), as in Figure 1, but also obl(ique), comp(lement) – a closed sentential argument, xcomp – an open verbal argument, etc. Moreover, at least obj and obl are often indexed with thematic roles, grammatical cases, or particular prepositions. For example, in the case of sentence (2), the f-structure would contain another attribute apart from subj(for you) and obj(for me), namely, obj_theme (for your money). Similarly, in the case of (3), the grammatical function of to you could be obl_goal, etc. (see, e.g., Dalrymple et al. 2019: Section 10.3 and references therein).

(2) You never give me your money.

(3) But what I’ve got I’ll give to you.

The HPSG approach to naming arguments is radically different: normally only the subj(ect) is distinguished (see Pollard & Sag 1994: Chapter 9 and references therein), often for solely tree-configurational reasons, and all the other arguments are listed within the predicate’s comp(lement)s value. In the case of a 2-argument verb such as love this difference is not conspicuous, but in the case of, say, give, the two non-subject arguments would be elements of the comp list, whether they are realised as a direct object and a theme object, as in (2), or as a direct object and goal oblique, as in (3). Hence, the two attributes, subj and comps, suffice for any configuration of arguments.15

Note that this is a difference between LFG and HPSG qua linguistic theories, not qua linguistic formalisms. Either approach can be simulated in the other formalism. For example, within LFG, Alsina (1996) proposes to constrain explicitly named grammatical functions to subject and object, and Patejuk & Przepiórkowski (2016) and Przepiórkowski (2016) further justify this approach and provide an LFG formalisation inspired by HPSG analyses of extended argument structure.16

15 Also, the subj/comps dichotomy is not assumed in some versions of HPSG (including the early versions of Pollard & Sag 1987 and Pollard & Sag 1994: Chapters 1–8, as well as the Sign-Based Construction Grammar of Sag 2012, sometimes perceived as a version of HPSG) and in HPSG grammars of some languages (e.g., German; Stefan Müller, p.c.).

16 Two further – more formal – arguments for HPSG-like representations of grammatical functions in essentially LFG settings may be found in Johnson (1988: Chapter 4): first, they obviate the need for the LFG principles of completeness and coherence (cf. Section 3.3.4), which are encoded via formally problematic (cf. Section 4.3.3) constraining statements, and second, they
Conversely, explicit information about grammatical functions of particular arguments could be added to HPSG representations, as in Ackerman & Webelhuth (1998) or Hellan (2019).

2.3.3 Word forms

The final difference between the two representations in Figures 1 and 2 that I would like to point out concerns the place of the word string in these representations. Traditionally, in LFG the sequence of word forms – the form of the utterance – is the yield of the c-structure, i.e., the sequence of leaves. So finding an LFG representation of an utterance amounts to finding a grammatical representation in which the yield of the c-structure is that utterance.

On the other hand, in HPSG the sequence of words in an utterance is the value of that utterance’s PHON attribute. This means that finding an HPSG representation of an utterance boils down to finding a grammatical structure in which the value of PHON is the list of words in that utterance. Normally this amounts to the same sequence of words as that read off the leaves of the constituency tree. For example, if – in a simple binary tree – the PHON of the first constituent is \( \langle \text{come} \rangle \) and the PHON of the second is \( \langle \text{together} \rangle \), then the PHON of the mother is \( \langle \text{come, together} \rangle \) rather than \( \langle \text{together, come} \rangle \) (or \( \langle \text{drive, my, car} \rangle \), or whatever). This correspondence is explicitly present in the representation in Figure 2 and implicitly assumed in the shorthand representation in Figure 3, but there is a well-developed linearisation theory in HPSG which allows for controlled violations to this correspondence.

I will have more to say about the exact role of the string of word forms in both linguistic theories in Section 3.3.1.

2.4 Summary

Let us take stock of similarities and differences between the kinds of representations assumed in LFG and HPSG. The celebrated difference between the multi-level architecture of LFG and the monolithic structures assumed in HPSG is certainly important to many practitioners of both theories and has an impact on readability (of LFG representations) and the need to apply additional conventions and abbreviations (to render HPSG representations), but in my view it is

lead to an analysis of Dutch infinitive constructions which, unlike the standard – at that time – LFG analysis, does not violate the offline parsability constraint (cf. Section 3.3.3). (Some problems with Johnson’s (1988) own analysis of Dutch infinitive constructions are pointed out in Zaenen & Kaplan 1995.)
of little substantial consequence. It is trivial to devise a lossless conversion of LFG representations to HPSG-like representations, and also HPSG structures may be converted to LFG-like representations which distinguish between constituency structures, structures representing other syntactic information, and semantic structures.

However, there are at least two more substantial differences conspicuous in the representations in Figures 1 and 2. One concerns grammatical functions: one function per argument in LFG and just one distinguished argument in HPSG. The other concerns the place of the sequence of words which make up an utterance: in LFG this sequence is commonly assumed to correspond to the sequence of leaves in the c-structure, while HPSG allows for dissociation between the string of words and the constituency structure.

3 Grammars

What kinds of grammars lead to representations such as those in Figures 1 and 2? I will first consider LFG, then HPSG, and then I will compare the two approaches.

3.1 LFG

Here is the relevant part of an LFG grammar that produces the structures in Figure 1.\textsuperscript{17}

Grammar rules:

(4) \[ S \rightarrow \text{NP} \uparrow_{\text{subj}} = \downarrow \quad \text{VP} \uparrow = \downarrow \]
\[ (\downarrow \text{case}) = \text{nom} \quad (\downarrow \text{tense}) \]

(5) \[ \text{VP} \rightarrow \text{V} \uparrow = \downarrow \quad \text{NP} \]
\[ (\uparrow \text{obj}) = \downarrow \]
\[ (\downarrow \text{case}) = \text{acc} \]

(6) \[ \text{NP} \rightarrow \text{ Pron} \quad \uparrow = \downarrow \]

\textsuperscript{17}Only the core machinery is assumed here, mostly (apart from the \( \sigma \)-projected s-structures) present already in Kaplan & Bresnan (1982). See, e.g., Dalrymple et al. (2019: Chapter 6), for later additions such as functional uncertainty (including inside-out functional uncertainty), off-path constraints, the restriction operator, local names, templates, etc.
Lexicon:

(7) **loves** V

(↑ PRED) = ‘LOVE(SUBJ,OBJ)’
(↑ SUBJ INDEX PERS) = c 3
(↑ SUBJ INDEX NUM) = c SG
(↑ TENSE) = PRS
(↑ REL) = LOVE
(↑ ARG1) = (↑ SUBJ)σ
(↑ ARG2) = (↑ OBJ)σ

(8) **she** Pron

(↑ PRED) = ‘PRO’
(↑ PRONTYPE) = PERSONAL
(↑ CASE) = NOM
(↑ INDEX PERS) = 3
(↑ INDEX NUM) = SG
(↑ INDEX GEND) = F

(9) **you** Pron

(↑ PRED) = ‘PRO’
(↑ PRONTYPE) = PERSONAL
(↑ INDEX PERS) = 2

LFG grammars may be viewed as Context-Free Grammars (CFGs) with annotations; the purely CFG part of the grammar in (4)–(9) is this:

(4’) S → NP VP
(5’) VP → V NP
(6’) NP → Pron
(7’) V → loves
(8’) Pron → she
(9’) Pron → you

Within annotations, ↑ refers to the f-structure associated (via the ϕ function) with the mother node in the tree (i.e., with the preterminal node, in the case of lexical entries), and ↓ refers to the f-structure associated with the current node. For example, the functional equation (↑ SUBJ) = ↓ under the NP in rule (4) for S says that the f-structure associated with the S node has the SUBJ attribute whose value is the f-structure associated with the NP node. The other equation under the NP, (↓ CASE) = NOM, says that the f-structure for this NP has CASE with value NOM. The head equation ↑ = ↓ under the VP in the same rule says that S and VP are associated with the same f-structure.
These are so-called “defining equations” – they may be thought of as constructively building representations. The statement ($\downarrow$ TENSE) under VP in rule (4) is a constraining condition requiring the presence of the TENSE attribute within the f-structure associated with the VP node. This constraining condition cannot be replaced with a defining equation such as ($\downarrow$ TENSE) = PRS because infinitive verbs are assumed not to have the TENSE attribute at all, so the effect of such a defining equation would be to wrongly add the attribute TENSE (and its PRS value) to f-structures of such tenseless forms. Similarly, the constraining equation ($\uparrow$ SUBJ INDEX PERS) = $c$.3 in the lexical entry (7) for loves requires that the f-structure associated with the subject of this verb have the attribute INDEX whose value has the attribute PERS whose value is 3, but the verb does not itself assign this value – some other part of the grammar (in this case, the lexical entry (8) for she) must take care of that. As we will see in Section 4.3.3, the existence of such constraining statements presents a difficulty for model-theoretic formalisations of LFG.

