Chapter 32

HPSG and Lexical Functional Grammar

Stephen Wechsler
The University of Texas

Ash Asudeh
University of Rochester

Here is the abstract: concrete gets you through abstract better than abstract gets you through concrete

1 Semantics

HPSG was conceived from the start as a theory of the *sign* (de Saussure 1916), wherein each constituent is a pairing of form and meaning. So semantic representation and composition was built into HPSG (and the subsequent sister framework of Sign-Based Construction Grammar Boas & Sag 2012), as reflected in the title of the first HPSG book (Pollard & Sag 1987), *Information-Based Syntax and Semantics*. LFG was not founded as a theory that included semantics, but a semantic component was developed for LFG shortly thereafter (Halvorsen 1983). The direction of semantics for LFG changed some ten years later and the dominant tradition is now Glue Semantics (Dalrymple et al. 1993; Dalrymple 1999; 2001; Asudeh 2012; Dalrymple et al. 2019).

This section presents a basic introduction to Glue Semantics. Our goal is to present enough of the approach for readers to grasp the main intuitions behind it, without presupposing much knowledge of formal semantic theory. The references listed at the end of the previous paragraph (especially Dalrymple et al. (2019)) are good places to find additional discussion and references. The rest of this section is organized as follows. In Section 1.1 we present some more historical background on semantics for LFG and HPSG. In Section 1.2, we present

Stephen Wechsler & Ash Asudeh. HPSG and Lexical Functional Grammar. To appear in: Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.) HPSG Handbook Berlin: Language Science Press. [preliminary page numbering]

Glue Semantics (Glue), as a general compositional system in its own right. Then, in Section 1.3 we look at the syntax–semantics interface with specific reference to an LFG syntax. For further details on semantic composition and the syntax–semantics interface in constraint-based theories of syntax, see Koenig & Richter (2020), Chapter ?? of this volume for semantics for HPSG and Asudeh (2020) for semantics for LFG.

1.1 Brief history of semantics for LFG and HPSG

Various theories of semantic representation have been adopted by the different non-transformational syntactic frameworks over the years. The precursor to HPSG, GPSG, was paired by its designers with a then fairly standard static Montogovian semantics (Gazdar et al. 1985), but GPSG itself was subsequently adopted as the syntactic framework for Discourse Representation Theory (Kamp & Reyle 1993), a dynamic theory of semantics. Initial work on semantics for LFG also assumed a Montogovian semantics (Halvorsen 1983; Halvorsen & Kaplan 1988). But with the increasing interest in Situation Semantics (Barwise & Perry 1983) in the 1980s at Stanford University and environs (particularly SRI International and Xerox PARC), the sites of the foundational work on both HPSG and LFG, both frameworks also became associated with Situation Semantics. In the case of LFG, this association was not foundational, manifesting primarily in the form of Fenstad et al. (1987), and soon waned. In contrast, as noted above, HPSG is a theory of signs and therefore had semantics incorporated into the theory from the outset. Moreover, the chosen semantic theory was Situation Semantics, and this also carried over into the second main HPSG book (Pollard & Sag 1994) and further (Ginzburg & Sag 2000).

Beginning in the 90s, the focus subsequently shifted in new directions due to a new interest in computationally tractable theories of the syntax–semantics interface, to support efforts at large-scale grammar development, such as the ParGram project for LFG (Butt et al. 1999; 2002) and the LinGO/Grammar Matrix projects for HPSG (Flickinger 2000; Bender et al. 2002; 2010).[1] This naturally led to an interest in underspecified semantic representations, so that semantic

[1]Readers can explore the current incarnations of these projects at the following links (checked January 13, 2020):

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

LinGO http://lingo.stanford.edu

Grammar Matrix http://matrix.ling.washington.edu/index.html ambiguities such as scope ambiguity could be compactly encoded without the need for full enumeration of all scope possibilities. Two examples for HPSG are *Lexical Resource Semantics* (Richter 2004; Penn & Richter 2011) and *Minimal Recursion Semantics* (Copestake et al. 2005). Similarly, focus in semantics for LFG shifted to ways of encoding semantic ambiguity compactly and efficiently. This led to the development of Glue Semantics.

1.2 General Glue Semantics

In this section, we briefly review Glue Semantics itself, without reference to a particular syntactic framework. Glue Semantics is a general framework for semantic composition that requires *some* independent syntactic framework but does not presuppose anything about syntax except headedness, which is an uncontroversial assumption across frameworks. This makes the system flexible and adaptable, and it has been paired not just with LFG, but also with Lexicalized Tree-Adjoining Grammar (Frank & van Genabith 2001), HPSG (Asudeh & Crouch 2002b), Minimalism (Gotham 2018), and Universal Dependencies (Gotham & Haug 2018).

