Some casual observations argue against modular language processing. For example, the famously ambiguous sentences (2a) and (2b) can be disambiguated in speech by the stress patterns.

(2) a. I forgot how good beer tastes.
b. Dogs must be carried.

The two meanings of (2a) correspond to two different parses (one with *good* as part of the noun phrase *good beer* and the other with *how good* as a verb phrase modifier). The two meanings of (2b) have the same syntactic structure, but differ in whether the requirement is that all dogs be carried, or that everyone carry a dog. This interaction of prosody with syntax (in the case of (2a)) and with semantics (in the case of (2b)) is produced and perceived before the end of the utterance, suggesting that phonological information is available in the course of syntactic and semantic processing.

Moreover, non-linguistic knowledge influences the disambiguation in both of these cases. If (2a) is preceded by "I just finished three weeks without alcohol", the natural interpretation of *good* is as a modifier of *tastes*; but following "I just finished three weeks drinking only Bud Light", *good* is more naturally interpreted as a modifier of *beer*. In the case of (2b), only one interpretation (that anyone with a dog must carry it) is plausible, given our knowledge of the world. Indeed, most non-linguists fail to see the ambiguity of (2b) without a lengthy explanation.

More rigorous evidence of the non-modular character of language processing has been provided by a variety of types of experiments. The work of Michael Tanenhaus and his associates, using eye-tracking to investigate the time-course of sentence comprehension, played an important role in convincing most psycholinguists that human language understanding is non-modular. See, for example, Eberhard et al. (1995), McMurray et al. (2008), Tanenhaus, Spivey-Knowlton, Eberhard, et al. (1995), Tanenhaus et al. (1996), and Tanenhaus & Trueswell (1995). A recent survey of work arguing against modularity in language processing is provided by Spevack et al. (2018).

2.3 Importance of words

The individual properties of words play a central role in how people process phrases and sentences. Consider, for example, what is probably the most famous sentence in psycholinguistics, (3), due originally to (Bever 1970).

(3) The horse raced past the barn fell.

The extreme difficulty that people who have not previously been exposed to (3) have comprehending it depends heavily on the choice of words. A sentence like (4), with the same syntactic structure, is far easier to parse.

(4) The applicant interviewed in the morning left.

Numerous studies (e.g. Ford et al. 1982; Trueswell et al. 1993; MacDonald et al. 1994; Bresnan et al. 2007; Wasow et al. 2011) have shown that such properties of individual words as subcategorization preferences, semantic categories (e.g. animacy), and frequency of use can influence the processing of utterances.

2.4 Influence of context

Much of the evidence against modularity of the language faculty is based on the influences of non-linguistic context and world knowledge on language processing. The well-known McGurk effect (McGurk & MacDonald 1976) and the Stroop effect (Stroop 1935) demonstrate that, even at the word level, visual context can influence linguistic comprehension and production.

Linguistic context also clearly influences processing, as the discussion of examples (2a) and (2b) above illustrates. The same conclusion is supported by numerous controlled studies, including, among many others, those described by Crain & Steedman (1985), Altmann & Steedman (1988), Branigan (2007), Traxler & Tooley (2007), Matsuki et al. (2011), and Spevack et al. (2018). The last of these references concludes (p. 11), "when humans and their brains are processing language with each other, there is no format of linguistic information (e.g., lexical, syntactic, semantic, and pragmatic) that cannot be rapidly influenced by context."

2.5 Speed and accuracy of processing

A good deal of psycholinguistic literature is devoted to exploring situations in which language processing encounters difficulties, notably work on garden paths (in comprehension) and disfluencies (in production). Much more striking than the existence of these phenomena, however, is how little they matter in everyday language use. While ambiguities abound in normal sentences (see Wasow 2015), comprehenders very rarely experience noticeable garden paths. Similarly, disfluencies in spontaneous speech occur in nearly every sentence but rarely disrupt communication.

People are able to use speech to exchange information remarkably efficiently. A successful account of human language processing must explain why it works as well as it does.

3 Features of HPSG that fit well with processing facts

In this section, I review some basic design features of HPSG, pointing out ways in which they comport well with the properties of language processing listed in the previous section.

3.1 Constraint-based

Well-formedness of HPSG representations is defined by the simultaneous satisfaction of a set of constraints that constitutes the grammar. This lack of directionality allows the same grammar to be used in modeling production and comprehension.

check for crossreferencing other chapters once they are in

Consider, for instance, the example of quirky case assignment illustrated in (1) above. A speaker uttering (1) would need to have planned to use the verb *helfen* before beginning to utter the object NP. But a listener hearing (1) would encounter the dative case on the article *dem* before hearing the verb and could infer only that a verb taking a dative object was likely to occur at the end of the clause. Hence, the partial mental representations built up by the two interlocutors during the course of the utterance would be quite different. But the grammatical mechanism licensing the combination of a dative object with this particular verb is the same for speaker and hearer.

In contrast, theories of grammar that utilize sequential operations to derive sentences impose a directionality on their grammars. If such a grammar is then to be employed as a component in a model of language use (as the competence hypothesis stipulates), its inherent directionality becomes part of the models of both production and comprehension. But production involves mapping meaning onto sound, whereas comprehension involves the reverse mapping. Hence, a directional grammar cannot fit the direction of processing for both production and comprehension.[6]

Branigan & Pickering (2017) argue at length that "structural priming provides an implicit method of investigating linguistic representations".[7] They go on to conclude (p. 14) that the evidence from priming supports "frameworks that …

[6] This was an issue for early work in computational linguistics that built parsers based on the transformational grammars of the time, which generated sentences using derivations whose direction went from an underlying structure largely motivated by semantic considerations to the observable surface structure. See, for example, Hobbs & Grishman (1975).

[7] Priming is the tendency for speakers to re-use linguistic elements that occurred earlier in the context; structural priming (which Branigan & Pickering sometimes call *abstract priming*) is priming of linguistic structures, abstracted from the particular lexical items in those structures.

assume nondirectional and constraint-based generative capacities (i.e., specifying well-formed structures) that do not involve movement".[8] HPSG is one of the frameworks they mention that fit this description.

3.2 Surface-oriented

The features and values in HPSG representations are motivated by straightforwardly observable linguistic phenomena. HPSG does not posit derivations of observable properties from abstract underlying structures. In this sense it is surface-oriented.

The evidence linguists use in formulating grammars consists of certain types of performance data, primarily judgments of acceptability and meaning. Accounts of the data necessarily involve some combination of grammatical and processing mechanisms. The closer the grammatical descriptions are to the observable phenomena, the less complex the processing component of the account needs to be.

For example, the grammatical theory of Kayne (1994), which posits a universal underlying order of specifier-head-complement, requires elaborate (and directional) transformational derivations to relate these underlying structures to the observable data in languages whose surface order is different (a majority of the language of the world). In the absence of experimental evidence that the production and comprehension of sentences with different constituent orders involve mental operations corresponding to the grammatical derivations Kayne posits, his theory of grammar seems to be incompatible with the competence hypothesis.

Experimental evidence supports this reasoning. As Branigan & Pickering (2017) conclude (p. 9), "[P]riming evidence supports the existence of abstract syntactic representations. It also suggests that these are shallow and monostratal in a way that corresponds at least roughly to the assumptions of … Pollard & Sag (1994) …. It does not support a second, underlying level of syntactic structure or the syntactic representation of empty categories associated with the movement of constituents in some transformational analyses."

3.3 Informationally rich representations

The feature structure descriptions of HPSG include all types of linguistic information relevant to the well-formedness and interpretation of expressions. This

[8] Branigan & Pickering's conclusions are controversial, as is evident from the commentaries accompanying their target article.

includes phonological, morphological, syntactic, semantic, and contextual information. They can also incorporate non-linguistic contextual information (e.g. social information), though this has not been extensively explored.

The local cooccurrence of these different types of information within a single representation facilitates modeling production and comprehension processes that make reference to more than one of them. The architecture of the grammar is thus well suited to the non-modularity and context-sensitivity of language processing. It is interesting in this regard to consider the conclusions of two papers by psycholinguists who surveyed experimental evidence and inferred what types of grammatical information was essential for processing.

The following series of quotes captures the essence of what MacDonald et al. (1994) wrote regarding lexical representation, based on a survey of a wide range of psycholinguistic studies:

- "[T]he lexical representation for a word includes a representation of the word's phonological form, orthographic form, semantics, grammatical features (including grammatical category), morphology (at least inflectional), argument structure, and X-bar structure." (p. 684)
- "[T]he connection structure of the lexicon encodes relationships among different types of lexical information."[9] (p. 684)
- "In addition to constraints that hold between various aspects of lexical representations, sentence and discourse contexts also constrain lexical representations during processing…" (p. 686)

With the possible exception of "X-bar structure", this sounds very much like a description of the types of information included in HPSG feature structure descriptions.

Over twenty years later, Branigan & Pickering (2017) came to the following conclusions about linguistic representations, based on priming studies:

- "The syntactic representations capture local relationships between a 'mother' and its constituent 'daughter(s)' (e.g., a VP comprising a verb and two NPs), independent of the larger context in which the phrase appears (e.g., that the VP occurs within a subordinate clause), or the internal structure of the

[9] A reviewer asked what feature of HPSG this maps into. The answer is straightforward: a word's phonological form, semantics, grammatical features, morphology, and argument structure are all represented together in one feature structure description, and the different pieces of the description may be linked through coindexing or tagging.

subphrases that constitute it (e.g., that the first NP comprises a determiner, adjective, and noun)." (p. 9)

- "[S]ome elements that are not phonologically represented may be syntactically represented." (p. 10)
- "Other priming evidence similarly indicates that some semantically specified elements are not specified syntactically." (p. 11)
- "[T]he semantic level of representation contains at least specifications of quantificational information, information structure, and thematic roles." (p. 11)
- "Evidence from priming supports a range of mappings between information encoded in the semantic representation and information encoded in the syntactic representation: between thematic roles and grammatical functions, between thematic roles and word order, between animacy and syntactic structure, and between event structures and syntactic structures." (p. 12)

The two lists are quite different. This is in part because the focus of the earlier paper was on lexical representations, whereas the later paper was on linguistic representations more generally. It may also be attributable to the fact that MacDonald et al. framed their paper around the issue of ambiguity resolution, while Branigan & Pickering's paper concentrated on what could be learned from structural priming studies. Despite these differences, it is striking that the conclusions of both papers about the mental representations employed in language processing are very much like those arrived at by work in HPSG.

3.4 Lexicalism

A great deal of the information used in licensing sentences in HPSG is stored in the lexical entries for words. A hierarchy of lexical types permits commonalities to be factored out to minimize what has to be stipulated in individual entries, but the information in the types gets into the representations of phrases and sentences through the words that instantiate those types. Hence, it is largely the information coming from the words that determines the well-formedness of larger expressions. Any lexical decomposition would have to be strongly motivated by the morphology.

Branigan & Pickering (2017) note that grammatical structures (what some might call *constructions*) such as V-NP-NP can prime the use of the same abstract structure, even in the absence of lexical overlap. But they also note that the priming is consistently significantly stronger when the two instances share the same verb, a fact known as *the lexical boost*. They write, "To explain abstract priming, lexicalist theories must assume that the syntactic representations […] are shared across lexical entries." The types in HPSG's lexicon provide just such representations. Branigan & Pickering go on to say that the lexical boost argues for "a representation that encodes a binding between constituent structure and the lemma … of the lexical entry for the head." In HPSG, this "binding" is simply the fact that the word providing the lexical boost (say, *give*) is an instantiation of a type specifying the structures it appears in (e.g. the ditransitive verb type).

