Chapter 29

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 (Flickinger, Pollard & Wasow 2019: for an overview, see, Chapter 2 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. 2019. Computational linguistics and grammar engineering. In Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.), *Head-Driven Phrase Structure Grammar: The handbook*, 773–815. Berlin: Language Science Press. DOI:??

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 (Müller 2019c: for a comparison of HPSG and Construction Grammar, see, 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 HP labs 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 only a few. By isolating any given constraint on one type (as opposed

[1] There are implementations of Minimalism, notably Stabler 1997 and Herring 2016. (See also Torr et al. 2019 for a recent broad-verage, 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.

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* (Johnson 1988: Section 3.4). However, as discussed in the previous section, there

[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).

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 rules. However, if we can bound the number of possible feature structures as C,

[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".

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 list 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 complement's subjects (e.g., in raising/control constructions), 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 complexity 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).

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

[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.

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.

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

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

[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.

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*. 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 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

[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.

(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.

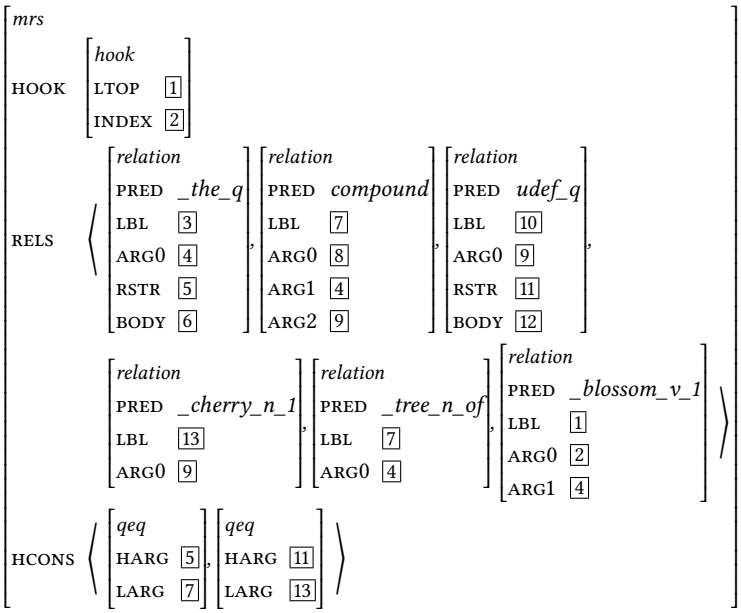
2.2.3 Semantic dependencies

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 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. A comparison of MRS, DMRS, and DM is given in Figures 1 and 2. The existence of such dependency graph formalisms, as well as software packages to manipulate such graphs (Copestake et al. 2016), has made it easier to use HPSG grammars in a number of practical

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

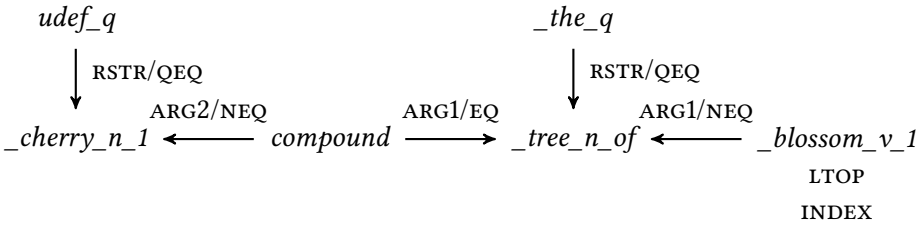


(a) MRS, as a feature structure.

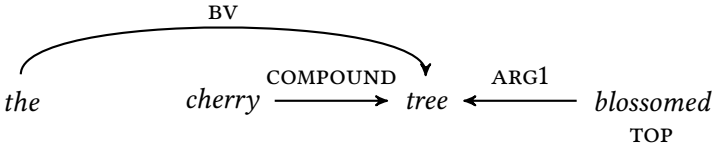
$$\begin{aligned}
&\textsc{index}: e_1 \\
l_1 &: _the_q(x_1, h_1, h_2), h_1 \textsc{ qeq } l_4 \\
l_2 &: udef_q(x_2, h_3, h_4), h_3 \textsc{ qeq } l_3 \\
l_3 &: _cherry_n_1(x_2) \\
l_4 &: _tree_n_of(x_1), compound(e_2, x_1, x_2) \\
\textsc{ltop}, l_5 &: _blossom_v_1(e_1, x_1)
\end{aligned}$$

(b) MRS, abstractly.