In Glue Semantics, meaningful linguistic expressions — including lexical items but also possibly particular syntactic configurations — are associated with *meaning constructors* of the following form:[2]

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

$\mathcal{M}$ is an expression from a *meaning language* which can be anything that supports the lambda calculus; G is an expression of *linear logic* (Girard 1987), which specifies the semantic composition (it "glues meanings together"), based on a syntactic parse. By convention a colon separates them. Glue Semantics is related to (Type-Logical) Categorial Grammar (Carpenter 1997; Morrill 1994; 2011; Moortgat 1997), but it assumes a separate syntactic representation for handling word order, so the terms of the linear logic specify just semantic composition without regard to word order (see Asudeh 2012 for further discussion). Glue Semantics is therefore useful in helping us focus on semantic composition in its own right.

The principal compositional rules for Glue Semantics are those for the linear implication connective, $\multimap$, which are here presented in a natural deduction format:

[2] It is in principle possible for a linguistic expression to have a phonology and syntax but not contribute to interpretation, such as expletive pronouns like *there* and *it* or the *do*-support auxiliary in English; see Asudeh (2012: 113) for some discussion of expletive pronouns in the context of Glue.

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

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

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

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

In each of these rules, the inference over the linear logic term, G, corresponds to an operation on the meaning term, via the Curry-Howard Isomorphism between formulas and types (Curry & Feys 1958; 1995; Howard 1980). The rule for eliminating the linear implication, which is just modus ponens, corresponds to functional application. The rule for introducing the linear implication, i.e. hypothetical reasoning, corresponds to functional abstraction. These rules will be seen in action shortly.

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

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

Let's also assume that s is mnemonic for the semantic correspondent of the (single word) phrase *Sam*, m similarly mnemonic for *Max*, and l for *likes*. In other words, the meaning constructor for *likes* would be associated with the lexical entry for the verb and specified in some general form such that it can be instantiated by the syntax (we will see an LFG example shortly); here we are assuming that the instantiation has given us the meaning constructor in (4).

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

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

As we will see below, the commutativity of the glue logic yields a simple and elegant treatment of quantifiers in non-subject positions, which are challenging for other frameworks (see, for example, the careful pedagogical presentation of the issue in Jacobson 2014: 244–263).

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

(6)
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\lambda y.\lambda x.f(y)(x) : a \multimap b \multimap c \qquad [v : a]^1}{\lambda x.f(v)(x) : b \multimap c}\multimap_{\mathcal{E}} \qquad [u : b]^2}{f(v)(u) : c}\multimap_{\mathcal{E}}}{\lambda v.f(v)(u) : a \multimap c}\multimap_{\mathcal{I},1}}{\lambda y.f(y)(u) : a \multimap c}\Rightarrow_{\alpha}}{\lambda u.\lambda y.f(y)(u) : b \multimap a \multimap c}\multimap_{\mathcal{I},2}}{\lambda x.\lambda y.f(y)(x) : b \multimap a \multimap c}\Rightarrow_{\alpha}$$

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

This presentation has been purposefully abstract to highlight what is intrinsic to the glue logic, but we of course need to see how this works with a syntactic framework to see how Glue Semantics actually handles semantic composition and the syntax–semantics interface. So next, in Section 1.3, we will review LFG+Glue, as this is the predominant pairing of syntactic framework and Glue.

1.3 Glue Semantics for LFG

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

(7) Blake called Alex.

(8) Blake called everybody.

(9) Everybody called somebody.

Example (7) is a simple case of a transitive verb with two proper name arguments, but is sufficient to demonstrate the basics of the syntax–semantics interface in LFG+Glue. Example (8) is a case of a quantifier in object position, which is challenging to compositionality because there is a type clash between the simplest type we can assign to the verb, $\langle e, \langle e, t\rangle\rangle$, and the simplest type that would be assigned to the quantifier, $\langle\langle e, t\rangle, t\rangle$. In other theories, this necessitates either a syntactic operation which is syntactically undermotivated, e.g. Quantifier Raising in interpretive theories of composition, such as Logical Form semantics (Heim & Kratzer 1998), or a type shifting operation of some kind in directly compositional approaches, as in categorial or type-logical frameworks; see Jacobson (2014) for further discussion and references. Example (9) also demonstrates this point, but it more importantly demonstrates that quantifier scope ambiguity can be handled in Glue without a) positing an undermotivated syntactic ambiguity and b) while maintaining the simplest types for both quantifiers.