Similarly, the fact, noted in Section 2.3 above, that a given structure may be more or less difficult to process depending on word choice is unsurprising in HPSG, so long as the processor has access to information about individual words and not just their types.

3.5 Underspecification

HPSG allows a class of linguistic structures that share some feature values to be characterized by means of feature structure descriptions that specify only the features whose values are shared. Such underspecification is very useful for a model of processing (particularly a model of the comprehender) because it allows partial descriptions of the utterance to be built up, based on the information that has been encountered. This property of the grammar makes it easy to incorporate into an incremental processing model.

4 Two phenomena of interest

4.1 Island constraints

Ever since Ross's seminal dissertation (1967) introduced the notion of "island constraints", linguists have sought explanations for their existence, often suggesting that they were motivated by processing considerations (notably Grosu 1972; Fodor 1983a; Deane 1991). The basic idea is that island constraints restrict the search space the parser needs to consider in looking for a gap to match a filler it has encountered, thereby facilitating processing. This then raises the question of whether island constraints need to be represented in grammar (language particular or universal), or can be attributed entirely to processing and/or

other factors, such as pragmatics.

In principle, this question is orthogonal to the choice among theories of grammar. But in recent years, a controversy has arisen between some proponents of HPSG and certain transformational grammarians, with the former (e.g. Chaves 2012 and 2019, Chapter 16 of this volume; Hofmeister & Sag 2010; Hofmeister, Jaeger, Arnon, Sag & Snider 2013) arguing that certain island phenomena should be attributed entirely to extra-grammatical factors, and the latter (e.g. Phillips 2013 and Sprouse et al. 2012) arguing that island constraints are part of grammar.

I will not try to settle this dispute here. Rather, my point in this subsection is to note that a theory in which there is a close fit between the grammar and processing mechanisms allows for the possibility that some island phenomena should be attributed to grammatical constraints, whereas others should be explained in terms of processing. Indeed, if the basic idea that islands facilitate processing is correct, it is possible that some languages, but not others, have grammaticalized some islands, but not others. That is, in a theory in which the grammar is a tightly integrated component of a processing model, the question of whether a particular island phenomenon is due to a grammatical constraint is an empirical one whose answer might differ from language to language.

Early work on islands (e.g. Ross 1967 and Chomsky 1973) assumed that, in the absence of negative evidence, island constraints could not be learned and hence must be innate and therefore universal. But cross-linguistic variation in island constraints, even between closely related languages, has been noted since the early days of research on the topic (see e.g. Erteschik-Shir 1973 and Engdahl & Ejerhed 1982).

This situation is what one might expect if languages differ with respect to the extent to which the processing factors that motivate islandhood have been grammaticalized. In short, a theory with a tight fit between its grammatical machinery and its processing mechanisms allows for hybrid accounts of islands that are not available to theories without such a fit.

One example of such a hybrid is Chaves's (2012) account of Ross's Coordinate Structure Constraint. Following much earlier work, Chaves distinguishes between the "conjunct constraint", which prohibits a gap from serving as a conjunct in a coordinate structure (as in **What did you eat a sandwich and?*) and the "element constraint", which prohibits a gap from serving as an element of a larger conjunct (as in **What did you eat a sandwich and a slice of?*). The conjunct constraint, he argues, follows from the architecture of HPSG and is therefore built into the grammar. The element constraint, on the other hand, has exceptions and,

he claims, should be attributed to extra-grammatical factors. See Chaves (2019), Chapter 16 of this volume for a more detailed discussion of islands.

4.2 Subject vs. object relative clauses

One of the most discussed phenomena in the literature on human sentence processing is the difference in processing complexity between relative clauses (RCs) in which the gap is the subject and those in which the gap is the object – or, as they are commonly called, "subject RCs" and "object RCs"; see, among many others, Wanner & Maratsos (1978), Gibson (1998), Traxler et al. (2002), and Gennari & MacDonald (2008). Relative clause processing complexity has been shown to be influenced by a number of factors other than the grammatical function of the gap, including the animacy and pronominality of the overt NP in the RC, as well as the frequency, animacy, and discourse properties of the head of the RC.[10] When these factors are controlled for, however, most psycholinguists accept that it has been established that subject RCs are generally easier to process than object RCs, at least in English.[11]

One approach to explaining this asymmetry has been based on the distance between the filler and the gap (see, among others, Wanner & Maratsos 1978; Gibson 1998; Hawkins 2004). In languages like English, with basic SVO clause order and RCs that follow the nouns they modify, the distance between the filler (the relativizer or head noun) and the gap is greater for an object gap than for a subject gap. If holding a filler in memory until the gap is encountered puts an extra burden on the processor, this could explain why object RCs are harder to process than subject RCs. This distance-based account makes an interesting prediction

[10] The stimuli in the experimental studies on this topic always have RCs with one overt NP, either in subject or object position and a gap corresponding to the other grammatical function. In most of the studies, that NP is non-pronominal and animate. See Reali & Christiansen (2007) and Roland et al. (2012) for evidence of the role of these factors in processing complexity.

[11] This processing difference corresponds to the top end of the "accessibility hierarchy" that Keenan & Comrie (1977) proposed as a linguistic universal. Based on a diverse sample of 50 languages, they proposed the hierarchy below, and hypothesized that any language allowing RC gaps at any point in the hierarchy would allow RC gaps at all points higher (to the left) on the hierarchy.

Subject > Direct Object > Indirect Object > Oblique > Genitive > Object of Comparison

Keenan & Comrie speculated that the generality of this hierarchy of relativizability lay in processing, specifically on the comprehension side. The extensive experimental evidence that has been adduced in support of this idea in the intervening decades has been concentrated on subject RCs vs. (direct) object RCs. The remainder of the hierarchy remains largely untested by psycholinguists.

for languages with different word orders. In languages like Japanese with SOV order and RCs that precede the nouns they modify, the distance relationships are reversed – that is, the gaps in object RCs are closer to their fillers than those in subject RCs. The same is true of Chinese, with basic SVO order and RCs that precede the nouns they modify. So the prediction of distance-based accounts of the subject/object RC processing asymmetry is that it should be reversed in these languages.

The experimental evidence on this prediction is somewhat equivocal. While Hsiao & Gibson (2003) found a processing preference for object RCs over subject RCs in Chinese, their findings were challenged by Lin & Bever (2006) and Vasishth et al. (2013), who claimed that Chinese has a processing preference for subject RCs. In Japanese, Miyamoto & Nakamura (2003) found that subject RCs were processed more easily than object RCs. The issue remains controversial, but, for the most part, the evidence has not supported the idea that the processing preference between subject RCs and object RCs varies across languages with different word orders.

The most comprehensive treatment of English RCs in HPSG is Sag (1997). Based entirely on distributional evidence, Sag's analysis treats (finite) subject RCs as fundamentally different from RCs whose gap does not function as the subject of the RC. The difference is that the SLASH feature, which encodes information about long-distance dependencies in HPSG, plays no role in the analysis of subject RCs. Non-subject RCs, on the other hand, involve a non-empty SLASH value in the RC.[12]

Sag deals with a wide variety of kinds of RCs. From the perspective of the processing literature, the two crucial kinds are exemplified by (5a) and (5b), from Gibson (1998).

(5) a. The reporter who attacked the Senator admitted the error.
    b. The reporter who the Senator attacked admitted the error.

A well-controlled experiment on the processing complexity of subject and object RCs must have stimuli that are matched in every respect except the role of the gap in the RC. Thus, the conclusion that object RCs are harder to process than subject RCs is based on a wide variety of studies using stimuli like (5). Sag's analysis of (5a) posits an empty SLASH value in the RC, whereas his analysis of (5b) posits a non-empty SLASH value.

[12] The idea that at least some subject gaps differ in this fundamental way from non-subject gaps goes back to Gazdar (1981).

There is considerable experimental evidence supporting the idea that unbounded dependencies – that is, what HPSG encodes with the SLASH feature – add to processing complexity; see, for example, Wanner & Maratsos (1978), King & Just (1991), Kluender & Kutas (1993), and Hawkins (1999). Combined with Sag's HPSG analysis of English RCs, this provides an explanation of the processing preference of subject RCs over object RCs. On such an account, the question of which other languages will exhibit the same preference boils down to the question of which other languages have the same difference in the grammar of subject and object RCs. At least for English, this is a particularly clear case in which the architecture of HPSG fits well with processing evidence.

5 Conclusion

This chapter opened with the observation that HPSG has not served as the theoretical framework for much psycholinguistic research. The observations in Sections 2 through 4 argue for rectifying that situation. The fit between the architecture of HPSG and what is known about human sentence processing suggests that HPSG could be used to make processing predictions that could be tested in the lab.

To take one example, the explanation of the processing asymmetry between subject and object RCs offered above is based on a grammatical difference in the HPSG analysis: all else being equal, expressions with non-empty SLASH values are harder to process than those with empty SLASH values. Psycholinguists could test this idea by looking for other cases of phenomena that look superficially very similar but whose HPSG analyses differ with respect to whether SLASH is empty. One such case occurs with pairs like Chomsky's famous minimal pair in (6).

(6) a. Chris is eager to please.
    b. Chris is easy to please.

Under the analysis of Pollard & Sag (1994), *to please* in (6b) has a non-empty SLASH value but an empty SLASH value in (6a). Processing (6a) should therefore be easier. This prediction could be tested experimentally, and modern methods such as eye-tracking could pinpoint the locus of any difference in processing complexity to determine whether it corresponds to the region where the grammatical analysis involves a difference in SLASH values.

The current disconnect between theoretical investigations of language structure and psycholinguistic studies is an unfortunate feature of our discipline. Because HPSG comports so well with what is known about processing, it could

serve as the basis for a reconnection between these two areas of study.

Acknowledgements

A number of people made valuable suggestions on a preliminary outline and an earlier draft of this chapter, leading to improvements in content and presentation, as well as the inclusion of previously overlooked references. In particular, I received helpful comments from (in alphabetical order): Emily Bender, Bob Borsley, Rui Chaves, Danièle Godard, Jean-Pierre Koenig, and Stefan Müller. Grateful as I am for their advice, I take sole responsibility for any shortcomings in the chapter.

References

Altmann, Gerry & Yuki Kamide. 1999. Incremental interpretation at verbs: Restricting the domain of subsequent refrence. *Cognition* 73(7). 247–264.

Altmann, Gerry & Mark Steedman. 1988. Interaction with context during human sentence processing. *Cognition* 30(3). 191–238.

Arnold, Jennifer E., Carla Hudson Kam & Michael K. Tanenhaus. 2007. If you say thee uh- you're describing something hard: The on-line attribution of disfluency during reference comprehension. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 33. 914–930.

Bever, Thomas. 1970. The cognitive basis for linguistic structures. In John R. Hayes (ed.), *Cognition and the language development*, 279–362. New York: Wiley & Sons.

Branigan, Holly. 2007. Syntactic priming. *Language and Linguistics Compass* 1(1–2). 1–16.

Branigan, Holly & Martin Pickering. 2017. An experimental approach to linguistic representation. *Behavioral and Brain Sciences* 40. 1–61.

Bresnan, Joan, Anna Cueni, Tatjana Nikitina & Harald Baayen. 2007. Predicting the dative alternation. In Gerlof Bouma, Irene Krämer & Joost Zwarts (eds.), *Cognitive foundations of interpretation*, 69–94. Amsterdam: KNAW.

Bresnan, Joan, Shipra Dingare & Christopher D. Manning. 2001. Soft constraints mirror hard constraints: Voice and person in English and Lummi. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG 01 Conference*. Standford, CA: CSLI.