While the symbols $\uparrow$ and $\downarrow$ only implicitly refer to the function $\phi$ mapping c-structures to f-structures, the $\sigma$ function mapping f-structures to s-structures is mentioned explicitly in some of the statements. For example, the statement ($\uparrow_{\sigma}$ ARG1) = ($\uparrow$ SUBJ)$_{\sigma}$ in the lexical entry (7) rather concisely says that there is an s-structure associated with the f-structure related to the preterminal V, this s-structure contains the attribute ARG1, and the value of this attribute is the s-structure associated with the f-structure which is the value of SUBJ within the f-structure related to this preterminal. It is easy to check that the representation in Figure 1 reflects this and all the other statements presented in this subsection.

3.2 HPSG

Theoretical HPSG grammars have a very different feel: they do not have a CFG backbone, but rather contain statements about various types of linguistic objects – not only phrases and words, but also valencies, contents, cases, etc.\(^\text{18}\) HPSG grammars consist of two parts: a type hierarchy (already mentioned in Section 2.2, also called “sort hierarchy” and “signature”) and a theory proper.

A small fragment of the type hierarchy assumed in the AVM of Figure 2 was given in Figure 4, and a much larger part is presented in Figure 7. This type hierarchy seems to mention all types occurring in Figure 2, but in fact it does not contain types for the word forms which appear within PHON values; on the simplest approach to values of PHON each word form is an atom of a type such as

\(^{18}\)However, some such statements, namely, Immediate Dominance Schemata (Pollard & Sag 1994: Section 1.5), directly encode some of the effects of phrase structure rules.
she or loves. (On a more comprehensive approach such as Höhle 1999, values of phon are highly structured and contain various kinds of phonological information.) In a more realistic grammar, the type hierarchy would also contain more subtypes of headed-phrase (see Abeillé & Borsley 2021: Sections 5–6 and references therein), a much larger type subhierarchy below content (see, e.g., Richter & Sailer 1997, 1999 and Davis 2001 for two very different proposals targeting different aspects of semantic representations), a multiple inheritance hierarchy of subtypes of head (Malouf 1998), many more subtypes of vform, etc. As shown in Figure 7, type hierarchies are more than just plain taxonomies of types: they also determine attributes that may occur in structures of particular types, as well as types of values of such attributes.

Theory proper is a set of statements – often called principles – which impose additional, more complex constraints. For example, the famous Head Feature Principle (HFP) says that, in a headed-phrase, the mother has the same value of the head attribute as the head daughter. Formally, this principle may be stated as follows:

(10) Head Feature Principle:

\[
\text{headed-phrase} \Rightarrow \left[ \text{synsem}|\text{cat}|\text{head} \right]_{1} \left[ \text{hd-dtr}|\text{synsem}|\text{cat}|\text{head} \right]_{1}
\]

Such principles are understood universally: every linguistic object must satisfy them. For this reason they are usually implicational, with the antecedent defining the scope of the principle. In the case of (10), either an object is of type headed-phrase, so the antecedent is true and, hence, the consequent must also be true, or the object is not of this type, in which case the antecedent is false and the whole implication is trivially true.

The AVM in Figure 2 describes a configuration of objects containing two objects of type headed-phrase, i.e., satisfying the antecedent of HFP: the root object of type hd-subj-ph and its hd-dtr value of type hd-comp-ph. Both satisfy HFP – see the three occurrences of (10) in that figure. All other objects in this configuration satisfy HFP trivially, as they are not described by the antecedent of HFP; this holds for the word objects representing she, loves, and you, the synsem objects which are values of the synsem attribute, the list objects which are values of various occurrences of phon, subj, comps, and dtrs, etc.

There are also constraints relating the values of val and dtrs. The role of valency attributes is similar to the role of slashes in Categorial Grammar (Ajdukiewicz 1935; Lambek 1958; see also Kubota 2021 and references therein) – they express information about the combinatory potential of an element. For example, the word structure for loves in Figure 2 specifies that this word expects
Figure 7: A larger part of an HPSG type hierarchy
a complement and a subject. Once it combines with the complement you, the
mother phrase of type \textit{hd-comp-ph} needs only a subject in order to be a fully
saturated phrase (i.e., a sentence) – its \textit{comps} list is empty (“⟨ ⟩” is a synonym of
the \textit{elist} type in Figure 7). Moreover, once this phrase combines with the subject
she, both valency lists become empty. This behaviour is regulated by principles
such as the following:

\begin{align*}
\text{(11) Valence Principles (modified and simplified):} \\
\text{a. } \textit{hd-subj-ph} \Rightarrow \\
\begin{array}{l}
\text{SYNSEM|CAT|VAL } [\text{SUBJ } ⟨⟩] \\
\text{COMPS } ⟨⟩ \\
\text{DTRS } ⟨⟨\text{SYNSEM }2⟩, 0⟩ \\
\text{HD-DTR } 1 [\text{SYNSEM|CAT|VAL } [\text{SUBJ } ⟨2⟩] ] \\
\text{COMPS } ⟨⟩ \\
\end{array}
\end{align*}

\begin{align*}
\text{b. } \textit{hd-comps-ph} \Rightarrow \\
\begin{array}{l}
\text{SYNSEM|CAT|VAL } [\text{SUBJ } 0] \\
\text{COMPS } ⟨⟩ \\
\text{DTRS } ⟨1, [\text{SYNSEM }2]⟩ \\
\text{HD-DTR } 1 [\text{SYNSEM|CAT|VAL } [\text{SUBJ } 0] ] \\
\text{COMPS } ⟨2⟩ \\
\end{array}
\end{align*}

The constraint (11a) is saying that, in phrases of type \textit{hd-subj-ph}, the head
daughter (1) only requires a subject (its \textit{comps} list is empty), this subject (2) is
the (\textit{SYNSEM} value of the) first daughter of the phrase, while the second daughter
(0) is the head daughter, and the phrase itself is fully saturated (both \textit{subj} and
\textit{comps} are empty). Similarly, (11b) is saying that, in phrases of type \textit{hd-comps-ph},
the single \textit{comps} element of the head daughter is realised as its second daughter,
the first daughter being the head, and the phrase does not expect a complement
anymore. On the other hand, it still expects whatever subject (if any) is expected
by the head daughter. The actual Valence Principle assumed in HPSG is more
general; in particular, it allows for longer \textit{comps} lists and the realisation of mul-
tiple arguments in a single local tree (see, e.g., Pollard & Sag 1994: 348).

Note that the values of valency attributes are lists of \textit{synsem} structures (see 2
in (11)), not whole phrases. This is an attempt to encode locality constraints on
selection: a predicate may specify its arguments only by providing the kind of
information that is encoded in \textit{SYNSEM} values, so it cannot select an argument
on the basis of its \textit{phon} value or with reference to the internal constituency
structure of that argument (as it is encoded in the values of \textit{dtrs} and \textit{hd-dtr}).\footnote{Note also that these principles do not say anything about values of \textit{phon}. We will deal with \textit{phon} values shortly, in Section 3.3.1.}
What about the lexicon? HPSG has full-fledged theories of the hierarchical lexicon, which make it possible to encode various generalisations across lexical items (see Davis & Koenig 2021 and references therein), but for the purpose of this comparison the simple principle in Figure 8 will do.

Figure 8: Word Principle
What this principle is saying is that any word object must either satisfy the description in the first disjunct (which defines the word she), or the second disjunct (you), or the third disjunct (loves), etc. Again, it is easy to see that the structure described by the AVM in Figure 2 complies with this principle.

All the principles given or alluded to above constrain the shape of signs—words and phrases—but principles may also refer to other types of objects. For example, the type hierarchy in Figure 7 only says that values of SUBJ and COMPS are lists, but the values of SUBJ cannot be of any length—their maximum length is one (a single predicate cannot have two subjects). This can be regulated with the constraint in (12) or—equivalently (given the type hierarchy in Figure 7)—but more concisely—(13).

\[(12) \quad \text{valence} \Rightarrow \text{[SUBJ\ ellist]} \lor \text{[SUBJ\ REST\ elist]}\]

\[(13) \quad \text{valence} \Rightarrow \neg\text{[SUBJ\ REST\ nelist]}\]

Moreover, values of SUBJ and COMPS cannot be just any lists— they must be lists of synsem objects. This may be achieved via constraint (14), whose antecedent is not just a type specification, with the predicate list-of-synsems defined as in (15).

\[(14) \quad \text{[SUBJ} \quad \text{COMPS} \quad \text{]} \Rightarrow \text{list-of-synsems([1])} \land \text{list-of-synsems([2])}\]

\[(15) \quad \text{list-of-synsems(elist)}.
\quad \text{list-of-synsems}([\text{FIRST} \quad \text{synsem} \quad \text{REST} \quad \text{0}]) \iff \text{list-of-synsems([0])}.\]

This simple constraint illustrates an important aspect of contemporary HPSG, namely, the possibility to define and use in constraints any relation (Richter 1999, 2004). The notation for defining such relations is inspired by the programming language Prolog. The definition in (15) consists of two clauses jointly specifying what kinds of objects have the list-of-synsems property: the first clause says that the empty list is a list of synsems, and the second (recursive) clause says that a non-empty list whose first element is a synsem object is a list of synsems if the rest of this list is a list of synsems. Nothing else is a list of synsems.