Figure 1: Minimal Recursion Semantics (MRS) representation for the sentence *The cherry tree blossomed*, as produced by the English Resource Grammar (ERG). For simplicity, we have omitted some details, including features such as number and tense, individual constraints (ICONS), and the use of difference lists. 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. QEQ (equality modulo quantifiers) indicates a scopal constraint, with quantifiers possibly intervening (for details, see Copestake et al. 2005 or Koenig & Richter (2019), Chapter 24 of this volume).



(a) A Dependency Minimal Recursion Semantics (DMRS) representation that captures all of the information in the MRS in Figure 1. Predicates are represented as nodes, while semantic roles and scopal constraints are represented as dependencies. RSTR stands for "restriction" (of a quantifier). EQ indicates that the linked nodes share a label in the MRS (which is generally true for modifiers), while NEQ indicates that they don't share a label.



(b) A DELPH-IN MRS Dependencies (DM) representation. 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). BV stands for "bound variable" (of a quantifier).

Figure 2: MRS-based dependency graphs for the sentence *The cherry tree blossomed*, based on the MRS given in Figure 1.

tasks, as we will discuss in Section 4.2.

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[16] 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. 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;

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

2013).

3.2 The DELPH-IN Consortium

The DELPH-IN[17] 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 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

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

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[18], and Agree (Slayden 2012)), of multiple software systems for processing bulk grammar output ([incr tsdb()] (Oepen 2001), art[19], and PyDelphin[20]) 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., 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 *sentence chunking*, 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

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

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[21] 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).

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

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

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[22] is a broad-coverage grammar of English, semi-automatically acquired from the Penn Treebank (Miyao et al. 2005). 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 constituents (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.

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

4.1 Language documentation and linguistic hypothesis testing

As decribed 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 mulitple 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.[23]

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) (Müller 2019a: see also, Chapter 10 of this volume). It also includes constraints for argument structure and linking (Wechsler, Koenig & Davis 2019: see, Chapter 9 of this volume), as well as for information structure (Kuthy 2019: see, Chapter 25 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

[23] 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 Funcational 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.

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; 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 analyzed and of phenomena that are as yet not well-understood.[24] 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, the GF Resource Library, or the Grammar Matrix.

This is the motivation for the AGGREGATION[25] project, which starts from two observations: (1) descriptive linguists produce extremely rich annotations

[24] 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).

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

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).[26]

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 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).[27] 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

[26] 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.

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

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) 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.[28] 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 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.

[28] 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).

Finally, there are also automatically annotated treebanks. These are not as reliable as manually annotated treebanks, but they can be considerably larger. WikiWoods[29] 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.[30]

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 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

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

[30] 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).

a natural language 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, 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 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 dataset, and combining this approach with a purely statistical system outperforms previous approaches. Zamaraeva et al. (2018) use the ERG for negation detection and then use that information to refine the (machine-learning) features in a system that classifies 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[31] (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: 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.

[31] DM stands for DELPH-IN MRS dependencies; see Figure 2.

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) and Oepen et al. (2015) present shared tasks on semantic dependency parsing, including both DM dependencies and Enju predicate-argument structures. The best-performing systems in these shared tasks can 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 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.[32]

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.

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

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 subject, and an infinitival VP complement, and which establishes a control relation between the object of *on* and the VP's missing subject:[33]

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

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

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

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.

References

Adolphs, Peter, Stephan Oepen, Ulrich Callmeier, Berthold Crysmann, Dan Flickinger & Bernd Kiefer. 2008. Some fine points of hybrid natural language parsing. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis & Daniel Tapias (eds.), *Proceedings of the sixth international conference on language resources and evaluation (LREC'08)*. Marrakech, Morocco: European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Baayen, R Harald, Richard Piepenbrock & Leon Gulikers. 1995. *The CELEX lexical database*. Distributed by the Linguistic Data Consortium, University of Pennsylvania.