Chaves, Rui. 2019. Island phenomena and related matters. In Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.), *Head-Driven Phrase Structure Grammar*, i–xxxvi. Berlin: Language Science Press. DOI:??

Chaves, Rui P. 2012. On the grammar of extraction and coordination. *Natural Language and Linguistic Theory* 30(2). 465–512.

Chomsky, Noam. 1965. *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.

Chomsky, Noam. 1973. Conditions on transformations. In Stephen R. Anderson & Paul Kiparsky (eds.), *A festschrift for Morris Halle*, 232–286. New York: Holt, Rinehart & Winston.

Crain, Stephen & Mark J. Steedman. 1985. On not being led up the garden path: The use of context by the psychological syntax processor. In David R. Dowty, Lauri Karttunen & Arnold M. Zwicky (eds.), *Natural language processing*, 320–358. Cambridge, UK: Cambridge University Press.

Deane, Paul D. 1991. Limits to attention: A cognitive theory. *Cognitive Linguistics* 2(1). 1–63.

Eberhard, Kathleen M., Michael J. Spivey-Knowlton, Julie C. Sedivy & Michael K. Tannenhaus. 1995. Eye movements as a window into real-time spoken language comprehension in natural contexts. *Journal of Psycholinguist Research* 24. 409–436.

Engdahl, Elisabeth & Eva Ejerhed. 1982. *Readings on unbounded dependencies in Scandinavian languages*. Stockholm: Almqvist & Wiksell International.

Erteschik-Shir, Nomi. 1973. *On the nature of island constraints*. Cambridge, MA: MIT dissertation.

Flickinger, Dan, Carl Pollard & Tom Wasow. 2019. The evolution of HPSG. In Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.), *Head-Driven Phrase Structure Grammar*, i–xxiv. Berlin: Language Science Press. DOI:??

Fodor, Janet Dean. 1983a. Phrase structure parsing and the island constraints. *Linguisitcs and Philosophy* 6. 163–223.

Fodor, Jerry A. 1983b. *The modularity of mind: An essay on faculty psychology*. Cambridge, MA: MIT Press.

Fodor, Jerry A., Thomas G. Bever & Merrill F. Garrett. 1974. *The psychology of language: An introduction to psycholinguistics and Generative Grammar*. New York: McGraw-Hill Book Co.

Ford, Marilyn, Joan Bresnan & Ronald Kaplan. 1982. A competence-based theory of syntactic closure. In Joan Bresnan (ed.), *The mental representation of grammatical relations*, 727–796. Cambridge, MA: MIT Press.

Francis, Elaine. 2019. *Gradient acceptability and linguistic theory*. Ms. Purdue University, Under review.

Gazdar, Gerald. 1981. Unbounded dependencies and coordinate structure. *Linguistic Inquiry* 12. 155–184.
Gennari, Silvia P. & Maryellen C. MacDonald. 2008. Semantic indeterminacy in object relative clauses. *Journal of Memory and Language* 8. 161–187.
Gibson, Edward. 1998. Linguistic complexity: Locality of syntactic dependencies. *Cognition* 68(1). 1–76.
Gollan, Tamar H., Timothy J. Slattery, Diane Goldenberg, Eva Van Assche, Wouter Duyck & Keith Rayner. 2011. Frequency drives lexical access in reading but not in speaking: The frequency-lag hypothesis. *Journal of Experimental Psychology: General* 140(2). 186–209.
Grosu, A. 1972. *The strategic content of island constraints* (Working Papers in Linguistics 13). Ohio Sate University.
Hawkins, John A. 1999. Processing complexity and filler-gap dependencies across grammars. *Language* 75(2). 244–285.
Hawkins, John A. 2004. *Efficiency and complexity in grammars*. Oxford: Oxford University Press.
Hawkins, John A. 2014. *Cross-linguistic variation and efficiency*. Oxford: Oxford University Press.
Hobbs, Jerry R. & Ralph Grishman. 1975. The automatic transformational analysis of English sentences: An implementation. *International Journal of Computer Mathematics* 5(1–4). 267–283.
Hofmeister, Philip, T. Florian Jaeger, Inbal Arnon, Ivan A. Sag & Neal Snider. 2013. The source ambiguity problem: Distinguishing the effects of grammar and processing on acceptability judgments. *Language and Cognitive Processes* 28(1–2). 48–87.
Hofmeister, Philip & Ivan A. Sag. 2010. Cognitive constraints and island effects. *Language* 86(2).
Hsiao, Franny & Edward Gibson. 2003. Processing relative clauses in chinese. *Cognition* 90. 3–27.
Kayne, Richard S. 1994. *The antisymmetry of syntax* (Linguistic Inquiry Monographs 25). Cambridge, MA: MIT Press.
Keenan, Edward L. & Bernard Comrie. 1977. Noun phrase accessibility and Universal Grammar. *Linguistic Inquiry* 8(1). 63–99.
King, Jonathan & Marcel Just. 1991. Individual differences in syntactic processing: The role of working memory. *Journal of Memory and Language* 30. 580–602.
Kluender, Robert & Marta Kutas. 1993. Bridging the gap: Evidence from ERPs on the processing of unbounded dependencies. *Journal of Cognitive Neuroscience* 5(2). 196–214.

Konieczny, Lars. 1996. *Human sentence processing: A semantics-oriented parsing approach*. Universität Freiburg Dissertation. IIG-Berichte 3/96.

Lin, Chien-Jer Charles & Thomas Bever. 2006. Subject preference in the processing of relative clauses in Chinese. In Donald Baumer, David Montero & Michael Scanlon (eds.), *Proceedings of the 25th West Coast Conference on Formal Linguistics*, 254–260. Somerville, MA: Cascadilla Proceedings Project.

Linadarki, Evita. 2006. *Linguistic and statistical extensions of data oriented parsing*. University of Essex dissertation. Unpublished.

MacDonald, Maryellen C. 2013. How language production shapes language form and comprehension. *Frontiers in Psychology* 4(226). 1–16.

MacDonald, Maryellen C., Neal J. Pearlmutter & Mark. S. Seidenberg. 1994. The lexical nature of syntactic ambiguity resolution. *Psychological Review* 101(4). 676–703.

Marslen-Wilson, William & Lorraine Tyler. 1987. Against modularity. In Jay L. Garfield (ed.), *Modularity in knowledge representation and Natural Language Processing*, 37–62. Cambridge, MA: MIT Press.

Matsuki, Kazunaga, Tracy Chow, Mary Hare, Christoph Elman Jeffrey L. Scheepers & Ken McRae. 2011. Event-based plausibility immediately influences online language comprehension. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 37(4). 913–934.

McGurk, Harry & John MacDonald. 1976. Hearing lips and seeing voices. *Nature* 264(5588). 746–748.

McMurray, Bob, Meghan A. Clayards, Michael K. Tanenhaus & Richard Aslin. 2008. Tracking the time course of phonetic cue integration during spoken word recognition. *Psychonomic Bulletin* 15(6). 1064–1071.

Miyamoto, Edson T & Michiko Nakamura. 2003. *Subject/object asymmetries in the processing of relative clauses in Japanese*. Gina Garding & Mimu Tsujimura (eds.). Somerville, MA.

Miyao, Yusuke & Jun'ichi Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics* 34(1). 35–80.

Momma, Shota & Colin Phillips. 2018. The relationship between parsing and generation. *Annual Review of Linguistics* 4. 233–254.

Phillips, Colin. 2013. On the nature of island constraints I: Language processing ad reductionist accounts. In Jon Sprouse & Norbert Hornstein (eds.), *Experimental syntax and island effects*, 64–108. Cambridge: Cambridge University Press.

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

Reali, Florencia & Morten H. Christiansen. 2007. Processing of relative clauses is made easier by frequency of occurrence. *Journal of Memory and Language* 57(1). 1–23.

Roland, Douglas, Gail Mauner, Carolyn O'Meara & Hongoak Yun. 2012. Discourse expectations and relative clause processing. *Joural of Memory and Language* 66(3). 470–508.

Ross, John R. 1967. *Constraints on variables in syntax*. MIT, Cambridge, Massachusetts. [Published in 1986 as *Infinite Syntax!* Norwood, NJ: Ablex Publishing] Ph.D. Dissertation.

Sag, Ivan A. 1997. English relative clause constructions. *Journal of Linguistics* 33(2). 431–484.

Sag, Ivan A. & Thomas Wasow. 2011. Performance-compatible competence grammar. In Robert D. Borsley & Kersti Börjars (eds.), *Non-transformational syntax: Formal and explicit models of grammar: A guide to current models*, 359–377. Oxford, UK/Cambridge, MA: Blackwell Publishers Ltd.

Sag, Ivan A. & Thomas Wasow. 2015. Flexible processing and the design of grammar. *Journal of Psycholinguistic Research* 44(1). 47–63.

Sag, Ivan A., Thomas Wasow & Emily M. Bender. 2003. *Syntactic theory: A formal introduction*. 2nd edn. (CSLI Lecture Notes 152). Stanford, CA: CSLI Publications.

Spevack, Samuel C., J. Benjamin Falandays & Michael J. Spivey. 2018. Interactivity of language. *Language and Linguistics Compass* 12(7). e12282.

Sprouse, Jon, Matt Wagers & Colin Phillips. 2012. A test of the relation between working memory capacity and syntactic island effects. *Language* 88(1). 82–123.

Stroop, John Ridley. 1935. Studies of interference in serial verbal reactions. *Journal of Experimental Psychology* 18(6). 643–662.

Tanenhaus, Michael K., Michael J. Spivey-Knowlton, Kathleen M. Eberhard & Julie C. Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science* 268(5217). 1632–1634. DOI:10.1126/science.7777863

Tanenhaus, Michael K., Michael J. Spivey-Knowlton, Kathleen M. Eberhard & Julie C. Sedivy. 1996. Using eye movements to study spoken language comprehension: Evidence for visually mediated incremental interpretation. In Toshio Inui & James L. McClelland (eds.), *Information integration in perception and communication* (Attention and Performance XVI), 457–478. Cambridge, MA: MIT Press.

Tanenhaus, Michael K., Michael J. Spivey-Knowlton & Kathleen M. Eberhard & Julie C. Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science* 268(5217). 1632–1634.

Tanenhaus, Michael K. & John C. Trueswell. 1995. Sentence comprehension. In Joanne L. Miller & Peter D. Eimas (eds.), *Handbook of cognition and perception*. San Diego, CA: Academic Press.

Traxler, Matthew J., Robin K. Morris & Rachel E. Seely. 2002. Processing subject and object relative clauses: Evidence from eye movements. *Journal of Memory and Language* 47(1). 69–90.

Traxler, Matthew J. & Kristen M. Tooley. 2007. Lexical mediation and context effects in sentence processing. *Brain Research* 1146. 59–74.

Trueswell, John C., Micahel K. Tanenhaus & Christopher Kello. 1993. Verb-specific constraints in sentence processing: Separating effects of lexical preference from garden-paths. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 19(3). 528–553.

Vasishth, Shravan, Chen Zhong, Qiang Li & Gueilan Guo. 2013. Processing Chinese relative clauses: Evidence for the subject-relative advantage. *PLoS ONE* 8(10). e77006.

Wanner, Eric & Michael Maratsos. 1978. An ATN approach to comprehension. In Morris Halle, Joan Bresnan & George A. Miller (eds.), *Linguistic theory and psychological reality*, 119–161. Cambridge, MA: MIT Press.

Wasow, Thomas. 2015. Ambiguity avoidance is overrated. In Susanne Winkler (ed.), *Ambiguity: Language and communication*, 29–47. Berlin: De Gruyter.