---

\(^{20}\)I extend the notational conventions defined in Richter (2004: Section 3.2) in such a way that boxed variables appearing in the antecedents of implications are understood as bound by universal quantifiers scoping over the whole formula. So, the quantificational schema of (14) is:

\[\forall \text{[1]} \forall \text{[2]} (\phi([1] [2]) \Rightarrow \psi([1] [2])).\]
3.3 Comparison

3.3.1 Word order

One clear difference between the two frameworks stems from the fact that LFG grammars – but not HPSG grammars – are based on a CFG backbone. Traditionally (but see below) the sentence string is the yield of the c-structure, i.e., it is read off the leaves of the tree. In the case of free word order languages, this leads to trees in which functionally related constituents – for example, a noun and its adjectival modifier – are not always directly related configurationally.

Consider the Warlpiri sentence (16) from Simpson (1991: 257).

    child-DU-ERG PRS-3.DU dog.ABS chase-NPST small-DU-ERG
    ‘Two small children are chasing the dog.’
    ‘Two children are chasing the dog and they are small.’

In this example, *wita-jarra-rlu* ‘small’ is a modifier of *kurdu-jarra-rlu* ‘children’, but on LFG analyses they do not form a constituent, as other constituents linearly intervene between these two words. For example, Austin & Bresnan (1996: 225) propose an analysis which results in the c-structure in Figure 9 (cf. Dalrymple et al. 2019: 112).

![Figure 9: LFG c-structure of (16)](image-url)
By contrast, it is possible to propose an HPSG analysis of Warlpiri word order on which *wita-jarra-rlu* ‘small’ and *kurdu-jarra-rlu* ‘children’ do form a constituent in (16) (in the sense in which the attribute DTRS represents immediate constituents). A shorthand and very schematic representation of the result of such an analysis is given in Figure 10 (after Donohue & Sag 1999: 13). Note that the order of words within the PHON value of the root of this tree is different from the order of PHON values of the leaves.

This analysis is possible because values of PHON are subject to the same constraints as any other structures. The usual tree behaviour, with PHON values of the mother being the concatenation of the PHON values of the daughters in the order in which they occur on the DTRS list, could be simulated with the constraint in (17), where append-phones ([2], [1]) holds if [1] is the concatenation of PHON values of the elements of [2].

\[(17) \begin{array}{c}
\text{phrase} \\
\text{PHON [1]} \\
\text{DTRS [2]}
\end{array} \Rightarrow \text{append-phones ([2], [1])}\]

If the constraint in (17) were included in the grammar of Warlpiri, the representation in Figure 10 would be ill-formed.

However, other definitions are possible, which relax this usual approach. In fact, there is a long history of such linearisation accounts in HPSG, dating back to Reape (1992, 1996), Kathol & Pollard (1995), and Kathol (1995, 2000) (see also Müller 2021 and references therein); such a relaxed approach to word order is commonly assumed in HPSG analyses of ellipsis and coordination (see Nykiel & Kim 2021, Abeillé & Chaves 2021, and references therein). On such analyses, the two sentences in (18) (from Chaves 2008: 286) have exactly the same constituency structures but differ in PHON values.

\[(18) \begin{array}{ll}
a. & \text{Tim gave a rose to Mary and a tulip to Sue.} \\
b. & \text{Tim gave a rose to Mary and Tim gave a tulip to Sue.}
\end{array}\]

---

21 Formally, the relation append-phones is defined as in (i), and the relation append it relies on – as in (ii):

\[(\text{i) append-phones (elist, elist)}).
\]\n
\[
\text{append-phones}\begin{cases}
\text{FIRST [PHON [1]}, [3]) \iff \text{append-phones ([2], [1]) \land}
\text{append ([1], [2], [3])}. \\
\text{REST [2]} \end{cases}
\]

\[(\text{ii) append (elist, list, elist)}).
\]\n
\[
\text{append}\begin{cases}
\text{FIRST [2], [3]list, [FIRST [4]}, [3]) \iff \text{append ([2], [3], [4])}. \\
\text{REST [2] \end{cases}
\]
Figure 10: HPSG constituency structure of (16), with the form escaping the default word order constraints marked in bold.
In LFG such a relaxed approach to word order is also in principle possible, on the assumption that there is a representation of the sentence string separate from c-structure. Such a separate string structure – sometimes called s-string (Dalrymple et al. 2019: Section 3.5) – is programmatically proposed in Kaplan (1987) and substantiated in Wescoat (2002), Asudeh (2009) and, especially, Dalrymple & Mycock (2011), among other works, but it is commonly assumed that the order of words in this additional string structure is the same as the order of leaves in c-structure. One exception to this common assumption are the analyses of cliticisation in Bögel et al. (2010) and in Lowe (2016), on which the position of clitics in the s-string may differ from their position in the tree. However, to the best of my knowledge, there are no LFG analyses which would make a more substantial use of the possibility of relaxing the mapping between the s-string and the c-structure, analogous to those common in HPSG accounts of ellipsis.

### 3.3.2 Optionality of attributes

As has already been alluded to above and as will become fully clear in Section 4, grammars may be understood as theories describing certain linguistic objects. Figures such as 1 and 2 are representations of such objects. Both these figures represent all information that follows from all grammatical rules and principles of the respective grammars sketched in this section, but there is a sense in which the LFG representation in Figure 1 is complete while the HPSG representation in Figure 2 is only partial: it represents the effects of all constraints in the grammar proper, but it does not contain all information that follows from the type hierarchy.

Let us have a closer look at the index values within structures corresponding to the word you. In both representations in Figures 1 and 2 this value is represented as an AVM with just one attribute, pers, with a value indicating 2nd person. In the case of LFG, this is a complete description of the underlying feature structure; the linguistic object described by this subsidiary AVM has exactly one attribute: pers. However, in the case of HPSG, the corresponding AVM is marked as representing a structure of type ref(erential index) and – according to the type hierarchy in Figure 7 (and the standard HPSG type system; see Pollard & Sag 1994: 399) – every ref object has exactly three attributes: pers(on), num(ber), and gend(er). Thus, the subsidiary AVM representing the value of index for you is only a partial description of a complete linguistic object; any such object must also have specific values of num (sg or pl) and gend (m, f, or n). That is, this

\[ \text{pers} \]

\[ \text{num} \]

\[ \text{gend} \]
subsidiary AVM describes six different kinds of linguistic objects, differing in number and gender.

This technical difference between the two formalisms reflects a potentially important linguistic difference between the two theories: to what extent are the described linguistic objects allowed to be partial or indeterminate? Such partial objects were the staple of the original HPSG of Pollard & Sag (1987), where the described objects were understood not as strictly linguistic objects but rather as informational objects – bits of information (including disjunctive and negative information) that competent speakers have about language. But it seems that LFG sides with the latter-day HPSG in describing linguistic objects rather than informational objects. So the difference between the two representations of the index value for you seems to be a linguistically contentful – and potentially verifiable – difference: on the LFG view the pronoun you is specified for person but unspecified or neutralised for number and gender, while on the HPSG view it is ambiguous between different values of number and gender.

Interestingly, it is easy to simulate the HPSG approach in LFG, but it is far from obvious how to simulate the LFG approach in HPSG. In LFG, the lexical entry of you could be extended from (9) to (19), with the last two statements requiring that num and gend be present and have values within appropriate sets:

(19) you       Pron (↑ pred) = ‘pro’
   (↑ prontype) = personal
   (↑ index pers) = 2
   (↑ index num) ∈ {sg, pl}
   (↑ index gend) ∈ {m, f, n}

This leads to six different f-structure representations of the pronoun you, differing in the values of num and gend.

Within HPSG, a more complex type subhierarchy could allow for different subtypes of ref, one of which would only be specified for the attribute pers; let us call this subtype ref-pers. Then, the pronoun you could have the index value of type ref-pers, with pers equal to 2. Another subtype, let us call it ref-pers-num, would be specified for pers and num, and it would be appropriate for index values of pronouns I (pers 1, num sg) and we (pers 1, num pl). This solution, however, is problematic in view of the following examples:

(20) Creatures, I give you yourselves... (C.S. Lewis, The Magician’s Nephew)
(21) Creature, I give you yourself...