Baldridge, Jason, Sudipta Chatterjee, Alexis Palmer & Ben Wing. 2007. DotCCG and VisCCG: Wiki and programming paradigms for improved grammar engineering with OpenCCG. In Tracy Holloway King & Emily M. Bender (eds.), *Grammar Engineering across Frameworks 2007* (Studies in Computational Linguistics ONLINE), 5–25. Stanford, CA: CSLI Publications. http://csli-publications.stanford.edu/GEAF/2007/, accessed 2018-2-25.

Baldwin, Timothy, John Beavers, Emily M. Bender, Dan Flickinger, Ara Kim & Stephan Oepen. 2005. Beauty and the beast: What running a broad-coverage precision grammar over the BNC taught us about the grammar — and the corpus. In Stephan Kepser & Marga Reis (eds.), *Linguistic evidence: Empirical, theoretical, and computational perspectives* (Studies in Generative Grammar 85), 49–69. Mouton de Gruyter.

Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer & Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 178–186. Sofia, Bulgaria: Association for Computational Linguistics. http://aclweb.org/anthology/W13-2322.

Bangalore, Srinivas & Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics* 25(2). 237–265. http://aclweb.org/anthology/J99-2004.

Bender, Emily M. 2008. Grammar engineering for linguistic hypothesis testing. In Nicholas Gaylord, Alexis Palmer & Elias Ponvert (eds.), *Proceedings of the Texas Linguistics Society X Conference: Computational linguistics for less-studied languages*, 16–36. Stanford CA: CSLI Publications ONLINE.

Bender, Emily M. 2016. Linguistic typology in natural language processing. *Linguistic Typology* 20(3). 645–660.

Bender, Emily M., Scott Drellishak, Antske Fokkens, Laurie Poulson & Safiyyah Saleem. 2010. Grammar customization. *Research on Language & Computation* 8(1). 23–72. http://dx.doi.org/10.1007/s11168-010-9070-1. 10.1007/s11168-010-9070-1.

Bender, Emily M., Dan Flickinger & Stephan Oepen. 2011. Grammar engineering and linguistic hypothesis testing: Computational support for complexity in syntactic analysis. In E.M. Bender & Arnold J.E. (eds.), *Language from a cognitive perspective: Grammar, usage and processing*, 5–29. Stanford, CA: CSLI Publications.

Bender, Emily M., Dan Flickinger, Stephan Oepen, Annemarie Walsh & Timothy Baldwin. 2004. Arboretum. Using a precision grammar for grammar checking

in CALL. In *Proceedings of the InSTIL Symposium on NLP and Speech Technologies in Advanced Language Learning Systems*. Venice, Italy.

Bender, Emily M., Daniel P. Flickinger & Stephan Oepen. 2002. The Grammar Matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In John Carroll, Nelleke Oostdijk & Richard Sutcliffe (eds.), *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, 8–14. Taipei, Taiwan.

Bender, Emily M., Sumukh Ghodke, Timothy Baldwin & Rebecca Dridan. 2012. From database to treebank: enhancing hypertext grammars with grammar engineering and treebank search. In Sebastian Nordhoff & Karl-Ludwig G. Poggeman (eds.), *Electronic grammaticography*, 179–206. Honolulu: University of Hawaii Press.

Bender, Emily M, Michael Wayne Goodman, Joshua Crowgey & Fei Xia. 2013. Towards creating precision grammars from interlinear glossed text: inferring large-scale typological properties. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, 74–83. Sofia, Bulgaria: Association for Computational Linguistics. http://aclweb.org/anthology/W13-2710.

Boguraev, Bran, John Carroll, Ted Briscoe & Claire Grover. 1988. Software support for practical grammar development. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, 54–58. http://aclweb.org/anthology/C88-1012.

Bond, Francis, Sanae Fujita, Chikara Hashimoto, Kaname Kasahara, Shigeko Nariyama, Eric Nichols, Akira Ohtani, Takaaki Tanaka & Shigeaki Amano. 2004. The Hinoki treebank: a treebank for text understanding. In *Proceedings of the 1st International Joint Conference on Natural Language Processing (IJCNLP)*, 158–167.

Bond, Francis, Stephan Oepen, Eric Nichols, Dan Flickinger, Erik Velldal & Petter Haugereid. 2011. Deep open-source machine translation. *Machine Translation* 25(2). 87.

Bouma, Gosse, Robert Malouf & Ivan A. Sag. 2001. Satisfying constraints on extraction and adjunction. *Natural Language and Linguistic Theory* 19(1). 1–65.