Wasow, Thomas, Florian T. Jaeger & David Orr. 2011. Lexical variation in relativizer frequency. In Horst J. Simon & Heike Wiese (eds.), *Expecting the unexpected: Exceptions in grammar*, 175–195. Berlin: De Gruyter.

Chapter 28

Computational linguistics and grammar engineering

Emily M. Bender
University of Washington

Guy Emerson
University of Cambridge

We discuss the relevance of HPSG for computational linguistics, and the relevance of computational linguistics for HPSG.

1 Introduction

From the inception of HPSG in the 1980s, there has been a close integration between theoretical and computational work (for an overview, see **chapters/chap-evolution** Chapter **??** of this volume). In this chapter, we discuss computational work in HPSG, starting with the infrastructure that supports it (both theoretical and practical) in Section 2. Next we describe several existing large-scale projects which build HPSG or HPSG-inspired grammars (see Section 3) and the deployment of such grammars in applications including both those within linguistic research and otherwise (see Section 4). Finally, we turn to linguistic insights gleaned from broad-coverage grammar development (see Section 5).

2 Infrastructure

2.1 Theoretical considerations

There are several properties of HPSG as a theory that make it well-suited to computational implementation. First, the theory is kept separate from the formalism:

Emily M. Bender & Guy Emerson. N.d. Computational linguistics and grammar engineering. In Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.), *Head-Driven Phrase Structure Grammar.* prepublished version. Berlin: Language Science Press. [Preliminary page numbering]

the formalism is expressive enough to encode a wide variety of possible theories. While some theoretical work does argue for or against the necessity of particular formal devices (e.g., the shuffle operator (Reape 1994)), much of it proceeds within shared assumptions about the formalism. This is in contrast to work in the context of the Minimalist Program (Chomsky 1995), where theoretical results are typically couched in terms of modifications to the formalism itself. From a computational point of view, the benefit of differentiating between theory and formalism is that the formalism is relatively stable. That enables the development and maintenance of software systems that target the formalism (Boguraev et al. 1988), such as software for parsing, generation, and grammar exploration (see Section 3 below for some examples).[1]

A second important property of HPSG that supports a strong connection between theoretical and computational work is an interest in both so-called "core" and so-called "peripheral" phenomena. Most implemented grammars are built with the goal of handling naturally occurring text.[2] This means that they will need to handle a wide variety of linguistic phenomena not always treated in theoretical syntactic work (Baldwin et al. 2005). A syntactic framework that excludes research on "peripheral" phenomena as uninteresting provides less support for implementational work than does one, like HPSG or Construction Grammar, that values such topics (for a comparison of HPSG and Construction Grammar, see Müller 2019c, Chapter 36 of this volume).

Finally, the type hierarchy characteristic of HPSG lends itself well to developing broad-coverage grammars which are maintainable over time (see Sygal & Wintner 2011). The use of the type hierarchy to manage complexity at scale comes out of the work of Flickinger (1987) and others at in the project where HPSG was originally developed. The core idea is that any given constraint is (ideally) expressed only once on types which serve as supertypes to all entities that bear that constraint.[3] Such constraints might represent broad generalizations that apply to many entities or relatively narrow, idiosyncratic properties that apply to

[1] There are implementations of Minimalism, notably Stabler 1997 and Herring 2016. (See also Torr et al. 2019 for a recent broad-coverage, treebank-trained parser in this framework.) However, implementing a theory requires fixing the formalism, and so these implementations are unlikely to be useful for testing theoretical ideas as the theory moves on.

[2] It is possible, but less common, to do implementation work strictly against test suites of sentences constructed specifically to focus on phenomena of interest.

[3] Originally this only applied to lexical entries in Flickinger's work. Now it also applies to phrase structure rules, lexical rules, and types below the level of the sign which are used in the definition of all of these. See **chapters/chap-evolution** Chapter **??** of this volume for further discussion.

only a few. By isolating any given constraint on one type (as opposed to repeating it in multiple places), we build grammars that are easier to update and adapt in light of new data that require refinements to constraints. Having a single locus for each constraint also makes the types a very useful target for documentation (Hashimoto et al. 2008) and grammar exploration (Letcher 2018).

2.2 Practical considerations

HPSG allows practical implementations because it uses a well-defined formalism. Furthermore, because HPSG is defined to be bi-directional, an implemented grammar can be used for both parsing and generation. In this section, we discuss how HPSG allows tractable algorithms, which enables linguists to empirically test hypotheses and which also enables HPSG grammars to be used in a range of applications, as we will see in Sections 4.1 and 4.2, respectively.

2.2.1 Computational complexity

One way to measure how easy or difficult it is to use a syntactic theory in practical computational applications is to consider the *computational complexity*[4] of parsing and generation algorithms (Gazdar & Pullum 1985). Computational complexity includes both how much memory and how much computational time a parsing algorithm needs to process a particular sentence.[5] Considering parsing time, longer sentences will take longer to process, but the more complex the algorithm is, the more quickly the amount of processing time increases. Parsing complexity can thus be measured by considering sentences containing n tokens, and then increasing n to see how the amount of time changes. This can be done based on the average amount of time for sentences in a corpus (average-case complexity), or based on the longest amount of time for all theoretically possible sentences (worst-case complexity).

At first sight, analyzing computational complexity would seem to paint HPSG in a bad light, because the formalism allows us to write grammars which can be arbitrarily complex; in technical terminology, the formalism is *Turing-complete*

[4] Computational complexity is related to the complexity hierarchy of language classes in formal language theory. More complex language classes tend to require parsing and generation algorithms with higher computational complexity, but this relationship is not exact. For example, the class of strictly local languages is a proper subset of the class of regular languages, but both classes can be parsed in linear time. Müller (2019b: ch. 17) discusses HPSG from the point of view of formal language theory.

[5] In this section, we only consider parsing algorithms, but a similar analysis can be done for generation (e.g., Carroll et al. 1999).

(Johnson 1988: Section 3.4). However, as discussed in the previous section, there is a clear distinction between theory and formalism. Although the HPSG formalism rules out the possibility of efficient algorithms that could cope with any possible feature-structure grammar, a particular theory (or a particular grammar) might well allow efficient algorithms.

Keeping processing complexity manageable is handled differently in other computationally-friendly frameworks, such as Combinatory Categorial Grammar (CCG),[6] or Tree Adjoining Grammar (TAG; Joshi 1987; Schabes et al. 1988). The formalisms of CCG and TAG inherently limit computational complexity: for both of them, as the sentence length n increases, worst-case parsing time is proportional to n^6 (Kasami et al. 1989). This is a deliberate feature of these formalisms, which aim to be just expressive enough to capture human language, and not any more expressive. Building this kind of constraint into the formalism itself highlights a different school of thought from HPSG. Indeed, Müller (2015: 64) explicitly argues in favor of developing linguistic analyses first, and improving processing efficiency second. As discussed above in Section 2.1, separating the formalism from the theory means that the formalism is stable, even as the theory develops.

It would be beyond the scope of this chapter to give a full review of parsing algorithms, but it is instructive to give an example. For grammars that have a context-free backbone (every analysis can be expressed as a phrase-structure tree plus constraints between mother and daughter nodes), it is possible to adapt the standard *chart-parsing* algorithm Kay (1973) for context-free grammars. The basic idea is to parse "bottom-up", starting by finding analyses for each token in the input, and then finding analyses for increasingly longer sequences of tokens (called *spans*, until the parser reaches the entire sentence.

For a context-free grammar, there is a finite number of nonterminal symbols, and each span is analyzed as a subset of the nonterminals. For a feature-structure grammar, each span must be analyzed as a set of feature structures,[7] which makes the algorithm more complicated. In principle, a grammar may allow an infinite number of possible feature structures, for example if it includes recursive unary

[6] For an introduction, see Steedman & Baldridge (2011). For a comparison with HPSG, see Kubota 2019, Chapter 33 of this volume.

[7] Much theoretical work in HPSG, including Pollard & Sag (1994), distinguishes between fully resolved feature structures and possibly underspecified feature structure descriptions. Much computational work, by contrast, operates entirely with partially specified feature structures, at both the level of grammar and the level of analyses licensed by the grammar. In keeping with this tradition, we use the term "feature structure" to refer to both fully specified and partially specified objects, and have no need for the term "feature structure description".

rules. However, if we can bound the number of possible feature structures as C, then the worst-case parsing time is proportional to $C^2 n^{\rho+1}$, where ρ is the maximum number of children in a phrase-structure rule (Carroll 1993: Section 3.2.3). This is less complex than for an arbitrary grammar (which means that this class of grammars is *not* Turing-complete), but C may nonetheless be very large.

But is the number of possible feature structures bounded in implemented HPSG grammars? For DELPH-IN grammars (see Section 3.2), the answer is yes. Assuming a system without relational constraints, the potential for unboundedness in the number of feature structures stems from the potential for recursion in feature paths: A listslist is a simple example,[8] and as another example, the elements on a COMPS list also include the feature COMPS.

However, in practice, such recursive paths do not need to be considered by the parsing algorithm. For example, selecting heads might place constraints on their complements' subjects (e.g., in raising/control constructions),raisingcontrol but no further than that (e.g., a complement's complement's subject). Similarly, while lists that are potentially unbounded in length are used in semantic representations, these are never involved in constraining grammaticality. The only lists that constrain grammaticality are valence lists, but in practical grammars these are never greater than length four or five.[9]

When parsing real corpora, it turns out that the average-case complexitycomplexity!average-case is much better than might be expected (Carroll 1994). On the one hand, grammatical constructions do not generally combine in the worst-case way, and on the other hand, when a grammar writer is confronted with multiple possible analyses for a particular construction, they may opt for the analysis that is more efficient for a particular parsing algorithm (Flickinger 2000). To measure the efficiency of grammars and parsing algorithms in practice, it can be helpful to use a test suite composed of a representative sample of sentences (Oepen & Flickinger 1998).

[8] More precisely, in the standard implementation of a list as a feature structure, the type *list* has two subtypes *null* and *non-empty-list*, and *non-empty-list* has the features FIRST and REST, where the value of REST is of type *list*. The value of REST can itself have the feature REST.

[9] In part, this is because DELPH-IN does not adopt proposals like the DEPS list of Bouma, Malouf & Sag (2001). Furthermore, in many DELPH-IN grammars, including the ERG, the SLASH list cannot have more than one element. If an unbounded SLASH list is required (such as to model cross-serial dependencies), the number of possible structures might still be bounded as a function of sentence length; this would allow us to bound worst-case parsing complexity, but it will be a higher bound.

2.2.2 Parse ranking

Various kinds of ambiguity are well-known in linguistics (such as modifier attachment and part-of-speech assignment), to the point that examples like (1) are stock in trade:

(1) a. I saw the kid with the telescope.
    b. Visiting relatives can be annoying.

A well-constructed grammar should be expected to return multiple parses for each ambiguous sentence.

However, people are naturally very good at resolving ambiguity, which means most ambiguity is not apparent, even to linguists. It is only with the development of large-scale grammars that the sheer scale of ambiguity has become clear. For example, (2) might seem unambiguous, but there is a second reading, where *my favorite* is the topicalized object of *speak*, which would mean that town criers generally speak the speaker's favorite thing (perhaps a language) clearly. There is also a third, even more implausible reading, where *my favorite town* is the topicalized object. Such implausible readings don't easily come to mind, and in fact, the 2018 version of the English Resource Grammar (ERG; Flickinger 2000; 2011) gives a total of 21 readings for this sentence. With increasingly long sentences, such ambiguities stack up very quickly. For (3), the first line of a newspaper article,[10] the ERG gives 35,094 readings.