---

23 See also Kaplan (2019) for a discussion of this and related issues.
24 See Richter (2004: Chapter 2) for a discussion of the differences between early and later HPSG.
The anaphoric pronouns *yourselves* and *yourself* are specified for person (2nd) and for number (plural and singular, respectively), but not for gender, so they should have index values of type *ref-pers-num*. But, given the standard HPSG binding theory (cf. Section 4.3.2 below), these index values should be equal to the index values of the binder – the pronoun *you* in both examples above – so they should be of type *ref-pers*. The only way this is possible is that the two types, *ref-pers* and *ref-pers-num*, have a common subtype. But this common subtype would have to inherit the attribute *num* from *ref-pers-num*, so *ref-pers* would have a subtype with attribute *num*. Given that all objects in HPSG models – including all values of attributes – must bear maximally specific types (this will be made clear in Section 4.1 below), the pronoun *you* would be ambiguous: on one interpretation its index would have a value (of this shared subtype) with *num sg*, on the other – with *num pl*. This would contradict the original motivation for the multiple subtypes of *ref*, namely, to make the pronoun *you* indeterminate with respect to number and gender, rather than ambiguous. It is not clear to me how to simulate within HPSG the behaviour of LFG – that is, how to make the pronoun *you* indeterminate with respect to number by default (i.e., apart from binding contexts such as (20)–(21)) – without complicating the standard HPSG binding theory.

In summary, while either approach may perhaps be simulated in the other theory, HPSG analyses naturally lead to a multiplicity of models differing in ways that linguists often do not care about, while LFG grammars naturally specify fewer linguistic objects, differing only in linguistically relevant aspects. We will return to this issue in Section 4.3.

### 3.3.3 Expressiveness

What is the relation of LFG and HPSG to the Chomsky–Schützenberger hierarchy of grammar formalisms (Chomsky 1956)? That is, what classes of languages do possible LFG and HPSG grammars describe? This question cannot be answered without making the notion of a “possible LFG/HPSG grammar” more precise. Given that both theories evolve and that at any particular point there are competing proposals about various aspects of the theories, this notion is not fully explicit and perhaps never will be.

Nevertheless, it is possible to ask about the complexity of the underlying formalisms, and it is clear that – without additional constraints – both are equivalent to Turing machines, i.e., they may describe any language that is algorithmically describable at all. There is no space here to formally prove this claim, but it is based on the well-known fact that attribute–value grammars may encode Turing
machines (Johnson 1988: Section 3.4.2; see also Kaplan & Bresnan 1982: fn. 32). In particular, the unification grammar schema for simulating the effect of any Turing machine (i.e., for defining the same language as that recognised by that Turing machine) presented in Francez & Wintner (2012: Section 6.2) can be easily encoded in the formalisms underlying LFG and HPSG.\footnote{In the case of LFG, the schemata $\rho_1-\rho_8$ of Francez & Wintner (2012: 230–232) can be directly translated into LFG phrase structure rules by taking cat values to be node labels and by encoding all the other information present in the AVMs in $\rho_1-\rho_8$ via straightforward functional equations. In the case of HPSG, these schemata may be encoded as Immediate Dominance Schemata (Pollard & Sag 1994: Section 1.5), with an additional phon attribute collecting the terminal symbols (dually to how they are collected in the values of the left attribute in Francez & Wintner 2012). Schemata $\rho_1-\rho_8$ are essentially – appropriately annotated (which is the source of the additional complexity) – right-linear grammars with binary branching rules for reading the terminal symbols and with unary branching rules – including an empty production – for simulating a Turing machine. It is easy to modify this encoding to get rid of the empty production (the unary rules encoding transitions of a Turing machine could be used at the top of the tree instead of at the right-hand bottom), but the use of effectively unary rules with possible repetitions of non-terminal symbols along unary chains is non-negotiable.}

Given this formal power of the underlying formalisms, the recognition problem (given a grammar and a sentence, is this sentence predicted by this grammar?) is undecidable – there is no general algorithm which could take an arbitrary grammar and sentence and always answer that question in finite time. In the case of LFG, this potential problem was recognised very early and a solution was proposed (Kaplan & Bresnan 1982: 266–267) in terms of what later became known as offline parsability (Pereira & Warren 1983: 142): a global condition on constituency structures, namely, that, first, they do not contain unary chains (subtrees with only unary branching) in which the same category appears twice and, second, that they do not use empty productions (i.e., that there are no empty leaves in the tree). The encoding of Turing machines in Francez & Wintner (2012: Section 6.2) violates both conditions (cf. fn. 25). A different way to make LFG grammars tractable is proposed – and references to other attempts are given – in Wedekind & Kaplan (2020).\footnote{Simplifying, Wedekind & Kaplan (2020) require of grammars that there be an upper bound on the number of different c-structure nodes that may map to the same f-structure.}

In the case of HPSG, the dominant underlying formalism (RSRL, Richter 1999, 2004; see Section 4.1) is known to be undecidable (Kepser 2004). A different formalisation, based on an extension of modal logic (namely, polyadic dynamic logic), is proposed and shown to have more desirable complexity properties in Søgaard & Lange (2009) but, to the best of my knowledge, it has remained largely unnoticed within the HPSG community.

Let me reiterate, however, that any less than desirable complexity results mentioned above pertain to formalisms underlying the linguistic theories, not to the
theories themselves. As has been repeatedly noted in both frameworks (see, e.g., Kaplan & Bresnan 1982: 271–272, Johnson 1988: 94–95, and Richter 2004: 242–243), it is very well possible that linguistic constraints sufficiently delimit the space of possible grammars to make the recognition problem decidable and efficient, and – conversely – it is also possible that human languages are in fact undecidable. That means that high complexity results for a formalism underlying a linguistic theory should not necessarily be held against that theory.

3.3.4 Generative-enumerative or model-theoretic?

Pullum & Scholz (2001) divide syntactic frameworks into “generative-enumerative” (GE) and “model-theoretic” (MT). GE frameworks have a derivational feel: at their centre are instructions for rewriting certain strings or structures into other strings or structures. Typical examples are formal grammars in the sense of the Chomsky hierarchy, for example, CFGs such as that in (4′–9′), where particular rules are such instructions. In the top-down mode, one starts with the string “S” and uses the rules to rewrite any non-terminal symbols – e.g., the rule (4′) to replace “S” with “NP VP” – until the resulting string contains only terminal symbols, e.g., “she loves you”. In the bottom-up mode, one starts with a string of terminal symbols, e.g., “she loves you”, and uses the rules in the other direction, until the resulting string “S”, e.g.: “she loves you” → “she loves Pron” → “she loves NP” → “she V NP” → “she VP” → ... → “S”. The language defined by a grammar is the set of those strings of terminal symbols for which this procedure succeeds. Examples of GE systems are various transformational grammars, Categorial Grammars, Tree-Adjoining Grammars, etc. GE frameworks have an analogue in syntactic – proof-theoretic – aspects of logic.

By contrast, MT frameworks have an analogue in semantic – model-theoretic – aspects of logic. Grammars are sets of logical formulae which may be understood as defining models (namely, those models in which all the formulae are true). An early – historical – example is Arc-Pair Grammar, but currently HPSG seems to be the most clear case of an MT framework (Pullum 2019: 60). We will have a closer look at models of HPSG grammars shortly, in Section 4.1.

Some GE frameworks have a somewhat mixed character: they have a GE backbone but they also impose certain constraints on the resulting structures.27 Two examples are the transformational grammar of the 1980s (GB; Chomsky 1981, 1986) and, to some extent, Generalized Phrase Structure Grammar (GPSG; Gazdar et al. 1985). It seems that, at least as originally conceived, LFG belongs in

27Thanks to Geoff Pullum for discussion and for the clarification that such “mixed” frameworks should still be classified as unambiguously GE.
the same category: there is a generative CFG backbone responsible for building c-structures (Kaplan & Bresnan 1982: 175), but also for generating functional statements which act as constraints on f-structures associated with particular c-structure nodes (Kaplan & Bresnan 1982: 181). The following quote makes this dual nature of the original LFG particularly clear:

A string’s constituent structure is generated by a context-free c-structure grammar. The grammar is augmented so that it also produces a finite collection of statements specifying various properties of the string’s f-structure.

(Kaplan & Bresnan 1982: 180–181)

If such statements – i.e., functional equations – cannot be satisfied, then the whole description for a given input fails, even if the c-structure rules produced an appropriate constituency tree for this input. The functional component thus acts as a filter on the output of the c-structure component (as explicitly stated in Kaplan & Bresnan 1982: 203–204).

Also some general LFG principles are formulated as constraints on possible f-structures (Kaplan & Bresnan 1982: 178–179, Dalrymple et al. 2019: Section 2.4.6): completeness and coherence jointly state that, simplifying a little, grammatical functions mentioned in pred values must be exactly the grammatical functions occurring as attributes. The main f-structure in Figure 1 satisfies this constraint: pred mentions subj and obj and these are exactly the attributes which characterise grammatical functions in this f-structure. Similarly, f-structures which are values of subj and obj satisfy this constraint: their pred values do not mention any grammatical functions and none appears as an attribute in these f-structures.

Generative-enumerative frameworks may often be given model-theoretic reformulations. McCawley (1968) is usually credited with the observation that phrase structure rules may be understood as conditions on trees,28 and fully-worked out MT equivalents of various GE formalisms were proposed by Rogers (1997, 1998).