Bouma, Gosse & Gertjan van Noord. 1998. Word order constraints on verb clusters in German and Dutch. In Erhard W. Hinrichs, Andreas Kathol & Tsuneko Nakazawa (eds.), *Complex predicates in nonderivational syntax* (Syntax and Semantics 30), 43–72. San Diego: Academic Press.

Bouma, Gosse & Gertjan van Noord. 2017. Increasing return on annotation investment: the automatic construction of a Universal Dependency Treebank for Dutch. In *Proceedings of the NoDaLiDa 2017 workshop on Universal Dependencies (UDW 2017)*, 19–26.

Bouma, Gosse, Gertjan van Noord & Robert Malouf. 2001. Alpino: wide-coverage computational analysis of Dutch. In Walter Daelemans, Khalil Sima'an, Jorn Veenstra & Jakub Zavrel (eds.), *Computational linguistics in the Netherlands 2000: Selected papers from the Eleventh CLIN Meeting* (Language and Computers 37). Amsterdam/New York, NY: Rodopi.

Butt, Miriam, Tracy Holloway King, María-Eugenia Niño & Frédérique Segond. 1999. *A grammar writer's cookbook* (CSLI Lecture Notes 95). Stanford, CA: CSLI Publications.

Buys, Jan & Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL), long papers*, 1215–1226. http://aclweb.org/anthology/P17-1112.

Callmeier, Ulrich. 2000. PET: A platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering* 6(1). 99–108. Special Issue on Efficient Processing with HPSG: Methods, Systems, Evaluation.

Callmeier, Ulrich, Andreas Eisele, Ulrich Schäfer & Melanie Siegel. 2004. The DeepThought core architecture framework. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*. Lisbon, Portugal: European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2004/pdf/603.pdf.

Carroll, John. 1994. Relating complexity to practical performance in parsing with wide-coverage unification grammars. In James Pustejovsky (ed.), *32th Annual Meeting of the Association for Computational Linguistics. Proceedings of the conference*, 287–294. Las Cruses: Association for Computational Linguistics.

Carroll, John Andrew. 1993. *Practical unification-based parsing of natural language*. University of Cambridge dissertation. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-314.ps.gz.

Carroll, John, Ann Copestake, Dan Flickinger & Victor Poznański. 1999. An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation*, 86–95.

Carter, David. 1997. The TreeBanker. a tool for supervised training of parsed corpora. In *Proceedings of the 1997 ACL workshop on computational environments for grammar development and linguistic engineering (ENVGRAM)*, 9–15. Madrid, Spain. http://aclweb.org/anthology/W97-1502.

Chen, Yufei, Junjie Cao, Weiwei Sun & Xiaojun Wan. 2017. Peking at EPE 2017: a comparison of tree approximation, transition-based and maximum subgraph models for semantic dependency analysis. In *Proceedings of the 2017 shared task on extrinsic parser evaluation, at the 4th International Conference on Dependency Linguistics and the 15th International Conference on Parsing Technologies*, 60–64. http://svn.nlpl.eu/epe/2017/public/proceedings.pdf.

Chen, Yufei, Weiwei Sun & Xiaojun Wan. 2018. Accurate SHRG-based semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), long papers*, 408–418. http://aclweb.org/anthology/P18-1038.

Chomsky, Noam. 1995. *The Minimalist Program* (Current Studies in Linguistics 28). Cambridge, MA: MIT Press.

Clark, Stephen. 2015. Vector space models of lexical meaning. In Shalom Lappin & Chris Fox (eds.), *The handbook of contemporary semantic theory*, 2nd, chap. 16, 493–522. Wiley. http://www.cl.cam.ac.uk/~sc609/pubs/sem_handbook.pdf.

Clark, Stephen & James R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*. http://aclweb.org/anthology/C04-1041.

Copestake, Ann. 2002a. Definitions of typed feature structures. In Stephan Oepen, Dan Flickinger, Jun-ichi Tsujii & Hans Uszkoreit (eds.), *Collaborative language engineering*, 227–230. Stanford, CA: CSLI Publications.

Copestake, Ann. 2002b. *Implementing typed feature structure grammars* (CSLI Lecture Notes 110). Stanford, CA: CSLI Publications.

Copestake, Ann. 2009. Slacker semantics: why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 1–9. http://aclweb.org/anthology/E09-1001.