(2) My favorite town criers speak clearly.

(3) A small piece of bone found in a cave in Siberia has been identified as the remnant of a child whose mother was a Neanderthal and father was a Denisovan, a mysterious human ancestor that lived in the region.

While exploring ambiguity can be interesting for a linguist, typical practical applications require just one parse per input sentence and specifically the parse that best reflects the intended meaning (or only the top few parses, in case the one put forward as "best" might be wrong). Thus, what is required is a *ranking* of the parses, so that the application can only use the most highly-ranked parse, or the top N parses.

[10] https://www.theguardian.com/science/2018/aug/22/offspring-of-neanderthal-and-denisovan-identified-for-first-time, accessed 16 August 2019

Parse ranking is not usually determined by the grammar itself, because of the difficulty of manually writing disambiguation rules.[11] Typically, a statistical system is used (Toutanova et al. 2002; 2005). First, a corpus is *treebanked*: for each sentence in the corpus, an annotator (often the grammar writer) chooses the best parse, out of all parses produced by the grammar. The set of all parses for a sentence is often referred to as the *parse forest*, and the selected best parse is often referred to as the *gold standard* or *gold parse*. Given the gold parses for the whole corpus, a statistical system is trained to predict the gold parse from a parse forest, based on many features[12] of the parse. From the example in (2), a number of different features all influence the preferred interpretation: the likelihood of a construction (such as topicalization), the likelihood of a valence frame (such as transitive *speak*), the likelihood of a collocation (such as *town crier*), the likelihood of a semantic relation (such as speaking a town), and so on.

Because of the large number of possible parses, it can be helpful to *prune* the search space: rather than ranking the full set of parses, ranking is restricted to a smaller set of parses. Carefully choosing how to restrict the parser's attention can drastically reduce processing time without hurting parsing accuracy, as long as the algorithm for selecting the subset includes the correct parse sufficiently frequently. One method, called *supertagging*,[13] exploits the fact that HPSG is a lexicalized theory: choosing the correct lexical entry brings in rich information that can be exploited to rule out many possible parses. Thus if the correct lexical entry can be chosen prior to parsing (e.g., on the basis of the prior and following words), the range of possible analyses the parser must consider is drastically reduced. Although there is a chance that the supertagger will predict the wrong lexical entry, using a supertagger can often improve parsing accuracy, by ruling out parses that the parse-ranking model might incorrectly rank too high. Supertagging was first applied to HPSG by Matsuzaki et al. (2007), building on previous work for TAG (Bangalore & Joshi 1999) and CCG (Clark & Curran 2004). To allow multi-word expressions (such as *by and large*), where the grammar assigns

[11] In fact, in earlier work, this task was undertaken by hand. One of the authors (Bender) had the job of maintaining rule weights in addition to developing the Jacy grammar (Siegel, Bender & Bond 2016) at YY Technologies in 2001–2002. No systematic methodology for determining appropriate weights was available and the system was both extremely brittle (sensitive to any changes in the grammar) and next to impossible to maintain.

[12] In the machine-learning sense of "feature", not the feature-structure sense.

[13] The term *supertagging*, due to Bangalore & Joshi (1999), refers to *part-of-speech tagging*, which predicts a part-of-speech for each input token, from a relatively small set of part-of-speech tags. Supertagging is "super", in that it predicts detailed lexical entries, rather than simple parts of speech.

a single lexical entry to multiple tokens, Dridan (2013) proposes an extension of supertagging, called *ubertagging*, which jointly predicts both a segmentation of the input and supertags for those segments. Dridan manages to increase parsing speed by a factor of four, while also improving parsing accuracy.

Finally, in order to train these statistical systems, we need to first annotate a treebank. When there are many parses for a sentence, it can be time-consuming to select the best parse. To efficiently use an annotator's time, it can be helpful to use *discriminants*, properties which hold for some parses but not for others (Carter 1997). For example, discriminants might include whether to analyze an ambiguous token as a noun or a verb, or where to attach a prepositional phrase. This approach to treebanking also means that annotations can be re-used when the grammar is updated (Oepen et al. 2004; Flickinger et al. 2017). For more on treebanking, see Section 4.1.4.

2.2.3 Semantic dependencies

In practical applications of HPSG grammars, the full phrase-structure trees and the full feature structures are often unwieldy, containing far more information than necessary for the task at hand. It is therefore often desirable to extract a concise semantic representation.

In computational linguistics, a popular approach to semantics is to represent the meaning of a sentence as a *dependency graph*, as this enables the use of graph-based algorithms.[14] Several types of dependency graph have been proposed based on Minimal Recursion Semantics (MRS; Copestake et al. 2005), with varying levels of simplification. Oepen & Lønning (2006) observe that if every predicate has a unique *intrinsic argument*, an MRS can be converted to a variable-free semantic representation, by replacing each reference to a variable with a reference to the corresponding predicate. They present Elementary Dependency Structures (EDS), semantic graphs which maintain predicate-argument structure but discard some scope information. (For many applications, scope information is less important than predicate-argument structure.) Copestake (2009) builds on this idea to create a more expressive graph-based representation called Dependency Minimal Recursion Semantics (DMRS), which is fully interconvertible with MRS.[15] This expressivity is achieved by adding annotations on the edges to

[14] In this section, we are concerned with *semantic* dependencies. For *syntactic* dependencies, see Hudson 2019, Chapter 35 of this volume. Some practical applications of HPSG use syntactic dependencies (including many applications of the Alpino grammar, discussed in Section 3.3.1).

[15] More precisely, for DMRS and MRS to be fully interconvertible, every predicate (except for

indicate scope information. Finally, DELPH-IN MRS Dependencies (DM; Ivanova et al. 2012) express predicate-argument structure purely in terms of the surface tokens, without introducing any abstract predicates.

For example, the English Resource Grammar (ERG) produces the MRS representation in (4) for the sentence *The cherry tree blossomed.* For simplicity, we have omitted some details, including features such as number and tense, individual constraints (ICONS), and the use of difference lists. By convention, DELPH-IN predicates beginning with an underscore correspond to a lexical item, and have a three-part format, consisting of a lemma, a part-of-speech tag, and (optionally) a sense. Predicates without an initial underscore are abstract predicates. The *qeq* constraints (equality modulo quantifiers) are scopal relationships, where quantifiers may possibly intervene (for details, see Copestake et al. 2005 or Koenig & Richter 2019, Chapter 23 of this volume).

(4)

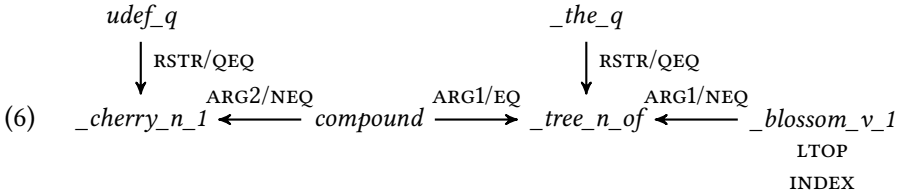
$$\begin{bmatrix} \textit{mrs} \\ \textsc{hook} & \begin{bmatrix} \textit{hook} \\ \textsc{ltop} & \boxed{1} \\ \textsc{index} & \boxed{2} \end{bmatrix} \\ \textsc{rels} & \left\langle \begin{bmatrix} \textit{relation} \\ \textsc{pred} & \textit{\_the\_q} \\ \textsc{lbl} & \boxed{3} \\ \textsc{arg0} & \boxed{4} \\ \textsc{rstr} & \boxed{5} \\ \textsc{body} & \boxed{6} \end{bmatrix}, \begin{bmatrix} \textit{relation} \\ \textsc{pred} & \textit{compound} \\ \textsc{lbl} & \boxed{7} \\ \textsc{arg0} & \boxed{8} \\ \textsc{arg1} & \boxed{4} \\ \textsc{arg2} & \boxed{9} \end{bmatrix}, \begin{bmatrix} \textit{relation} \\ \textsc{pred} & \textit{udef\_q} \\ \textsc{lbl} & \boxed{10} \\ \textsc{arg0} & \boxed{9} \\ \textsc{rstr} & \boxed{11} \\ \textsc{body} & \boxed{12} \end{bmatrix}, \right. \\ & \left. \begin{bmatrix} \textit{relation} \\ \textsc{pred} & \textit{\_cherry\_n\_1} \\ \textsc{lbl} & \boxed{13} \\ \textsc{arg0} & \boxed{9} \end{bmatrix}, \begin{bmatrix} \textit{relation} \\ \textsc{pred} & \textit{\_tree\_n\_of} \\ \textsc{lbl} & \boxed{7} \\ \textsc{arg0} & \boxed{4} \end{bmatrix}, \begin{bmatrix} \textit{relation} \\ \textsc{pred} & \textit{\_blossom\_v\_1} \\ \textsc{lbl} & \boxed{1} \\ \textsc{arg0} & \boxed{2} \\ \textsc{arg1} & \boxed{4} \end{bmatrix} \right\rangle \\ \textsc{hcons} & \left\langle \begin{bmatrix} \textit{qeq} \\ \textsc{harg} & \boxed{5} \\ \textsc{larg} & \boxed{7} \end{bmatrix}, \begin{bmatrix} \textit{qeq} \\ \textsc{harg} & \boxed{11} \\ \textsc{larg} & \boxed{13} \end{bmatrix} \right\rangle \end{bmatrix}$$

For readability, it can be easier to express an MRS in a more abstract mathematical form, as shown in (5). This is equivalent to the feature structure in (4).

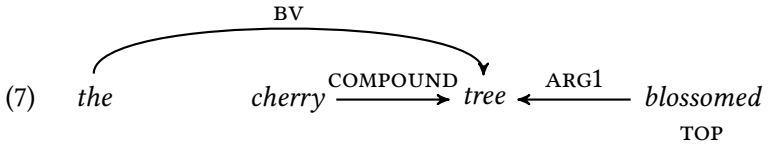
quantifiers) must have an intrinsic argument, and every variable must be the intrinsic argument of exactly one predicate.

(5)
$$\begin{aligned}
&\textsc{index}: e_1 \\
&\quad l_1: _the_q(x_1, h_1, h_2), h_1 \textsc{ qeq } l_4 \\
&\quad l_2: udef_q(x_2, h_3, h_4), h_3 \textsc{ qeq } l_3 \\
&\quad l_3: _cherry_n_1(x_2) \\
&\quad l_4: _tree_n_of(x_1), compound(e_2, x_1, x_2) \\
&\textsc{ltop}, l_5: _blossom_v_1(e_1, x_1)
\end{aligned}$$

The corresponding Dependency Minimal Recursion Semantics (DMRS) representation is shown in (6). This captures all of the information in the MRS in (5). Predicates are represented as nodes, while semantic roles and scopal constraints are represented as directed edges, called *dependencies* or *links*. Each dependency has two labels. The first is an argument label, such as ARG1, ARG2, or RSTR (the restriction of a quantifier). The second is a scopal constraint, such as QEQ[16], EQ (the linked nodes share a label in the MRS, which is generally true for modifiers), or NEQ (the linked nodes don't share a label).

(6)


Finally, the corresponding DELPH-IN MRS Dependencies (DM) representation is shown in (7). This is a simplified version of MRS, where all nodes are tokens in the sentence. Some abstract predicates are dropped (such as *udef_q*), while others are converted to dependencies (such as *compound*). Some scopal information is dropped (such as EQ vs NEQ). The label BV stands for the "bound variable" of a quantifier, equivalent to the RSTR/QEQ of DMRS.