While there is no comprehensive MT formalisation of LFG, the description of the general architecture of LFG in Kaplan (1989: Section 2) is formulated in terms of conditions on particular structures, also on c-structures, and on correspondences between them,29 and this view is prevalent in contemporary LFG.30 For this rea-

29The slightly modified version of Kaplan (1989) published a few years later explicitly invokes “model-based approach” as “of course, the hallmark of LFG” (Kaplan 1995: 11). An earlier model-theoretic formalisation of an LFG-like formalism (but without the distinction between defining and constraining statements) is Johnson (1988). See also Blackburn & Gardent (1995) for another attempt (also limited to defining statements; cf. Börjars & Payne 2013).
30For example: “In LFG, phrase structure rules are not rewrite rules, rather they are ‘node admis-
son, Pullum & Scholz (2001: 20) classify “recent LFG” as “perhaps” MT. I will have much more to say about model-theoretic aspects of LFG and HPSG in Section 4.

3.4 Summary

In this section we looked at two rather specific differences between LFG and HPSG grammars and two more general aspects. One specific difference concerns word order: in HPSG, but not in LFG, the string is often – especially, in analyses of ellipsis – assumed to be dissociated from the constituency structure. The other concerns determinacy: HPSG analyses often lead to multiple structures, i.e., to ambiguity, while LFG analyses more naturally lead to more compact indeterminate structures. Interestingly, despite the expressive power of the two theories, it is not always clear how to elegantly simulate in one theory the analysis commonly assumed in the other.

The two more general issues are expressivity and relation to the generative-enumerative vs. model-theoretic dichotomy postulated in Pullum & Scholz (2001). Underlying formalisms of both theories, unless additionally constrained, have the expressive power of Turing machines; such additional constraints were proposed in LFG right at the beginning and are the topic of ongoing work, while much less attention is devoted to the matter of complexity in HPSG. Finally, HPSG is a prototypical model-theoretic theory, while the place of LFG in this dichotomy is less clear, as no explicit model theory has ever been proposed for LFG. This is the issue to which I turn next.

4 Models

Grammars like those discussed in Section 3 are descriptions of collections of linguistic objects, pictures like those in Figures 1 and 2 of Section 2 are representations of particular configurations of such objects, but what exactly are these objects themselves? That is, what are the models of LFG and HPSG grammars? The two theories differ considerably in the extent to which answers to these question are provided: fully explicit model theories are proposed in HPSG, but only sketches and intuitive ideas may be found in LFG. For this reason, in this section I start with HPSG. First, however, a few words about models in general.

Take the following formulae of first-order logic:

\[ \forall x. \text{black}(x) \leftrightarrow \neg \text{white}(x) \]
Together they are saying that everything is either black or white (see 22) and that there is a relation, bw, which holds between black things and white things (see 23) such that every black thing is in this relation with some (at least one) white thing (see (24)) and every white thing is related to some (at least one) black thing (see (25)). Informally speaking, the previous sentence is a description of possible models of formulae (22)–(25). One model is a two-element set such that one element is black, the other is white, and they are related. Another has two black elements and two white elements such that they are pairwise related, i.e., the relation bw denotes two pairs of elements. Another – one that also has two black elements and two white elements – is illustrated in Figure 11. The empty set is also a model, and there are infinitely many other models, both finite (of any cardinality apart from 1) and infinite (of any transfinite cardinality).

![Figure 11: A model – one of many – of (22)–(25)](image)

The meaning of the theory (22)–(25) may be equated with the collection of all models of that theory. However, we may want to exclude some models as not interesting or not really capturing what the formulae (22)–(25) are meant to capture. For example, perhaps we want models to be non-empty and – while possibly arbitrarily large – finite. The first condition may be stated by extending the theory with the formula $\exists x. x = x$.31 However, the second condition, arbitrary finiteness, cannot be stated within first-order logic, so it must be stated meta-theoretically, as an additional constraint on permitted models.32 As we will see below, both theories make use of such meta-theoretical conditions on models.

31 Given the formulae (22)–(25), the same effect may be achieved, e.g., with $\exists x. \text{black}(x)$ or with $\exists x. \text{white}(x)$.

32 Alternatively, a more expressive logic could be adopted.
4.1 HPSG

Of all linguistic theories, HPSG is perhaps unique in its concentrated attention to the issue of what grammars actually describe – what the models of HPSG theories are. There is no place here to summarise the different proposals found in the HPSG literature; some of them are critically discussed in Richter (2004: Section 2.2). Here, I will describe informally – and in terms which facilitate comparison with standard logical models and with potential LFG models – what I assume to be the standard HPSG approach, namely, the model theory of RSRL (Richter 1999, 2004).33

As in mathematical logic, RSRL models are sets of objects which may have various properties and relations defined on them. The properties correspond to the maximal types – called species – of type hierarchies: each object is assigned exactly one species.34 For example, assuming the hierarchy of Figure 7, it is not enough for an object to have the property list; it must be either enlist or nelist. Similarly, any sign object must actually be either a word, or a hd-subj-ph, or a hd-comps-ph, or a non-headed-phrase. In other words, species of HPSG type hierarchies partition sets of objects in HPSG models, just like the properties black and white partition sets of objects in models of the first-order theory (22)–(25).

Attributes correspond to relations. For example, still assuming the type hierarchy in Figure 7, the attribute REST is modelled as a relation between nelist objects and list (i.e., enlist and nelist) objects. Similarly, SYNSEM relates signs (i.e., objects of one of the species: word, hd-subj-ph, hd-comps-ph, and non-headed-phrase) to objects of type synsem (which is a species, according to this type hierarchy). This is similar to the possible interpretations of the relation bw as defined in (22)–(25): the domain of that relation is the set of black objects, and the co-domain – the set of white objects. However, the meanings of HPSG attributes are not just any relations; they are total functions on sets of appropriate species (nelist, in the case of REST) with values in the set of objects of appropriate species (elist and nelist, in the case of REST).

Additional constraints on objects and relations between them are provided by the theory proper, i.e., by principles such as the HFP in (10), repeated below as (26), and the principles in (11)–(14).

34That is, in the HPSG lingo, objects are sort-resolved (Pollard & Sag 1994: 18).
Adam Przepiórkowski

(26) Head Feature Principle:
\[
\text{headed-phrase} \Rightarrow \left[ \text{SYNSEM} | \text{CAT} | \text{HEAD} \, \square \right]
\]

For example, HFP is saying that whenever there is an object of type \textit{headed-phrase} – i.e., of species \textit{hd-subj-ph} or \textit{hd-comps-ph} – there must be other objects related via functions corresponding to \text{HD-DTR}, \text{SYNSEM}, \text{CAT}, and \text{HEAD} as illustrated in Figure 12. In this case, the value of the variable \( \square \) of (26) is the object number 7.

\[hd-subj-ph \, \bullet \, 1\]

\[\text{SYNSEM} \quad \quad \quad \quad \text{CAT} \quad \quad \quad \quad \text{HEAD}\]

\[\text{HD-DTR} \quad 4 \quad \quad \quad \quad \quad \text{SYNSEM} \quad 5 \quad \quad \quad \text{CAT} \quad 6 \quad \text{HEAD}\]

Figure 12: Configuration of objects satisfying the Head Feature Principle

Given the other principles and the type hierarchy, we know much more about this configuration of objects than is explicitly said in Figure 12. For example, the type hierarchy implies that the species of object 4 must be a maximal subtype of \textit{sign}, the species of objects 2 and 5 must be \textit{synsem}, etc. Additional constraints on configurations involving objects of type \textit{hd-subj-ph} are imposed via one of the Valence Principles (namely, (11a)), etc.

Now, HPSG models are simply collections of objects such that each object satisfies all constraints following from the type hierarchy and the theory proper. For example, all seven objects in Figure 12 must satisfy HFP, not just object 1. And they all do, albeit – apart from object 1 – trivially: since objects 2–7 are not of type \textit{headed-phrase}, the antecedent of HFP is false of them and the whole statement is true. But the configuration in this figure is not a complete model. For example, according to the type hierarchy in Figure 7, object 7, which is a value of \textit{HEAD}, must be of type \textit{head}, i.e., of species \textit{verb} or \textit{noun}. If it is a \textit{verb}, there should be \textit{vform} and \textit{bool} objects in the model related to object 7 via attributes \textit{vform}, \textit{aux}, and \textit{inv}. If it is a \textit{noun}, there should be an object related to object 7 via \textit{case}. Similarly, according to the type hierarchy, object 1 should be related to two more objects via \textit{phon} and \textit{dtrs}, and according to the Valence Principle (11a), the value of \textit{dtrs} should be a two-element list, etc.