The relevant aspects of the lexical entries involved are shown in Table 1. Other syntactic aspects of the lexical items, such as the fact that *called* has a SUBJECT and an OBJECT, are specified in its meaning constructor. Minimal f-structures are provided below for each example. The subscript σ indicates the semantic structure that corresponds to the annotated f-description. The types for the lexical items are the minimal types that would be expected. Note that in Glue these are normally associated directly with the semantic structures, for example $\uparrow_{\sigma_e}$ and $(\uparrow\textsc{obj})_{\sigma_e} \multimap (\uparrow\textsc{subj})_{\sigma_e} \multimap \uparrow_{\sigma_t}$, but they have been presented separately for better exposition; see Dalrymple et al. (2019: 299–305) for further discussion. We do not show semantic structures here, as they are not necessary for this simple demonstration.

The generalized quantifier functions associated with everybody and somebody are, respectively, **every** and **some** in the meaning language. The universal symbol $\forall$ in the glue logic/linear logic terms for the quantifiers ranges over semantic structures of type t. It is unrelated to the generalized quantifiers. Hence even the existential word *somebody* has the universal $\forall$ in its linear logic glue term. The $\forall$-terms thus effectively say that *any* type t semantic structure S that can be found by application of proof rules such that the quantifier's semantic structure implies S can serve as the scope of the quantifier; see Asudeh (2005: 393–394) for basic discussion of the interpretation of $\forall$ in linear logic. This will become clearer when quantifier scope is demonstrated shortly.

Let us assume the following f-structure for (7):

Table 1: Relevant lexical details for the three examples in (7–9)

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

(10) $c \begin{bmatrix} \textsc{pred} & \text{‘call’} \\ \textsc{subj} & b\begin{bmatrix} \textsc{pred} & \text{‘Blake’} \end{bmatrix} \\ \textsc{obj} & a\begin{bmatrix} \textsc{pred} & \text{‘Alex’} \end{bmatrix} \end{bmatrix}$

Note that here, unlike in previous sections, the PRED value for the verb does not list its subcategorization information. This is because we’ve made the standard move in much Glue work to suppress this information.[3] The f-structures are named mnemonically by the first character of their PRED value. All other f-structural information has been suppressed for simplicity. Based on these f-structure labels, the meaning constructors in the lexicon in Table 1 are instantiated as follows (σ subscripts suppressed):

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

These meaning constructors yield the following proof, which is the only available normal form proof for the sentence:[4]

[3] Indeed, one could go further and argue that PRED values do not list subcategorization at all, in which case the move is not just notational, and that the Principles of Completeness and Coherence instead follow from the resource-sensitivity of Glue Semantics; for some discussion, see Asudeh (2012); Asudeh & Giorgolo (2012); Asudeh et al. (2014).

[4] The reader can think of the normal form proof as the minimal proof that yields the conclusion, without unnecessary steps of introducing and discharging assumptions; see Asudeh & Crouch (2002a) for some basic discussion.

(12) **Proof**

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

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

Let us next assume the following f-structure for (8):

(13) $$c\begin{bmatrix} \textsc{pred} & \text{‘call’} \\ \textsc{subj} & b\begin{bmatrix} \textsc{pred} & \text{‘Blake’} \end{bmatrix} \\ \textsc{obj} & e\begin{bmatrix} \textsc{pred} & \text{‘everybody’} \end{bmatrix} \end{bmatrix}$$

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

(14) **Instantiated meaning constructors**

$\lambda y.\lambda x.\mathbf{call}(y)(x) : e \multimap b \multimap c$

$\lambda Q.\mathbf{every}(\mathbf{person}, Q) : \forall S.(e \multimap S) \multimap S$

$\mathbf{blake} : b$

These meaning constructors yield the following proof, which is again the only available normal form proof:[5]

(15) **Proof**

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

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

[5] We have not presented the proof rule for Universal Instantiation, but it is trivial; see Asudeh (2012: 396).