(7)


The existence of such dependency graph formalisms, as well as software packages to manipulate such graphs (e.g., Ivanova et al. 2012, Copestake et al. 2016, Hershcovich et al. 2019, or PyDelphin[17]), has made it easier to use HPSG grammars in a number of practical tasks, as we will discuss in Section 4.2.

[16] An alternative notation is to write /H instead of /QEQ.

[17] https://github.com/delph-in/pydelphin/, accessed 16 August 2019

3 Development of HPSG resources

In this section we describe various projects that have developed computational resources on the basis of or inspired by HPSG. As we'll discuss in Section 4 below, such resources can be used both in linguistic hypothesis testing as well as in various practical applications. The intended purpose of the resources influences the form that they take. The CoreGram Project (Section 3.1) and Babel (Section 3.3.3) primarily target linguistic hypothesis testing, the Alpino and Enju parsers (Section 3.3.1 and 3.3.2) primarily target practical applications, and the DELPH-IN Consortium (Section 3.2) attempts to balance these two goals.

3.1 CoreGram

The CoreGram[18] Project aims to produce large-scale HPSG grammars, which share a common "core" grammar (Müller 2015). At the time of writing, large grammars have been produced for German (Müller 2007), Danish (Müller & Ørsnes 2013), Persian (Müller & Ghayoomi 2010), Maltese (Müller 2009), and Mandarin (Müller & Lipenkova 2013). Smaller grammars are also available for English, Yiddish, Spanish, French, and Hindi.

All grammars are implemented in the TRALE system (Meurers et al. 2002; Penn 2004), which accommodates a wide range of technical devices proposed in the literature, including: phonologically empty elements, relational constraints, implications with complex antecedents, and cyclic feature structures. It also accomodates macros and an expressive morphological component. Melnik (2007) observes that, compared to other platforms like the LKB (see Section 3.2 below), this allows grammar engineers to directly implement a wider range of theoretical proposals.

An important part of CoreGram is the sharing of grammatical constraints across grammars. Some general constraints hold for all grammars, while others hold for a subset of the grammars, and some only hold for a single grammar. Müller (2015) describes this as a "bottom-up approach with cheating" (p. 43)—the aim is to analyze each language on its own terms (hence "bottom-up"), but to re-use analyses from existing grammars if possible (hence "with cheating"). The use of a core set of constraints is motivated not just for practical reasons, but also for theoretical ones. By developing multiple grammars in parallel, analyses can be improved by cross-linguistic comparison. The constraints encoded in the core grammar can be seen as an hypothesis about the structure of human language.

[18] https://hpsg.hu-berlin.de/Projects/CoreGram.html

as we will discuss in Section 4.1.1.

CoreGram grammar development aims to incrementally increase coverage of each language. To measure progress, grammars are evaluated against test suites, collections of sentences each annotated with a grammaticality judgment (Oepen et al. 1997; Müller 2004b). This allows a grammarian to check for unexpected side effects when modifying a grammar, to avoid situations when implementing an analysis of one phenomenon would break the analysis of another phenomenon. This is particularly important when modifying a constraint that is used by several grammars. To help achieve these aims, grammar development is supported by a range of software tools, including the test suite tool [incr tsdb()] (Oepen 2001; see also Section 3.2), and the graphical debugging tool Kahina (Dellert et al. 2010; 2013).

3.2 The DELPH-IN Consortium

The DELPH-IN[19] Consortium was established in 2001 to facilitate the development of large-scale, linguistically motivated HPSG grammars for multiple languages in tandem with the software required for developing them and deploying them in practical applications. At the time that DELPH-IN was founded, the English Resource Grammar (ERG; Flickinger 2000; 2011) had been under development already for 8 years and the Verbmobil project (Wahlster 2000) had also spurred the development of grammars for German (GG; Müller & Kasper 2000; Crysmann 2003) and Japanese (Jacy; Siegel, Bender & Bond 2016). Project DeepThought (Callmeier, Eisele, Schäfer & Siegel 2004) was exploring methodologies for combining deep and shallow processing in practical applications across multiple languages. This inspired the development of the LinGO Grammar Matrix (Bender, Flickinger & Oepen 2002), which began as a core grammar, consisting of constraints hypothesized to be cross-linguistically useful, abstracted out of the ERG with reference to Jacy and GG. The goal of the Grammar Matrix is to serve as a starting point for the development of new grammars making it easy to reuse what has been learned in the development of existing grammars. In the years since, it has been extended to include "libraries" of analyses of cross-linguistically variable phenomena (e.g., Drellishak 2009; Bender et al. 2010).

DELPH-IN provides infrastructure (version control repositories, mailing lists, annual meetings) and an emphasis on open-source distribution of resources. Both of these support the collaboration of a global network of researchers working on

[19] This stands for DEep Linguistic Processing in HPSG INitiative; see http://www.delph-in.net, accessed 16 August 2019

interoperable components. These include repositories of linguistic knowledge, that is, both grammars and meta-grammars (including the Matrix and CLIMB, Fokkens 2014); processing engines that apply that knowledge for parsing and generation (discussed further below); software for supporting the development of grammar documentation (e.g., Hashimoto et al. 2008), software for creating treebanks (Oepen et al. 2004; Packard 2015; see also Section 4.1.4 below), and parse ranking models trained on them (Toutanova et al. 2005; see also Section 2.2.2 above), and software for robust processing, i. e. using the knowledge encoded in the grammars to return analyses for sentences even if the grammar deems them ungrammatical (Zhang & Krieger 2011; Buys & Blunsom 2017; Chen et al. 2018).

A key accomplishment of the DELPH-IN Consortium is the standardization of a formalism for the declaration of grammars (Copestake 2002a), a formalism for the semantic representations (Copestake et al. 2005), and file formats for the storage and interchange of grammar outputs (e.g., the forest that results from parsing a sentence, as well as the results of treebanking (Oepen 2001; Oepen et al. 2004)). These standards facilitate the development of multiple different parsing and generation engines which can all process the same grammars (including, so far, the LKB (Copestake 2002b), PET (Callmeier 2000), ACE[20], and Agree (Slayden 2012)), of multiple software systems for processing bulk grammar output ([incr tsdb()] (Oepen 2001), art[21], and PyDelphin[22]) and of multilingual downstream systems which can be adapted to additional languages by plugging in different grammars. These tools and standards have in turn helped support a thriving community of users who furthermore accumulate and share information about best practices. Melnik (2007: 234) credits this community and the information it shares as a key factor that makes the grammar engineering with DELPH-IN ecosystem more accessible to HPSG linguists, compared to other platforms like TRALE (see Section 3.1 above).

The DELPH-IN community maintains research interests in both linguistics and practical applications. The focus on linguistics means that DELPH-IN grammarians strive to create grammars which capture linguistic generalizations and model grammaticality. This, in turn, leads to grammars with lower ambiguity than one finds with treebank-trained grammars and, importantly, grammars which produce well-formed strings in generation. The focus on practical applications leads to several kinds of additional research goals. Practical applications require robust processing, which in turn requires methods for handling unknown words (e.g.,

[20] http://sweaglesw.org/linguistics/ace/, accessed 16 August 2019
[21] https://sweaglesw.org/linguistics/libtsdb/art.html, accessed 16 August 2019
[22] https://github.com/delph-in/pydelphin/, accessed 16 August 2019

Adolphs et al. 2008), methods for managing extra-grammatical mark-up in text such as in Wikipedia pages (e.g., Flickinger, Oepen & Ytrestøl 2010) and strategies for processing inputs that are ungrammatical, at least according to the grammar (e.g., Zhang & Krieger 2011, see also Section 4.2.3). Processing large quantities of text motivates performance innovations, such as supertagging or ubertagging (e.g., Matsuzaki et al. 2007; Dridan 2013, see also Section 2.2.2) to speed up processing times. Naturally occurring text can include very long sentences which can run up against processing limits. Supertagging helps some here, too, but other strategies include , or the task of breaking a long sentence into smaller ones without loss of meaning (Muszyńska 2016). Working with real-world text (rather than curated testsuites designed for linguistic research only) requires the integration of external components such as morphological analyzers (e.g., Marimon 2013) and named entity recognizers (e.g., Waldron et al. 2006; Schäfer et al. 2008). As described in Section 2.2.2, working with real-world applications requires parse ranking (e.g., Toutanova et al. 2005), and similarly ranking of generator outputs (known as *realization ranking*; e.g., Velldal 2009). Finally, research on embedding broad-coverage grammars in practical applications inspires work towards making sure that the semantic representations can serve as a suitable interface for external components (e.g., Flickinger et al. 2005). These efforts are also valuable from a strictly linguistic point of view, i. e. one not concerned with practical applications. First, the broader the coverage of a grammar, the more linguistic phenomena it can be used to explore. Second, external constraints on the form of semantic representations provide useful guide points in the development of semantic analyses.

3.3 Other HPSG and HPSG-inspired broad-coverage grammars

3.3.1 Alpino

Alpino[23] is a broad-coverage grammar of Dutch (Bouma, van Noord & Malouf 2001; van Noord & Malouf 2005; van Noord 2006). The main motivation is practical: to provide coverage and accuracy comparable to state-of-the-art parsers for English. Nonetheless, it also includes theoretically interesting analyses, such as for cross-serial dependencies (Bouma & van Noord 1998). In addition to using hand-written rules, lexical information (such as subcategorisation frames) has also been extracted from two existing lexicons, Celex (Baayen et al. 1995) and Parole (Kruyt & Dutilh 1997).

[23] http://www.let.rug.nl/vannoord/alp/Alpino/, accessed 16 August 2019

Alpino produces syntactic dependency graphs, following the annotation format of the Spoken Dutch Corpus (Oostdijk 2000). These dependencies are constructed directly in the feature-structure formalism, exploiting the fact that a feature structure can be formalised as a directed acyclic graph. Each lexical entry encodes a partial dependency graph, and these graphs are composed through phrase structure rules to give a dependency graph for a whole sentence.

Although these dependencies differ from the semantic dependencies discussed in Section 2.2.3, a common motivation is to make the representations easier to use in practical applications. To harmonize with other computational work on dependency parsing, Bouma & van Noord (2017) have also produced a mapping from this format to Universal Dependencies (UD; Nivre et al. 2016), as discussed in Section 4.1.4 below. Alpino uses a statistical model trained on a dependency treebank, and in fact the same statistical model can be used in both parsing and generation (de Kok et al. 2011).

3.3.2 Enju

Enju[24] (Miyao et al. 2005) is a broad-coverage grammar of English, semi-automatically acquired from the Penn Treebank (Marcus et al. 1993). This approach aims to reduce the cost of writing a grammar by leveraging existing resources. The basic idea is that, by viewing Penn Treebank trees as partial specifications of HPSG analyses, it is possible to infer lexical entries.

Miyao et al. converted the relatively flat trees in the Penn Treebank to binary-branching trees, and percolated head information through the trees. They also had to convert analyses for certain constructions, including subject-control verbs, auxiliary verbs, coordination, and extracted arguments. Each converted tree can then be combined with a small set of hand-written HPSG schemata, to induce a lexical entry for each word in the sentence.

Development of Enju has focused on performance in practical applications, and the grammar is supported by an efficient parser (Tsuruoka et al. 2004; Matsuzaki et al. 2007), using a probabilistic model for feature structures (Miyao & Tsujii 2008). Enju has been used in a variety of NLP tasks, as will be discussed in Section 4.2.2.