Since the late 1980s, all approaches to HPSG models agree with this general view of models, but they all impose additional – technical and sometimes philosophical – constraints on what counts as an interesting model. For example, the
empty set is a model (all elements in this set satisfy all constraints), but a trivial one. Also a set consisting of just one object of species *elist* is a model, but it is not interesting. The common view is that HPSG models should be models of whole languages, that they should be exhaustive; in particular, a single exhaustive model contains configurations corresponding to all utterances licensed by the grammar. A little more technically but still very informally, exhaustive models simulate all other models: if there is a structure in some model, then this (or rather, an isomorphic) structure must also occur in an exhaustive model (King 1999). So, within a single exhaustive model, there are configurations of objects corresponding to the AVM in Figure 2, other configurations corresponding to the utterance (2) (*You never give me your money*), and similarly for any other structures licensed by the grammar. This is a somewhat unusual approach to modelling; an analogous exhaustivity requirement in the case of the first-order theory (22)–(25) would mean that only infinite models are admitted, namely those which contain all possible correspondences of black and white objects. We will return to this issue in Section 4.3.

The above considerations still leave open the question: What exactly are the objects in these models? For King (1999) they are bits of reality, actual linguistic tokens (e.g., every utterance of *She loves you* by anybody, ever), but also non-actual – potential – linguistic tokens, i.e., grammatical utterances which have the bad luck of never being actually uttered. This last notion is ontologically dubious, and also leads to proliferation of isomorphic structures within a single model, so it is not frequently subscribed to within the HPSG community. Rather, it is often assumed that the objects in HPSG models are set-theoretic objects – or abstract feature structures – which only stand in conventional correspondence to actual or possible utterances (Pollard & Sag 1994; Pollard 1999). These abstract objects are designed in such a way that – simplifying again – any two isomorphic structures must actually be the same structure. Alternatively, the issue of what exactly these objects are is left unspecified, but an additional requirement is imposed that exhaustive models are minimal in the sense that they contain only one copy of any relevant configuration (Richter 2007).

35 Recall that the AVM in that figure is still an underspecified description, as it does not fix values of *num* and *gend* within the *ref* object marked as [5]. It is also underspecified in other respects, to be discussed in Section 4.3.2.

36 In fact, such models would be so large that they would not be sets anymore, but would rather be proper classes.

37 Also, apart from the curious notion of non-actual tokens, it is not clear what counts as a single utterance token. For example, when John Lennon and Paul McCartney sing together *She loves you*, is this a single token, or two tokens (or perhaps none, because they are singing rather than speaking)? Does the answer depend on whether they sing in unison or in harmony? How many linguistic tokens are there when the song is broadcast on the radio, if any? Does that depend on the number of listeners at different locations?

xxxv
While Kaplan (1995: 11) characterises LFG as “model-based”, no explicit and worked-out model theory has ever been proposed for LFG, as far as I know. Let us, nevertheless, try to construct a possible model corresponding to the representation in Figure 1, a model that is consistent with informal descriptions in Kaplan & Bresnan (1982) and Kaplan (1995).

First of all, the model must contain a collection of objects representing the nodes of the c-structure, as well as a collection of node labels (Kaplan 1995: 10). I assume that both grammatical categories (e.g., S or Pron) and orthographic forms (e.g., she) are labels. There are three relations defined on these objects: M (mother) is the partial function from nodes to nodes, defined on all nodes apart from the root; < is the partial ordering relation on nodes, and λ is a function from nodes to labels. So a part of the model for the representation in Figure 1, one that corresponds to the c-structure, may look as in Figure 13 (with the linear relation < not represented explicitly). There are 18 objects in this model: eight labels (S, VP, V, NP, Pron, she, loves, you) and ten nodes (objects whose exact nature is left unspecified). For an LFG grammar to lead to such models, it must be translated into appropriate formulae, appropriate tree axioms must be stated explicitly, and these axioms should be formulated in such a way that they apply to tree nodes and not to labels or objects corresponding to feature structures.

Kaplan & Bresnan (1982) and Kaplan (1995) are much more explicit about the kinds of objects that correspond to feature structures. There are three types of objects involved in models of feature structures: atoms (e.g., pred, subj, nom, 3, etc.), semantic forms (to the first approximation, strings such as ‘love〈subj,obj〉’), and sets. In particular, feature structures are modelled as finite functions – sets of pairs such that the first element of a pair is an atom and the second element is either an atom, or a semantic form, or a feature structure (i.e., a set again). For example, the AVM in (27) is a representation of the set of pairs in (28), i.e. – given the commonly assumed Kuratowski’s encoding of a pair ⟨a, b⟩ as the set {{a}, {a, b}} (see, e.g., Enderton 1977: 36) – the set in (29).

---

38 Together with Kaplan & Bresnan (1982) and Kaplan (1995), I do not take into consideration sets other than those which model feature structures, i.e., I ignore here coordinate structures, values of the attribute adjunct, etc.

39 One potential problem with this standard LFG understanding of f-structures as sets is that, given the possibility of cyclic f-structures – naturally occurring in analyses of various types of modification, e.g., Johnson (1988: 19–20), Zweigenbaum (1988), and Haug & Nikitina (2012: 298), and in other contexts, e.g., Fang & Sells (2007: 209), Przepiórkowski & Patejuk (2012: Section 4.3.2), and Dalrymple et al. (2020) – sets that are used for modelling f-structures are not the well-founded sets of the standard (Zermelo–Fraenkel) set theory, but must rather rely on the non-standard notion of non-well-founded sets (see Aczel 1988: 103–112 on the history of this notion). To the best of my knowledge, this has not been noticed in the LFG literature so far.
Figure 13: A possible LFG model of the c-structure of *She loves you* (without explicit representation of `<`)

\[
\lambda
\]

\[S\rightarrow M\]

\[VP\rightarrow M\]

\[NP\rightarrow M\]

\[V\rightarrow M\]

\[Pron\rightarrow M\]

\[she\rightarrow M\]

\[loves\rightarrow M\]

\[you\rightarrow M\]

Since some parts of f-structures (values of particular attributes, as well as attributes themselves) may be directly referred to in functional equations, they must all be direct elements of the model. That is, sets representing f-structures cannot be considered unanalysable elements of models; rather, the subsets and atoms within such sets must also be elements of LFG models, so they should be explicitly related by the (converse of the) membership relation \(\in\). Hence, the set in (29) corresponding to the f-structure (27) translates into the configuration of model objects in Figure 14. There are 10 nodes in this configuration that encode particular sets (with node 1 representing the whole f-structure (27)) and six nodes are atoms.
The model in Figure 14 is rather complex, when compared to the simplicity of the AVM in (27). Why not assume the model in Figure 15 instead? Unfortunately, as explained in more detail presently (in Section 4.3.1), this simpler model is incompatible with the LFG idea that attributes and atomic values are ontologically the same kinds of entities, namely, atoms. By contrast, according to the model in Figure 15, atomic values are atoms – objects of the universe of the model – but attributes are binary relations on such objects, i.e., ontologically very different entities. Hence, in the following I will assume the model in Figure 14 as most directly reflecting the LFG view that f-structures are finite functions.

Let me finish this section by noting that configurations in Figures 13 and 14 are fragments of a larger model corresponding to the representation of *She loves you* given in Figure 1. The complete model would also contain strings representing semantic forms, as well as more atoms, many more sets representing the full f-structure, sets representing the s-structure, and relations \( \phi \) and \( \sigma \).

### 4.3 Comparison

#### 4.3.1 Modelling feature structures

It should be clear from the above discussion that AVM representations correspond to very different model configurations in the two theories. For example, while the HPSG model of the AVM in (30), shown in Figure 16, contains just four nodes corresponding directly to the whole index (object 1 of species ref), to 3rd person (object 2 of species 3), to singular number (3 – sg), and to feminine gender (4 – f), the LFG model of the corresponding AVM in (27), shown in Figure 14, contains 16 nodes modelling not only the whole f-structure and the respective values of the three attributes, but also the attributes themselves and various intermediate sets.

\[
\begin{bmatrix}
\text{ref} \\
\text{PERS} \\
\text{NUM} \\
\text{GEND}
\end{bmatrix}
\begin{bmatrix}
3 \\
\text{sg} \\
\text{f}
\end{bmatrix}
\]

This is not an incidental difference between the two theories. In HPSG, attributes such as \text{PERS}, \text{NUM}, and \text{GEND} and types such as 3, sg, and f have very different interpretations: attributes denote relations (partial functions) between

---

40 Compare the HPSG model of (30) in Figure 16 below. Such simpler models, in which feature structures are represented as objects and attributes as relations on objects, are also common in other theories working with AVMs (see, e.g., Blackburn & Spaan 1993: 132–133).
Figure 14: A possible LFG model of the f-structure in (27)

Figure 15: A hypothetical simpler model of the AVM in (27)

Figure 16: An HPSG model of the AVM in (30)
objects in the model, while types denote properties that objects may have. In particular, different objects may – and often do – have the same species, so there can be many objects of type sg, etc. This difference between attributes and types is rendered typographically by using small capitals for attributes and italics for types.

On the other hand, in LFG, attributes such as pers, num, and gend and their atomic values such as 3, sg, and f are the same kinds of objects, namely atoms, each of which may occur in the model just once (there is only one atom sg, etc.). Hence, there is also no typographic distinction between attributes and atomic values of attributes.