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

Lastly, let us assume the following f-structure for (9):

(16) $$c\begin{bmatrix} \textsc{pred} & \text{`call'} \\ \textsc{subj} & e\begin{bmatrix}\textsc{pred} & \text{`everybody'}\end{bmatrix} \\ \textsc{obj} & s\begin{bmatrix}\textsc{pred} & \text{`somebody'}\end{bmatrix} \end{bmatrix}$$

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

(17) **Instantiated meaning constructors**

$\lambda y.\lambda x.\mathbf{call}(y)(x) : s \multimap e \multimap c$

$\lambda Q.\mathbf{some}(\mathbf{person}, Q) : \forall S.(s \multimap S) \multimap S$

$\lambda Q.\mathbf{every}(\mathbf{person}, Q) : \forall S.(e \multimap S) \multimap S$

These meaning constructors yield the following proofs, which are the only available normal form proofs, but there are two distinct proofs, because of the scope ambiguity:[6]

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

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

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

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

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

The final meaning language expressions in (18) and (19) give the two possible readings for the scope ambiguity, again assuming a standard model theory with generalized quantifiers. Once more, notice that neither quantifier need be moved in the syntax — they are respectively just a SUBJECT and an OBJECT in f-structure. And once more, no special type shifting is necessary. It is a key strength of this approach that even quantifier scope ambiguity can be captured without positing *ad hoc* syntactic operations (and, again, without complicating the type of the object quantifier or the transitive verb). This is once more ultimately due to the commutativity of the linear logic.

Abbreviations

Acknowledgements

References

Asudeh, Ash. 2005. Relational nouns, pronouns, and resumption. *Linguistics and Philosophy* 28(4). 375–446.

Asudeh, Ash. 2012. *The logic of pronominal resumption*. Oxford: Oxford University Press.

Asudeh, Ash. 2020. Glue semantics. In Mary Dalrymple (ed.), *Lexical Functional Grammar: the handbook*. Berlin: Language Science Press.

Asudeh, Ash & Richard Crouch. 2002a. Derivational parallelism and ellipsis parallelism. In Line Mikkelsen & Christopher Potts (eds.), *Proceedings of the 21st West Coast Conference on Formal Linguistics*, 1–14. Somerville, MA: Cascadilla Press.

Asudeh, Ash & Richard Crouch. 2002b. Glue Semantics for HPSG. In Frank van Eynde, Lars Hellan & Dorothee Beermann (eds.), *Proceedings of the 8th International HPSG Conference*, 1–19. Stanford, CA: CSLI Publications.

Asudeh, Ash & Gianluca Giorgolo. 2012. Flexible composition for optional and derived arguments. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG12 conference*, 64–84. Stanford, CA: CSLI Publications.

Asudeh, Ash, Gianluca Giorgolo & Ida Toivonen. 2014. Meaning and valency. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG14 conference*. Stanford, CA: CSLI Publications.

Barwise, Jon & John Perry. 1983. *Situations and attitudes*. Cambridge, MA: MIT Press.

Bender, Emily M., Scott Drellishak, Antske Fokkens, Laurie Poulson & Safiyyah Saleem. 2010. Grammar customization. *Research on Language & Computation* 8(1). 23–72.

Bender, Emily M., Dan Flickinger & Stephan Oepen. 2002. The grammar matrix: an open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In John Carroll, Nelleke Oostdijk & Richard Sutcliffe (eds.), *Proceedings of the workshop on grammar engineering and evaluation at the 19th international conference on computational linguistics*, 8–14. Taipei, Taiwan.

Boas, Hans C. & Ivan A. Sag (eds.). 2012. *Sign-based Construction Grammar* (CSLI Lecture Notes 193). Stanford, CA: CSLI Publications.

Butt, Miriam, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi & Christian Rohrer. 2002. The Parallel Grammar Project. In *Proceedings of coling-2002 workshop on grammar engineering and evaluation*, 1–7.

Butt, Miriam, Tracy Holloway King, María-Eugenia Niño & Frédérique Segond. 1999. *A grammar writer's cookbook*. Stanford, CA: CSLI Publications.

Carpenter, Bob. 1997. *Type-logical semantics*. Cambridge, MA: MIT Press.

Copestake, Ann, Dan Flickinger, Carl Pollard & Ivan A. Sag. 2005. Minimal recursion semantics: an introduction. *Research on Language and Computation* 3(4). 281–332.

Curry, Haskell B. & Robert Feys. 1995. The basic theory of functionality. analogies with propositional algebra. In Philippe de Groote (ed.), *The Curry-Howard isomorphism*, vol. 8 (Cahiers du Centre de Logique), 9–13. Louvain-la-neuve, Belgium: Academia.

de Groote, Philippe (ed.). 1995. *The Curry-Howard isomorphism*. Vol. 8 (Cahiers du Centre de Logique). Louvain-la-neuve, Belgium: Academia.