3.3.3 Babel

Babel is a broad-coverage grammar of German (Müller 1996; 1999). One interesting feature of this grammar is that it makes extensive use of discontinuous con-

[24] http://www.nactem.ac.uk/enju/, accessed 29 August 2019

stituents (Müller 2004a). Although this makes the worst-case parsing complexity much worse, parsing speed doesn't seem to suffer in practice. This mirrors the findings of Carroll (1994), discussed in Section 2.2.1 above.

4 Deployment of HPSG resources

There are several different ways in which computational resources based on HPSG are used. In Section 4.1, we first consider applications furthering linguistic research, including both language documentation and linguistic hypothesis testing. Then, in Section 4.2, we consider applications outside of linguistics.

4.1 Language documentation and linguistic hypothesis testing

As described by Müller (1999), Bender (2008) and Bender et al. (2011), grammar engineering, that is the building of grammars in software, is an essential technique for testing linguistic hypotheses at scale. By "at scale", we mean both against large quantities of data and as integrated models of language that handle multiple phenomena at once. In this section, we overview how this is done in the CoreGram and Grammar Matrix projects for cross-linguistic hypothesis testing, and in the AGGREGATION project in the context of language documentation.[25]

4.1.1 CoreGram

As described in Section 3.1, the CoreGram project develops grammars for a diverse set of languages, and shares constraints across grammars, in a bottom-up fashion, so that more similar languages share more constraints. Still, there are constraints shared across all of the grammars in the project which can be seen as an hypothesis about properties shared by all languages. Whenever the CoreGram project expands to cover a new language, it can be seen as a test of this hypothesis.

For example, the most general constraint set allows a language to have V2 word order (as exemplified by Germanic languages), but rules out verb-penultimate word order, as discussed by Müller (2015) (see also Müller 2019a, Chapter 10 of this volume). It also includes constraints for argument structure and linking (see

[25] Grammar engineering isn't specific to HPSG and in fact has a history going back to at least the early 1960s (Kay 1963; Zwicky et al. 1965; Petrick 1965; Friedman et al. 1971) and modern work in Lexical Functional Grammar (Butt et al. 1999), Combinatory Categorial Grammar (Baldridge et al. 2007), Grammatical Framework (Ranta 2009), and others. For reflections on grammar engineering for linguistic hypothesis testing in LFG, see Butt et al. 1999 and King 2016.

Wechsler, Koenig & Davis 2019, Chapter 9 of this volume), as well as for information structure (see Kuthy 2019, Chapter 24 of this volume).

4.1.2 Grammar Matrix

As noted in Section 3.2, the LinGO Grammar Matrix (Bender et al. 2002; 2010) was initially developed in the context of Project DeepThought with the goal of speeding up the development of DELPH-IN-style grammars for additional languages. It consists of a shared core grammar and a series of "libraries" of analyses for cross-linguistically variable phenomena. Both of these constitute linguistic hypotheses: the constraints in the core grammar are hypothesized to be cross-linguistically useful. However, in the course of developing grammars based on the Matrix for specific languages, it is not uncommon to find reasons to refine the core grammar. The libraries, in turn, are intended to cover the attested range of variation for the phenomena they model. Languages that are not covered by the analyses in the libraries provide evidence that the libraries need to be extended or refined.

Grammar Matrix grammar development is less tightly coordinated than that of CoreGram (see Section 3.1): in the typical use case, grammar developers start from the Grammar Matrix, but with their own independent copy of the Matrix core grammar. This impedes somewhat the ability of the Matrix to adapt to the needs of various languages (unless grammar developers report back to the Matrix developers). On the other hand, the Matrix libraries represent an additional kind of linguistic hypothesis testing: each library on its own represents one linguistic phenomenon, but the libraries must be interoperable with each other. This is the cross-linguistic analogue of how monolingual implemented grammars allow linguists to ensure that analyses of different phenomena are interoperable (Müller 1999: p.439–440; Bender 2008): the Grammar Matrix customization system allows its developers to test cross-linguistic libraries of analyses for interactions with other phenomena (Bender et al. 2011; Bender 2016). Without computational support — i. e. a computer keeping track of the constraints that make up each analysis, compiling them into specific grammars, and testing those grammars against testsuites — this problem space would be too complex for exploration.

4.1.3 AGGREGATION

In many ways, the most urgent need for computational support for linguistic hypothesis testing is the description of endangered languages. Implemented grammars can be used to process transcribed but unglossed text in order to find relevant examples more quickly, both of phenomena that have already been an-

alyzed and of phenomena that are as yet not well-understood.[26] Furthermore, treebanks constructed from implemented grammars can be tremendously valuable additions to language documentation (see Section 4.1.4 below). However, the process of building an implemented grammar is time-consuming, even with the start provided by a multilingual grammar engineering project like CoreGram, ParGram (Butt et al. 2002; King et al. 2005), the GF Resource Grammar Library Ranta (2009), or the Grammar Matrix.

This is the motivation for the AGGREGATION[27] project, which starts from two observations: (1) descriptive linguists produce extremely rich annotations on data in the form of interlinear glossed text (IGT); and (2) the Grammar Matrix's libraries are accessed through a customization system which elicits a grammar specification in the form of a series of choices describing either high-level typological properties or specific constraints on lexical classes and lexical rules. The goal of AGGREGATION is to automatically produce such grammar specifications on the basis of information encoded in IGT, to be used by the Grammar Matrix customization system to produce language-particular grammars. AGGREGATION uses different approaches for different linguistic subsystems. For example, it learns morphotactics by observing morpheme order in the training data, and the grouping of affixes together into position classes based on measures of overlap of stems they attach to (Wax 2014; Zamaraeva et al. 2017). For many kinds of syntactic information, it leverages syntactic structure projected from the translation line (English, easily parsed with current tools) through the gloss line (which facilitates aligning the language and translation lines) to the language line (Xia & Lewis 2007; Georgi 2016). Using this projected information, the AGGREGATION system can detect case frames for verbs, word order patterns, etc. (Bender et al. 2013; Zamaraeva et al. 2019).[28]

4.1.4 Treebanks and sembanks

A particularly valuable type of resource that can be derived from HPSG grammars are treebanks and sembanks. A *treebank* is a collection of text where each sentence is associated with a syntactic representation. A *sembank* has semantic representations (in some cases in addition to the syntactic ones). Treebanks and

[26] This methodology of using an implemented grammar as a sieve to sift the interesting examples out of corpora is demonstrated for English by Baldwin et al. (2005).

[27] http://depts.washington.edu/uwcl/aggregation/, accessed 16 August 2019

[28] The TypeGram project (Hellan & Beermann 2014) is in a similar spirit. TypeGram provides methods of creating HPSG grammars by encoding specifications of valence and inflection in particularly rich IGT and then creating grammars based on those specifications.

sembanks can be used for linguistic research, as the analyses allow for more detailed structure-based searches for phenomena of interest (Rohde 2005; Ghodke & Bird 2010; Kouylekov & Oepen 2014).[29] In the context of language documentation and description, searchable treebanks can also be a valuable addition, helping readers connect prose descriptions of linguistic phenomena to multiple examples in the corpus (Bender et al. 2012). In natural language processing, treebanks and sembanks are critical source material for training stochastic and neural parsers (see Section 4.2.3).

Traditional treebanks are created by doing a certain amount of preprocessing on data, including possibly chunking or CFG parsing, and then hand-correcting the result (Marcus et al. 1993; Banarescu et al. 2013). While this approach is a means to encode human insight about linguistic structure for later automatic processing, it is both inefficient and potentially error-prone. The Alpino project (van der Beek et al. 2002; see also Section 3.3.1 above) addresses this by first parsing the text with a broad-coverage HPSG-inspired grammar of Dutch and then having annotators select among the parses. The selection process is facilitated by allowing the annotators to mark potential lexical entries for words in the sentence at hand as correct, possibly correct, or wrong and to pre-mark some constituent boundaries. These constraints reduce the search space for the parser and consequently also the range of analyses the annotator has to consider before choosing one. A facility for adding one-off lexical entries to handle e.g., misspellings helps increase grammar coverage. Disambiguation is handled with the aid of *discriminants* i. e. properties true of some but not all trees in the parse forest (Carter 1997). Finally, the annotators may further edit analyses deemed insufficient. Though the underlying grammar is based on HPSG, the treebank stores dependency representations instead. The Alpino parser was similarly used to construct the Lassy treebanks of written Dutch (van Noord et al. 2013). In more recent work, these dependency representations have been mapped to the Universal Dependencies (UD) annotation standards (Nivre et al. 2016) to produce a UD treebank for Dutch (Bouma & van Noord 2017).

The Redwoods project (Oepen et al. 2004) also produces grammar-driven treebanks, in this case for English and without any post-editing of analyses.[30] As with Alpino, this is done by first parsing the corpus with the grammar and calcuating the discriminants for each parse forest. Finally, the treebanking software

[29] The WeSearch interface of Kouylekov & Oepen (2014) can be accessed at http://wesearch.delph-in.net/deepbank/search.jsp (accessed 16 August 2019).

[30] There are also Redwoods-style treebanks for other languages, including the Hinoki Treebank of Japanese (Bond et al. 2004) and the Tibidabo Treebank of Spanish (Marimon 2015).

stores not only the final full HPSG analysis that was selected, but also the decisions the annotator made about each discriminant. Thus when the grammar is updated to include for example a refinement to the semantic representations, the corpus can be reparsed and the decisions replayed, leaving only a small amount of further annotation work to be done to handle any additional ambguity introduced. The activity of treebanking in turn provides useful insight into grammatical analyses, including sources of spurious ambiguity and phenomena that are not yet properly handled and thus informs and spurs further grammar development. A downside to strictly grammar-based treebanking is that only items for which the grammar finds a reasonable parse can be included in the treebank. For many applications, this is not a drawback, so long as there are sufficient and sufficiently varied sentences that do receive analyses.

Finally, there are also automatically annotated treebanks. These are not as reliable as manually annotated treebanks, but they can be considerably larger. WikiWoods[31] covers 55m sentences of English (900m tokens). It was produced by Flickinger, Oepen & Ytrestøl (2010) and Solberg (2012) from the July 2008 dump of the full English Wikipedia, using the ERG and PET, with parse ranking trained on the manually treebanked subcorpus WeScience (Ytrestøl et al. 2009). As with the Redwoods treebanks, WikiWoods is updated with each release of the ERG.

4.2 Downstream applications

In this section, we discuss the use of HPSG grammars for practical tasks. There is a large number of applications, and we focus on several important applications here. In Section 4.2.1, we cover educational applications where a grammar is used directly. In Section 4.2.2, we cover applications where a grammar is used to provide features to help solve tasks in Natural Language Processing (NLP). Finally, in Section 4.2.3, we cover applications where a grammar is used to provide data for machine learning systems.[32]

4.2.1 Education

Precise syntactic analyses can be useful in language teaching, in order to automatically identify errors and give feedback to the student. In order to model

[31] http://moin.delph-in.net/WikiWoods, accessed 16 August 2019

[32] The DELPH-IN community maintains an updated list of applications of DELPH-IN software and resources at http://moin.delph-in.net/DelphinApplications (accessed 16 August 2019).

common mistakes, a grammar can be extended with so-called *mal-rules*. A mal-rule is like a normal rule, in that it licenses a construction, and can be treated the same during parsing — however, given a parse, the presence of mal-rules indicates that the student needs to be given feedback (Bender et al. 2004; Flickinger & Yu 2013; Morgado da Costa et al. 2016). A large scale system implementing this kind of computer-aided teaching has been developed by the Education Program for Gifted Youth at Stanford University, using the ERG (Suppes et al. 2014). This system has reached tens of thousands of elementary and middle school children, and has been found to improve the school results of underachieving children.