Copestake, Ann, Guy Emerson, Michael Wayne Goodman, Matic Horvat, Alexander Kuhnle & Ewa Muszyńska. 2016. Resources for building applications with Dependency Minimal Recursion Semantics. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*, 1240–1247. http://www.lrec-conf.org/proceedings/lrec2016/pdf/634_Paper.pdf.

Copestake, Ann, Daniel P. Flickinger, Carl J. Pollard & Ivan A. Sag. 2005. Minimal Recursion Semantics: An introduction. *Research on Language and Computation* 3(2–3). 281–332. DOI:10.1007/s11168-006-6327-9

Crysmann, Berthold. 2003. On the efficient implementation of German verb placement in HPSG. In *Proceedings of RANLP 2003*, 112–116. Borovets, Bulgaria.

de Kok, Daniël, Barbara Plank & Gertjan van Noord. 2011. Reversible stochastic attribute-value grammars. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL), short papers*, 194–199. http://aclweb.org/anthology/P11-2034.

Dellert, Johannes, Kilian Evang & Frank Richter. 2010. *Kahina, a debugging framework for logic programs and TRALE*. Presentation at the HPSG 2010 Conference.

Dellert, Johannes, Kilian Evang & Frank Richter. 2013. Kahina: A hybrid trace-based and chart-based debugging system for grammar engineering. In Denys Duchier & Yannick Parmentier (eds.), *Proceedings of the workshop on high-level methodologies for grammar engineering (HMGE 2013), Düsseldorf*, 75–86.

Drellishak, Scott. 2009. *Widespread but not universal: Improving the typological coverage of the Grammar Matrix*. University of Washington Doctoral dissertation.

Dridan, Rebecca. 2013. Ubertagging: joint segmentation and supertagging for English. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1201–1212. http://aclweb.org/anthology/D13-1120.

Emerson, Guy. 2018. *Functional distributional semantics: learning linguistically informed representations from a precisely annotated corpus*. University of Cambridge dissertation. https://www.repository.cam.ac.uk/bitstream/handle/1810/284882/thesis.pdf.

Erk, Katrin. 2012. Vector space models of word meaning and phrase meaning: a survey. *Language and Linguistics Compass* 6(10). 635–653. https://onlinelibrary.wiley.com/doi/epdf/10.1002/lnco.362.

Fang, Yimai, Haoyue Zhu, Ewa Muszyńska, Alexander Kuhnle & Simone Teufel. 2016. A proposition-based abstractive summariser. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, 567–578. http://aclweb.org/anthology/C16-1055.

Farkas, Richárd, Veronika Vincze, György Móra, János Csirik & György Szarvas. 2010. The CoNLL-2010 shared task: learning to detect hedges and their scope in natural language text. In *Proceedings of the 14th Conference on Computational Natural Language Learning (CoNNL), shared task*, 1–12. http://aclweb.org/anthology/W10-3001.

Firth, John Rupert. 1951. Modes of meaning. *Essays and Studies of the English Association* 4. 118–149. Reprinted in: Firth (1957), *Papers in Linguistics*, chapter 15, pages 190–215.

Firth, John Rupert. 1957. A synopsis of linguistic theory 1930–1955. In John Rupert Firth (ed.), *Studies in linguistic analysis* (Special volume of the Philological Society), chap. 1, 1–32. Blackwell.

Flickinger, Dan, Jan Tore Lønning, Helge Dyvik, Stephan Oepen & Francis Bond. 2005. SEM-I rational MT: enriching deep grammars with a semantic interface for scalable machine translation. In *Proceedings of machine translation summit X*, 165–172.

Flickinger, Dan, Stephan Oepen & Emily M. Bender. 2017. Sustainable Development and Refinement of Complex Linguistic Annotations at Scale. In Nancy Ide & James Pustejovsky (eds.), *Handbook of Linguistic Annotation*, 353–377. Dordrecht: Springer Netherlands. http://dx.doi.org/10.1007/978-94-024-0881-2_14. DOI:10.1007/978-94-024-0881-2_14

Flickinger, Dan, Stephan Oepen & Gisle Ytrestøl. 2010. WikiWoods: syntacto-semantic annotation for English Wikipedia. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)*, 1665–1671. http://www.lrec-conf.org/proceedings/lrec2010/pdf/432_Paper.pdf.