Curry, Haskell B. & Robert Feys. 1958. *Combinatory logic*. Vol. 1. Amsterdam: North-Holland.

Dalrymple, Mary (ed.). 1999. *Semantics and syntax in Lexical Functional Grammar: the resource logic approach*. Cambridge, MA: MIT Press.

Dalrymple, Mary. 2001. *Lexical Functional Grammar* (Syntax and Semantics 34). New York, NY: Academic Press.

Dalrymple, Mary, John Lamping & Vijay Saraswat. 1993. LFG semantics via constraints. In *Proceedings of the Sixth Meeting of the European ACL*, 97–105. University of Utrecht.

Dalrymple, Mary, John J. Lowe & Louise Mycock. 2019. *The Oxford Reference Guide to Lexical Functional Grammar*. Oxford: Oxford University Press.

de Saussure, Ferdinand. 1916. *Cours de linguistique générale* (Bibliothèque Scientifique Payot). Paris: Payot.

Fenstad, Jens Erik, Per-Kristian Halvorsen, Tore Langhold & Johan van Benthem. 1987. *Situations, language and logic*. Dordrecht: D. Reidel.

Flickinger, Dan. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering* 6(1). 15–28.

Frank, Anette & Josef van Genabith. 2001. LL-based semantics construction for LTAG — and what it teaches us about the relation between LFG and LTAG. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG01 conference*, 104–126. Stanford, CA: CSLI Publications.

Gazdar, Gerald, Ewan Klein, Geoffrey K. Pullum & Ivan A. Sag. 1985. *Generalized Phrase Structure Grammar*. Cambridge, MA: Harvard University Press.

Ginzburg, Jonathan & Ivan A. Sag. 2000. *Interrogative investigations: the form, meaning and use of English interrogatives*. Stanford, CA: CSLI Publications.

Girard, Jean-Yves. 1987. Linear logic. *Theoretical Computer Science* 50(1). 1–102.

Gotham, Matthew. 2018. Making logical form type-logical: glue semantics for minimalist syntax. *Linguistics and Philosophy*.

Gotham, Matthew & Dag Trygve Truslew Haug. 2018. Glue semantics for Universal Dependencies. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG11 conference*, 208–226. Stanford, CA: CSLI Publications.

Halvorsen, Per-Kristian. 1983. Semantics for Lexical-Functional Grammar. *Linguistic Inquiry* 14(4). 567–615.

Heim, Irene & Angelika Kratzer. 1998. *Semantics in generative grammar*. Oxford: Blackwell.

Howard, William A. 1980. The formulae-as-types notion of construction. In Jonathan P. Seldin & J. Roger Hindley (eds.), *To H.B. Curry: essays on combinatory logic, lambda calculus and formalism*, 479–490. London: Academic press.

Jacobson, Pauline. 2014. *Compositional semantics: An introduction to the syntax/semantics interface*. Oxford: Oxford University Press.

Kamp, Hans & Uwe Reyle. 1993. *From discourse to logic: introduction to modeltheoretic semantics of natural language, formal logic and Discourse Representation Theory*. Dordrecht: Kluwer.

Koenig, Jean-Pierre & Frank Richter. 2020. Semantics. In Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.), *Head-Driven Phrase Structure Grammar: The handbook*, 893–932. Berlin: Language Science Press.

Halvorsen, Per-Kristian & Ronald M. Kaplan. 1988. Projections and semantic description in Lexical-Functional Grammar. In *Proceedings of the international conference on fifth generation computer systems*, 1116–1122. Tokyo.

Moortgat, Michael. 1997. Categorial type logics. In Johan van Benthem & Alice ter Meulen (eds.), *Handbook of logic and language*, 1st edn., 93–177. Amsterdam: North-Holland.

Morrill, Glyn V. 1994. *Type Logical Grammar*. Dordrecht: Kluwer.

Morrill, Glyn V. 2011. *Categorial grammar: logical syntax, semantics, and processing*. Oxford: Oxford University Press.

Penn, Gerald & Frank Richter. 2011. Lexical resource semantics: from theory to implementation. In Stefan Müller (ed.), *Proceedings of the 18th international conference on Head-Driven Phrase Structure Grammar*, 423–443. Stanford, CA: CSLI Publications.

Pollard, Carl J. & Ivan A. Sag. 1987. *Information-based syntax and semantics*. CSLI Publications.

Pollard, Carl J. & Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar* (Studies in Contemporary Linguistics). Chicago: The University of Chicago Press.

Richter, Frank. 2004. *Foundations of Lexical Resource Semantics*.