Another way to use a precision grammar is to automatically produce teaching materials. Given a semantic representation, a grammar can generate one or more sentences. Flickinger (2017) uses the ERG to produce practice exercises for a student learning first-order logic. For each exercise, the student is presented with an English sentence and is supposed to write down the corresponding first-order logical form. By using a grammar, the system can produce syntactically varied questions and automatically evaluate the student's answer.

4.2.2 NLP tasks

Much NLP work focuses on specific *tasks*, where a system is presented with some input, and required to produce an output, with a clearly-defined metric to determine how well the system performs. HPSG grammars have been used in a range of such tasks, where the syntactic and semantic analyses provide useful features.

Information retrieval is the task of finding relevant documents for a given query. For example, Schäfer et al. (2011) present a tool for searching the ACL Anthology, using the ERG. *Information extraction* is the task of identifying useful facts in a collection of documents. For example, Reiplinger et al. (2012) aim to identify definitions of technical concepts from English text, in order to automatically construct a glossary. They find that using the ERG reduces noise in the candidate definitions. Miyao et al. (2008) aim to identify protein-protein interactions in the English biomedical literature, using Enju.

For these tasks, some linguistic phenomena are particularly important, such as negation and hedging (including adverbs like *possibly*, modals like *may*, and verbs of speculation like *suggest*). When it comes to identifying facts asserted in a document, a clause that has been negated or hedged should be treated with caution. MacKinlay et al. (2012) consider the biomedical domain, evaluating on the BioNLP 2009 Shared Task (Kim et al. 2009), where they outperform previous approaches for negation, but not for speculation. Velldal et al. (2012) consider negation and speculation in biomedical text, evaluating on the CoNLL 2010

Shared Task (Farkas et al. 2010), where they outperform previous approaches. Packard et al. (2014) propose a general-purpose method for finding the scope of negation in an MRS, evaluating on the *SEM 2012 Shared Task (Morante & Blanco 2012). They find that converting the output of the ERG with a relatively simple set of rules achieves high performance on this English dataset, and combining this approach with a purely statistical system outperforms previous approaches. Zamaraeva et al. (2018) use the ERG for detection and then use that information to refine the (machine-learning) features in a system that classifies English pathology reports and improve system performance. A common finding from these studies is that a system using the output of the ERG tends to have high precision (items identified by the system tend to be correct) but low recall (items are often overlooked by the system). One reason for low recall is that the grammar does not cover all sentences in natural text. As we will see in Section 4.2.3, recent work on robust parsing may help to close this coverage gap.

Negation resolution is also included in Oepen et al.'s (2017) Shared Task on Extrinsic Parser Evaluation. As mentioned in Section 2.2.3, dependency graphs can provide a useful tool in NLP tasks, and this shared task aims to evaluate the use of dependency representations (both semantic and syntactic), for three downstream applications: biomedical information extraction, negation resolution, and fine-grained opinion analysis. Some participating teams use DM dependencies[33] (Schuster et al. 2017; Chen et al. 2017). The results of this shared task suggest that, compared to other dependency representations, DM is particularly useful for negation resolution.

Another task where dependency graphs have been used is *summarization*. Most existing work on this task focuses on so-called *extractive summarization*: given an input document, a system forms a summary by extracting short sections of the input. This is in contrast to *abstractive summarization*, where a system generates new text based on the input document. Extractive summarization is limited, but widely used because it is easier to implement. Fang et al. (2016) show how a wide-coverage grammar like the ERG makes it possible to implement an abstractive summarizer with state-of-the-art performance. After parsing the input document into logical propositions, the summarizer prunes the set of propositions using a cognitively inspired model. A summary is then generated based on the pruned set of propositions. Because no text is directly extracted from the input document, it is possible to generate a more concise summary.

Finally, no discussion of NLP tasks would be complete without including *machine translation*. A traditional grammar-based approach uses three grammars:

[33] DM stands for DELPH-IN MRS dependencies; see (7).

a grammar for the source language, a grammar for the target language, and a *transfer grammar*, which converts semantic representations for the source language to semantic representations for the target language (Oepen et al. 2007; Bond et al. 2011). Translation proceeds in three steps: parse the source sentence, transfer the semantic representation, and generate a target sentence. The transfer grammar is needed both to find appropriate lexical items, and also to convert semantic representations when languages differ in how an idea might be expressed. The difficulty in writing a transfer grammar that is robust enough to deal with arbitrary input text means that statistical systems might be preferred. Horvat (2017) explores the use of statistical techniques, skipping out the transfer stage: a target-language sentence is generated directly from a semantic representation for the source language. Goodman (2018) explores the use of statistical techniques within the paradigm of parsing, transferring, and generating.

4.2.3 Data for machine learning

In Section 4.2.2, we described how HPSG grammars can be directly incorporated into NLP systems. Another use of HPSG grammars in NLP is to generate data on which a statistical system can be trained.

For example, one limitation of using an HPSG grammar in an NLP system is that the grammar is unlikely to cover all sentences in the data (Flickinger et al. 2012). One way to overcome this coverage gap is to train a statistical system to produce the same output as the grammar. The idea is that the trained system will be able to generalise to sentences that the grammar does not cover. Oepen et al. (2014), Oepen et al. (2015), and Oepen et al. (2019) present shared tasks on semantic dependency parsing, including both DM dependencies and Enju predicate-argument structures. As of 2015, the best-performing systems in these shared tasks could already produce dependency graphs almost as accurately as grammar-based parsers (for sentences where the grammar has coverage). Similarly, Buys & Blunsom (2017) develop a parser for EDS and DMRSDependency Minimal Recursion Semantics (DMRS) which performs almost as well as a grammar-based parser, but has full coverage, and can run 70 times faster.

In fact, in more recent work, the difference in performance has been effectively closed. Chen et al. (2018) consider parsing to EDS and DMRS graphs, and actually achieve slightly higher accuracy with their system, compared to a grammar-based parser. Unlike the previous statistical approaches, Chen et al. do not just train on the desired dependency graphs, but also use information in the phrase-structure tree. They suggest that using this information allows their system to learn compositional rules mirroring composition in the grammar, and thereby

allows their system to generalise better.

Another application of HPSG-derived dependency graphs is for *distributional semantics*. Here, the aim is to learn the meanings of words from a corpus, exploiting the fact that the context of a word tells us something about its meaning. This is known as the *distributional hypothesis*, an idea with roots in American structuralism (Harris 1954) and British lexicology (Firth 1951; 1957). Most work on distributional semantics learns a *vector space model*, where the meaning of each word is represented as a point in a high-dimensional vector space (for an overview, see Erk 2012 and Clark 2015). However, Emerson (2018) argues that vector space models cannot capture various aspects of meaning, including logical structure, as well as phenomena like polysemy. Instead, Emerson presents a distributional model which can learn truth-conditional semantics, using a parsed corpus like WikiWoods (see Section 4.1.4). This approach relies on the semantic analyses given by a grammar, as well as the infrastructure to parse a large amount of text.

Finally, there are also applications using grammars not to parse, but to generate. Kuhnle & Copestake (2018) consider the task of *visual question answering*, where a system is presented with an image and a question about the image, and must answer the question. This task requires language understanding, reference resolution, and grounded reasoning, in a way that is relatively well-defined. However, for many existing datasets, there are biases in the questions which mean that high performance can be achieved without true language understanding. For this reason, there is increasing interest in artificial datasets, which are controlled to make sure that high performance requires true understanding. Kuhnle & Copestake present ShapeWorld, a configurable system for generating artificial data. The system generates an abstract representation of a scene (coloured shapes in different configurations), and then generates an image and a caption based on this representation. The use of a broad-coverage grammar is crucial in allowing the system to be configurable and scale across a variety of syntactic constructions.

5 Linguistic insights

In Section 4.1 above, we described multiple ways in which computational methods can be used in the service of linguistic research, especially in testing linguistic hypotheses. Here, we highlight a few ways in which grammar engineering work in HPSG has turned up linguistic insights that had not previously been

discovered through non-computational means.[34]

5.1 Ambiguity

As discussed in Section 2.2.2, the scale of ambiguity has become clear now that broad-coverage precision grammars are available. By taking both coverage and precision seriously, it is possible to investigate ambiguity on a large scale, quantifying the sources of ambiguity and the information needed to resolve it. For example, Toutanova et al. (2002; 2005) found that in the Redwoods treebank (3rd Growth), roughly half of the ambiguity was lexical, and half syntactic. They also showed how combining sources of information (such as both semantic and syntactic information) is important for resolving ambiguity, and argue that using multiple kinds of information in this way is consistent with probabilistic approaches in psycholinguistics.

5.2 Long-tail phenomena

One of the strengths of HPSG as a theoretical framework is that it allows for the analysis of both "core" and "peripheral" phenomena within a single, integrated model. Indeed, by implementing large-scale grammars across a range of languages, it becomes possible to investigate the extent to which a particular phenomenon should be considered "core", "peripheral", or something in between (Müller 2014).

In fact, when working with actual data and large-scale grammars, it quickly becomes apparent just how long the long-tail of "peripheral" phenomena is. Furthermore, the sustained development of broad-coverage linguistic resources makes it possible to bring into view more and more low-frequency phenomena (or low-frequency variations on relatively high-frequency phenomena). A case in point is the range of raising and control valence frames found in the ERG (Flickinger 2000; 2011). As of the 2018 release, the ERG includes over 60 types for raising and control predicates, including verbs, adjectives, and nouns, many of which are not otherwise discussed in the syntactic literature. These include such low-frequency types as the one for *incumbent*, which requires an expletive *it* subject, an obligatory *(up)on* PP complement, and an infinitival VP complement, and which establishes a control relation between the object of *on* and the VP's missing subject:[35]

(8) It is incumbent on you to speak plainly.

[34] For similar reflections from the point of view of LFG, see King (2016).

[35] Our thanks to Dan Flickinger for this example.

5.3 Analysis-order effects

Grammar engineering means making analyses specific and then being able to build on them. This has both benefits and drawbacks: on the one hand, it means that additional grammar engineering work can build directly on the results of previous work. It also means that any additional grammar engineering work is constrained by the work it is building on. Fokkens (2014) observes this phenomenon and notes that it introduces artifacts: the form an implemented grammar takes is partially the result of the order in which the grammar engineer considered phenomena to implement. This is probably also true for non-computational work, as theoretical ideas developed with particular phenomena (and indeed languages) in mind influence the questions with which researchers approach additional phenomena. Fokkens proposes that the methodology of meta-grammar engineering can be used to address this problem: using her CLIMB methodology, rather than deciding between analyses of a given phenomenon without input from later-studied phenomena, the grammar engineer can maintain multiple competing analyses through time and break free, at least partially, of the effects of the timeline of grammar development. The central idea is that the grammar writer develops a meta-grammar, like the Grammar Matrix customization system (see Section 4.1.2), but for a single language. This customization system maintains alternate analyses of particular phenomena which are invoked via grammar specifications so the different versions of the grammar can be compiled and tested.

6 Summary

In this chapter, we have attempted to illuminate the landscape of computational work in HPSG. We have discussed how HPSG as a theory supports computational work, described large-scale computational projects that use HPSG, highlighted some applications of implemented grammars in HPSG, and explored ways in which computational work can inform linguistic research. This field is very active and our overview necessarily incomplete. Nonetheless, it is our hope that the pointers and overview provided in this chapter will serve to help interested readers connect with on-going research in computational linguistics using HPSG.

Acknowledgements

We'd like to thank Stephan Oepen for helpful comments on an early draft of this chapter and Stefan Müller for detailed comments as volume editor.