Flickinger, Dan, Stephan Oepen & Gisle Ytrestøl. 2010. Wikiwoods: syntacto-semantic annotation for english wikipedia. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner & Daniel Tapias (eds.), *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, 1665–1671. Valletta, Malta: European Language Resources Association (ELRA).

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: The handbook*, 7–43. Berlin: Language Science Press. DOI:??

Flickinger, Dan & Jiye Yu. 2013. Toward more precision in correction of grammatical errors. In *Proceedings of the 17th Conference on Computational Natural Language Learning (CoNLL): shared task*, 68–73. http://aclweb.org/anthology/W13-3609.

Flickinger, Dan, Yi Zhang & Valia Kordoni. 2012. DeepBank: a dynamically annotated treebank of the Wall Street Journal. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories (TLT)*, 85–96. http://tlt11.clul.ul.pt/ProceedingsTLT11.tgz.

Flickinger, Daniel. 2017. Generating English paraphrases from logic. In Martijn Wieling, Gosse Bouma & Gertjan van Noord (eds.), *From semantics to dialectometry: festschrift in honour of john nerbonne*, 99–108. Springer.

Flickinger, Daniel P. 1987. *Lexical rules in the hierarchical lexicon*. Stanford University dissertation.

Flickinger, Daniel P. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering* 6(1). 15–28. Special Issue on Efficient Processing with HPSG: Methods, Systems, Evaluation.

Flickinger, Daniel P. 2011. Accuracy vs. robustness in grammar engineering. In Emily M. Bender & Jennifer E. Arnold (eds.), *Language from a cognitive perspective: Grammar, usage, and processing*, 31–50. Stanford, CA: CSLI Publications.

Fokkens, Antske Sibelle. 2014. *Enhancing empirical research for linguistically motivated precision grammars*. Department of Computational Linguistics, Universität des Saarlandes dissertation.

Friedman, Joyce, Thomas H. Bredt, Robert W. Doran, Bary W. Pollack & Theodore S. Martner. 1971. *A computer model of Transformational Grammar* (Mathematical Linguistics and Automatic Language Processing 9). New York: Elsevier.

Gazdar, Gerald & Geoffrey K. Pullum. 1985. Computationally relevant properties of natural languages and their grammars. *New Generation Computing* (3). 237–306. Reprinted in Savitch, Bach, Marsh, and Safran-Naveh (eds.), *The Formal Complexity of Natural Language*.

Georgi, Ryan. 2016. *From Aari to Zulu: massively multilingual creation of language tools using interlinear glossed text*. University of Washington dissertation.

Ghodke, Sumukh & Steven Bird. 2010. Fast query for large treebanks. In *Proceedings of the 8th Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), long papers*, 267–275. Los Angeles, California. http://aclweb.org/anthology/N10-1034.

Goodman, Michael Wayne. 2018. *Semantic operations for transfer-based machine translation* dissertation. https://digital.lib.washington.edu/researchworks/bitstream/handle/1773/42432/Goodman_washington_0250E_18405.pdf.

Harris, Zellig Sabbetai. 1954. Distributional structure. *Word* 10. 146–162. Reprinted in: Harris (1970), *Papers in Structural and Transformational Linguistics*, chapter 36, pages 775–794; Harris (1981), *Papers on Syntax*, chapter 1, pages 3-22.

Hashimoto, Chikara, Francis Bond, Takaaki Tanaka & Melanie Siegel. 2008. Semi-automatic documentation of an implemented linguistic grammar augmented with a treebank. *Language Resources and Evaluation* 42(2). 117–126.

Hellan, Lars & Dorothee Beermann. 2014. Inducing grammars from IGT. In Zygmunt Vetulani & Joseph Mariani (eds.), *Human language technology challenges for computer science and linguistics*, 538–547. Cham: Springer International Publishing.

Herring, Joshua. 2016. *Grammar construction in the Minimalist Program*. Indiana University dissertation.

Horvat, Matic. 2017. *Hierarchical statistical semantic translation and realization*. University of Cambridge dissertation. http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-913.pdf.

Hudson, Dick. 2019. HPSG and Dependency Grammar. In Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.), *Head-Driven Phrase Structure Grammar: The handbook*, 1003–1006. Berlin: Language Science Press. DOI:??