This ontological uniformity of attributes and atomic values is taken advantage of in some LFG analyses. For instance, according to the analysis of oblique arguments in Kaplan & Bresnan (1982: 196–201), a “case-marking” preposition which may introduce such an oblique argument defines the value of the attribute pcase to be the oblique function homonymous with this preposition, e.g.:

\[(31) \quad \text{to} \quad \text{P} \quad (\uparrow \text{pcase}) = \text{TO}\]

This feature and its value are also present in the f-structure corresponding to the resulting PP constituent. Verb forms like handed, as used in (32) from Kaplan & Bresnan (1982: 196), expect – apart from any subject and objects – an argument bearing this grammatical function, see (33).

\[(32) \quad \text{A girl handed a toy to the baby.}\]
\[(33) \quad \text{handed} \quad \text{V} \quad (\uparrow \text{pred}) = \text{‘HAND\langle\text{subj, obj, to}\rangle’}\]
\[\quad (\uparrow \text{tense}) = \text{pst}\]

Finally, an appropriate VP rule – simplified here to (34) – contains the crucial equation (35) on the PP:

\[(34) \quad \text{VP} \rightarrow \text{V} \quad \text{NP} \quad \text{PP}\]
\[\quad \downarrow = \uparrow \quad (\uparrow \text{obj}) = \downarrow \quad (\uparrow (\downarrow \text{pcase})) = \downarrow \]
\[\quad (\downarrow \text{case}) = \text{acc}\]
\[(35) \quad (\uparrow (\downarrow \text{pcase})) = \downarrow\]

Applied to the sentence (32), with the PP to the baby, (\downarrow \text{pcase}) in equation (35) evaluates to TO, so the whole equation is equivalent to (\uparrow TO) = \downarrow. Note that TO, the atomic value of pcase of the preposition to, is used here as an attribute indicating an oblique grammatical function. While such double use of atoms as values and attributes is rare in actual LFG analyses, it is not unique to the account of obliques.
in Kaplan & Bresnan (1982); for example, it also occurs in the formalisation of information structure in Dalrymple & Nikolaeva (2011: Sections 4.3.3–4.3.5).

The above considerations do not imply that not distinguishing attributes from atomic values necessarily leads to such complex models as that partially illustrated in Figure 14. For example, Johnson (1988: Section 2.1.3) defines models of f-structures as consisting of a set of atoms, a set of objects directly modelling particular feature structures, and a 2-argument partial function \( \delta \) whose first argument is an f-structure, second argument is an atom \( qua \) attribute, and the value is the value of this attribute in this f-structure. On this approach the AVM in (27) receives a model that may be represented pictorially as in Figure 17. Note, however, that on this view feature structures are no longer sets of \( \langle attribute, value \rangle \) pairs, contrary to Kaplan & Bresnan (1982) and Kaplan (1995).

![Figure 17: A model of the f-structure in (27) as in Johnson (1988)](image)

### 4.3.2 Identity of indiscernibles?

Both theories have trouble with indiscernible structures. Let us illustrate this with sentence (36).

(36) She says she loves you.

Consider the LFG f-structure for this sentence in Figure 18. In the model configuration corresponding to this AVM there are single objects representing particular atoms: just one object \( nom \), one \( sg \), one \(prs\), one \( tense\), etc. Moreover, since feature structures are sets of \( \langle attribute, value \rangle \) pairs, the two \textit{index} values – the substructures marked as \( \mathbb{2} \) and \( \mathbb{4} \) – are the same set (namely, the one in
(29)), so they should be modelled with the same single object in the model (or, more precisely, with a single configuration of objects shown in Figure 14, rooted in the same object 1). The problem is that nothing in our reconstruction of the intended LFG model theory guarantees this: two different models are possible, one in which $\langle 2 \rangle = \langle 4 \rangle$, and one in which $\langle 2 \rangle \neq \langle 4 \rangle$. Only the first of these models properly encodes the idea that feature structures are sets.\footnote{Interestingly, the XLE platform for implementing LFG grammars (Crouch et al. 2008), normally very faithful to the LFG theory, does not treat f-structures as (standard) sets: there, two indiscernible f-structures are assumed to be different objects, unless there is a statement in the grammar that explicitly requires their identity.} We will return to this issue below, when discussing HPSG models.

Figure 18: F-structure for (36)

The bigger problem is that, if f-structures are sets, the two f-structures representing she, i.e., $\langle 1 \rangle$ and $\langle 3 \rangle$ in Figure 18, are the same set-theoretical object. (In the modelling of f-structures suggested above they may be the same object, but – as discussed in the previous paragraph – they do not have to be.) But LFG requires that they be different objects – we do not want to say that the two ‘pro’ values...
in these f-structures necessarily refer to the same person. The way LFG deals with this problem is to assume that \texttt{pred} values – semantic forms – come with unique indices (normally not shown in AVMs), i.e., that whenever an equation like \((\uparrow \texttt{pred}) = \texttt{`pro'}\) is used, a new index is assigned to the semantic form. So the two references to the lexical entry for \textit{she} in (8) that are made in the process of constructing the f-structure in Figure 18 result in two different equations, as if the following two statements were used:

\begin{align*}
(37) & \quad \text{a. } (\uparrow \texttt{pred}) = \texttt{`pro'}_1 \\
& \quad \text{b. } (\uparrow \texttt{pred}) = \texttt{`pro'}_2
\end{align*}

Unfortunately, this mechanism, as it stands, seems to be inherently procedural: at the relevant step of the derivation it must be known which indices have already been used so that a new index can be assigned to a new semantic form. It is not immediately clear how to translate this mechanism to the model-theoretic view of LFG.\footnote{Given that \texttt{pred} values are largely redundant (cf. Section 2.1 and fn. 12), this problem may be solved by removing \texttt{pred} from f-structures altogether. Another – perhaps more conservative – possible solution, suggested by Ash Asudeh (p.c.), is to provide indices with sufficient inherent structure to guarantee their uniqueness. In the simple case of (36), it would suffice to take indices to be the relevant c-structure nodes, but a more complex solution is required to also apply to ‘pro’ values of \texttt{pred} in the case of pro-dropped constituents (especially in languages which allow pro-dropping of multiple arguments of a single predicate).
}

Also HPSG has a problem with stating when exactly indiscernible structures should be treated as being the same structure.\footnote{The problem to be described presently is sometimes called “Höhle’s problem” (Pollard 2001, 2014: 113).} In HPSG, not even atoms are guaranteed to be unique, so one of the models of sentence (36) (\textit{She says she loves you}), whose partial AVM is given in Figure 19, might involve the configuration in Figure 20, with single objects of type 3 and sg and two different objects of type \(f\). Given two different \texttt{ref} objects, there are eight possible configurations of this part of the model, and given also the possibility of two different \texttt{nom} objects, two different \texttt{she} objects (in \texttt{phon} values), different \texttt{elist} objects, etc., there are billions of different models of sentence (36), all described by the AVM in Figure 19, differing in ways that linguists do not care about.\footnote{Each word introduces three lists (values of \texttt{phon}, \texttt{val|subj}, and \texttt{val|comps}), and there are five words in this sentence, so there are 15 \texttt{elist} objects stemming from words alone. The number of different ways to partition a set of \(n\) elements into equivalence classes is given by Bell numbers \(B_n\), and \(B_{15} = 1,382,958,545\) (see https://oeis.org/A000110/list). This should be multiplied by the eight configurations of the two \texttt{ref} objects, etc. Richter (2007) proposes a constraint to the effect that all \texttt{elist} objects are the same object, but the problem of the other spurious ambiguities remains.} This contrasts with the efforts
within HPSG (Pollard & Sag 1994; Pollard 1999; Richter 2007) to make models of various interpretations of utterances unique (at least up to isomorphism).

Now, it is possible to formulate within RSRL a constraint that makes sure that all indiscernible structures are in fact the same structure (Sailer 2003: Section 3.1.4), but such a constraint, if applied indeterminately, would be incompatible with various HPSG analyses – most importantly, with the standard HPSG binding theory (Pollard & Sag 1994: Chapter 6).45 There is no space to present that theory here (see Müller & Branco 2021 for an overview), but suffice it to say that the traditional generative notion of coindexation is understood here literally: as identity of index values. For example, the sentence (36) is assumed in HPSG to have two different structures corresponding to the following two indexations:

\[(38) \quad \begin{align*}
\text{a. } & \text{She}_i \text{ says she}_i \text{ loves you.} \\
\text{b. } & \text{She}_i \text{ says she}_j \text{ loves you.} \quad (i \neq j)
\end{align*}\]

So while any model of (38a) should equate index values within the two words she in this sentence, these index values must be different objects in any model of (38b), even though they are indiscernible.

To the best of my knowledge, the problem of avoiding spuriously distinct models in a way that does not conflict with existing HPSG analyses (in particular, with the standard binding theory) remains unsolved.