Ivanova, Angelina, Stephan Oepen, Lilja Øvrelid & Dan Flickinger. 2012. Who did what to whom?: a contrastive study of syntacto-semantic dependencies. In *Proceedings of the 6th Linguistic Annotation Workshop*, 2–11. http://aclweb.org/anthology/W12-3602.

Johnson, Mark. 1988. *Attribute-value logic and the theory of grammar* (CSLI Lecture Notes 16). Stanford, CA: CSLI Publications.

Joshi, Aravind K. 1987. Introduction to Tree Adjoining Grammar. In Alexis Manaster-Ramer (ed.), *The mathematics of language*, 87–114. Amsterdam: John Benjamins Publishing Co.

Kasami, Tadao, Hiroyuki Seki & Mamoru Fujii. 1989. Generalized context-free grammars and multiple context-free grammars. *Systems and Computers in Japan* 20(7). 43–52.

Kay, Martin. 1963. Rules of interpretation. an approach to the problem of computation in the semantics of natural language. In Cicely M. Popplewell (ed.), *Proceedings of IFIP Congress 62*, 318–21. Amsterdam, The Netherlands: North-Holland Publishing Company.

Kay, Martin. 1973. The MIND system. In R. Rustin (ed.), *Courant Computer Science Symposium 8: natural language processing*. Algorithmics Press.

Kim, Jin-Dong, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano & Jun'ichi Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the workshop on current trends in biomedical natural language processing (BioNLP): shared task*, 1–9. http://aclweb.org/anthology/W09-1401.

King, Tracy Holloway. 2016. Theoretical linguistics and grammar engineering as mutually constraining disciplines. In *Proceedings of the joint 2016 conference on Head-driven Phrase Structure Grammar and Lexical Functional Grammar*, 339–359.

Koenig, Jean-Pierre & Frank Richter. 2019. Semantics. In Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.), *Head-Driven Phrase*

Structure Grammar: The handbook, 653–686. Berlin: Language Science Press. DOI:??

Kouylekov, Milen & Stephan Oepen. 2014. RDF triple stores and a custom SPARQL front-end for indexing and searching (very) large semantic networks. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING), system demonstrations*, 90–94. Dublin, Ireland. http://aclweb.org/anthology/C14-2020.

Kruyt, Johanna G & MWF Dutilh. 1997. A 38 million words Dutch text corpus and its users. *Lexikos* 7. 229–244.

Kubota, Yusuke. 2019. HPSG and Categorial Grammar. In Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.), *Head-Driven Phrase Structure Grammar: The handbook*, 943–999. Berlin: Language Science Press. DOI:??

Kuhnle, Alexander & Ann Copestake. 2018. Deep learning evaluation using deep linguistic processing. In *Proceedings of the workshop on generalization in the age of deep learning*, 17–23. http://aclweb.org/anthology/W18-1003.

Kuthy, Kordula De. 2019. Information structure. In Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.), *Head-Driven Phrase Structure Grammar: The handbook*, 687–720. Berlin: Language Science Press. DOI:??

Letcher, Ned. 2018. *Discovering syntactic phenomena with and within precision grammars*. University of Melbourne dissertation.

MacKinlay, Andrew, David Martinez & Timothy Baldwin. 2012. Detecting modification of biomedical events using a deep parsing approach. *BMC Medical Informatics and Decision Making* 12(Supplement 1). S4. http://www.biomedcentral.com/1472-6947/12/S1/S4/. Proceedings of the ACM Fifth International Workshop on Data and Text Mining in Biomedical Informatics (DTMBio 2011).

Marcus, Mitchell P., Beatrice Santorini & Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics* 19. 313–330.

Marimon, Montserrat. 2013. The Spanish DELPH-IN grammar. *Language Resources and Evaluation* 47(2). 371–397. DOI:10.1007/s10579-012-9199-7

Marimon, Montserrat. 2015. Tibidabo: a syntactically and semantically annotated corpus of Spanish. *Corpora* 10(3). 259–276.

Matsuzaki, Takuya, Yusuke Miyao & Jun'ichi Tsujii. 2007. Efficient HPSG parsing with supertagging and CFG-filtering. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 1671–1676. https://www.aaai.org/Papers/IJCAI/2007/IJCAI07-270.pdf.

Melnik, Nurit. 2007. From "hand-written" to computationally implemented HPSG theories. *Research on Language and Computation* 5(2). 199–236.