### 4.3.3 Conditions on models

Both theories impose meta-theoretical conditions on what counts as an intended model. As mentioned in Section 4.1, the common constraint on HPSG models is that they be exhaustive, i.e., informally speaking, simulate all other models: they should contain all structures admitted by the grammar. The intuition behind this requirement is that a single model corresponds to the whole language described by the grammar.

LFG apparently assumes the more common view of models, where each model corresponds to a single utterance, and it is only the collection of all such models that corresponds to the whole language. However, meta-theoretical conditions on LFG models are in a way more complex than conditions imposed on HPSG models.

First of all, LFG models are required to be minimal. For example, functional equations in the lexical entry of *she* (see (8)) involving the attribute \textit{INDEX}, i.e.

---

45 Also the architecture for phonology proposed in Höhle (1999) crucially relies on not all indiscernible structures being the same structure. Sailer (2003) formulates the relevant constraint in such a way that it only applies to one type of structures.
Figure 19: Partial HPSG representation of (36)

Figure 20: A fragment of a possible HPSG model of the AVM in Figure 19
equations repeated below in (39), describe as a possible value of INDEX not only the feature structure in (27), repeated below as (40), but also the one in (41) and infinitely many others, including infinite feature structures (both infinitely embedded and – on the assumption that the set of atoms may be infinite – with an infinite number of attributes).

(39)  
(↑ INDEX PERS) = 3  
(↑ INDEX NUM) = SG  
(↑ INDEX GEND) = F

\[
\begin{bmatrix}
\text{PERS} & 3 \\
\text{NUM} & \text{SG} \\
\text{GEND} & \text{F}
\end{bmatrix}
\]

(40)  
\[
\begin{bmatrix}
\text{PERS} & 3 \\
\text{NUM} & \text{SG} \\
\text{GEND} & \text{F} \\
\text{ARBI} & \text{TRARY} \\
\text{NON} & [\text{SEN} \text{ SE}]
\end{bmatrix}
\]

Other constraints in the grammar do not preclude such values of INDEX, so a meta-theoretical constraint is needed to the effect that only minimal feature structures satisfying the grammar are admitted within models. Technically, this amounts to defining a partial order on models and admitting only the minimal elements of this order.

The second condition on models is more complex and concerns constraining statements such as (42a) (from the grammar rule (4)) and (42b) (from the lexical entry (7)).

(42)  
a. (↓ TENSE)  
b. (↑ SUBJ INDEX PERS) = 3

Such statements are understood as additional filters on the minimal models of a grammar, or – more precisely – on the minimal models of the version of the grammar with all such constraining statements removed.

The precise model-theoretic nature of this mechanism has never, to the best of my knowledge, been specified. Constraining statements of this kind are not mentioned in the model-theoretic view of LFG of Kaplan (1995), and they are explicitly excluded in previous attempts to provide LFG (or LFG-like) formalisms with a model theory (see Johnson 1988: Section 4.2 and Blackburn & Gardent 1995: Section 6; see also Börjars & Payne 2013). But once meta-theoretical quantification over models and relations on models are permitted – and they are already
inherent both in the HPSG notion of exhaustive models and the LFG notion of minimal models – it is possible to understand constraining statements in model-theoretic terms. One possibility is this:

- Let $\theta$ be an LFG grammar, understood as a set of logical formulae. Some of the (sub)formulae are marked as constraining, the others are understood as defining.

- Let $\theta_{all}$ be the whole grammar $\theta$ without any division of (sub)formulae into defining and constraining, and $\theta_{def}$ – the same grammar with all constraining (sub)formulae removed.

- Let $M_{all}$ be the collection of all models of $\theta_{all}$, and $M_{def}$ – the collection of all minimal models of $\theta_{def}$.

- Then $M \equiv M_{def} \cap M_{all}$ is the collection of admitted models of $\theta$.

The idea here is that $M_{def}$ is the collection of all minimal models before the constraining filters are applied, and the intersection with $M_{all}$, i.e., with models in which all constraining statements are satisfied, removes from $M_{def}$ those models which do not satisfy some constraining statements.

4.4 Summary

This section, aiming to present and compare model theories assumed in HPSG and LFG, is more speculative than the previous sections. The reason is that one object of comparison exists and the other does not, so it was necessary to reconstruct a possible model theory of LFG from informal and very partial suggestions.

Perhaps surprisingly, it turns out that the idea that f-structures are sets of $\langle$attribute, value$\rangle$ pairs does not translate into elegant models, but rather creates an overhead of the need to represent these sets as objects within models. Also, additional care needs to be taken to ensure that co-extensional sets are really the same model objects. Moreover, it is not immediately clear how to formally

---

46Given that statements may contain disjunctions, and that different constraining statements may occur in different disjuncts, the actual definition would have to be more complex: grammars would have to be converted to a disjunctive normal form and collections of models would have to be defined for each disjunct of this normal form. Then the final collection of models of the grammar would be the sum of all such collections.

47Formally, minimal models are the minimal elements of the subsumption relation defined on models as in Johnson (1988: Section 2.8).
and non-procedurally ensure unique indexation of semantic forms. Nevertheless, despite these difficulties, and despite the fact that constraining statements were excluded from previous attempts to construct a model theory for LFG, it is not difficult to imagine how to construct such a model theory, if only appropriately powerful meta-theoretical operations on candidate models are permitted (as – to some extent – they already are, given the minimality requirement).

Also somewhat surprisingly, while much attention has been devoted to model theory within HPSG, there are still unsolved problems there, concerning the multiplicity of different models admitted by typical HPSG grammars, differing in ways that linguists often do not suspect, and certainly do not care about.

Let the conclusion of this section be that, despite their age and stability, both theories would benefit from more work on their formal foundations.

5 Conclusion

So how similar are LFG and HPSG? I agree with Carl Pollard that in some ways they are more similar than sometimes perceived:

I believe that the difference between LFG and so-called PSG [i.e., theories such as GPSG and HPSG; AP] is no greater than the differences among various theoretical proposals within PSG, or even within HPSG itself. As far as I am concerned, then, the separation between PSG and LFG exists more at a sociological level than at the level of scientific content – but I am aware that not everyone agrees about this. (Pollard 1997: 4)

In particular, the difference between the multi-level representations of LFG and the monolithic AVMs assumed in HPSG is – in my view – of little formal consequence, although it is certainly important for the compactness and readability of resulting structures.

In fact, LFG and HPSG converge in many respects. As emphasised above, both theories are highly formalised and – unlike derivational theories or Categorial Grammar – both are self-described as constraint-based or model-theoretic, although HPSG may boast of much more developed model theories. Importantly, both have well-developed computational platforms for implementing grammars: XLE (Crouch et al. 2008) in the case of LFG and LKB (Copestake 2002) and Trale (Carpenter et al. 2003) in the case of HPSG, with XLE allowing for very direct
implementations of theoretical analyses. In both cases, large-scale grammars of multiple languages have been developed.

Also, unlike some of the other highly formalised and implementable theories, both LFG and HPSG are empirically rich. A plethora of analyses of multiple phenomena in typologically varied languages have been offered within each theory, in a great many articles appearing in the best linguistic journals and in numerous monographs published by the most prominent publishers. Both have very well developed semantic components, and both make it possible to formulate precise analyses encompassing multiple linguistic levels. As emphasised in Wechsler & Asudeh (2021), many phenomena receive similar accounts in the two theories.

In summary, it is clear that LFG and HPSG are close neighbours in the linguistic theoretical landscape of the early 2020s, and it is my hope that this chapter encourages more neighbourly collaboration between the two theories.

Acknowledgements

Many thanks to the following people for comments on the first version of this chapter and – in some cases – for discussion: Doug Arnold, Ash Asudeh, Stefan Müller, Agnieszka Patejuk, Geoff Pullum, Frank Richter, Shuly Wintner, Annemie Zaenen, and the anonymous reviewers. They should not be held responsible for any controversial opinions expressed here or for any remaining inadequacies; if some of their suggestions are less than fully addressed, it is only because this is just a chapter (and one that is too long already!), and not the monograph that a fuller presentation and comparison of LFG and HPSG deserves. And many thanks to Mary Dalrymple, the editor of this volume, for her infinite patience!

References


More precisely, XLE makes it possible to faithfully implement syntactic and – with extensions described in Dalrymple et al. (2020) – semantic parts of LFG analyses. In the case of HPSG, Trale seems to be closer to its constraint-based nature, while LKB is more efficient and well-developed.
Adam Przepiórkowski


Adam Przepiórkowski


Przepiórkowski, Adam. 2016. How not to distinguish arguments from adjuncts in LFG. In Doug Arnold, Miriam Butt, Berthold Crysmann, Tracy Holloway King & Stefan Müller (eds.), *Proceedings of the joint 2016 conference on Head-Driven*
Richter, Frank & Manfred Sailer. 1999. LF conditions on expressions of Ty2: An HPSG analysis of negative concord in Polish. In Robert D. Borsley & Adam