Meurers, Walt Detmar, Gerald Penn & Frank Richter. 2002. A web-based instructional platform for constraint-based grammar formalisms and parsing. In Dragomir Radev & Chris Brew (eds.), *Effective tools and methodologies for teaching NLP and CL*, 18–25. Association for Computational Linguistics. Proceedings of the Workshop held at 40th Annual Meeting of the Association for Computational Linguistics. Philadelphia, PA.

Miyao, Yusuke, Takashi Ninomiya & Jun'ichi Tsujii. 2005. Corpus-oriented grammar development for acquiring a Head-Driven Phrase Structure Grammar from the Penn Treebank. In Keh-Yih Su, Oi Yee Kwong, Jn'ichi Tsujii & Jong-Hyeok Lee (eds.), *Natural language processing IJCNLP 2004* (Lecture Notes in Artificial Intelligence 3248), 684–693. Berlin: Springer Verlag.

Miyao, Yusuke, Rune Sætre, Kenji Sagae, Takuya Matsuzaki & Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*. 46–54. http://aclweb.org/anthology/P08-1006.

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

Morante, Roser & Eduardo Blanco. 2012. *SEM 2012 shared task: resolving the scope and focus of negation. In *Proceedings of *SEM 2012: the 1st Joint Conference on Lexical and Computational Semantics*, 265–274. http://aclweb.org/anthology/S12-1035.

Morgado da Costa, Luis, Francis Bond & Xiaoling He. 2016. Syntactic well-formedness diagnosis and error-based coaching in computer assisted language learning using machine translation. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*, 107–116. http://aclweb.org/anthology/W16-4914.

Müller, Stefan. 1996. The Babel-System: An HPSG fragment for German, a parser, and a dialogue component. In *Proceedings of the Fourth International Conference on the Practical Application of Prolog*, 263–277. London.

Müller, Stefan. 1999. *Deutsche Syntax deklarativ: Head-Driven Phrase Structure Grammar für das Deutsche* (Linguistische Arbeiten 394). Tübingen: Max Niemeyer Verlag.

Müller, Stefan. 2004a. Continuous or discontinuous constituents? A comparison between syntactic analyses for constituent order and their processing systems. *Research on Language and Computation, Special Issue on Linguistic Theory and Grammar Implementation* 2(2). 209–257.

Müller, Stefan. 2004b. *Example sentences and making them useful for theoretical and computational linguistics*. Paper presented at the DGfS Jahrestagung: AG Empirische Fundierung der Modellbildung in der Syntax.

Müller, Stefan. 2007. *Head-Driven Phrase Structure Grammar: Eine Einführung*. 1st edn. (Stauffenburg Einführungen 17). Tübingen: Stauffenburg Verlag.

Müller, Stefan. 2009. A Head-Driven Phrase Structure Grammar for Maltese. In Bernard Comrie, Ray Fabri, Beth Hume, Manwel Mifsud, Thomas Stolz & Martine Vanhove (eds.), *Introducing Maltese linguistics: Papers from the 1st International Conference on Maltese Linguistics (Bremen/Germany, 18–20 October, 2007)* (Studies in Language Companion Series 113), 83–112. Amsterdam: John Benjamins Publishing Co.

Müller, Stefan. 2014. Kernigkeit: Anmerkungen zur Kern-Peripherie-Unterscheidung. In Antonio Machicao y Priemer, Andreas Nolda & Athina Sioupi (eds.), *Zwischen Kern und Peripherie* (studia grammatica 76), 25–39. Berlin: de Gruyter.

Müller, Stefan. 2015. The CoreGram project: Theoretical linguistics, theory development and verification. *Journal of Language Modelling* 3(1). 21–86. DOI:10.15398/jlm.v3i1.91

Müller, Stefan. 2019a. Constituent order. In Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.), *Head-Driven Phrase Structure Grammar: The handbook*, 265–308. Berlin: Language Science Press. DOI:??

Müller, Stefan. 2019b. *Grammatical theory: From Transformational Grammar to constraint-based approaches*. 3rd edn. (Textbooks in Language Sciences 1). Berlin: Language Science Press. DOI:10.5281/zenodo.3364215

Müller, Stefan. 2019c. HPSG and Construction Grammar. In Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.), *Head-Driven Phrase Structure Grammar: The handbook*, 1007–1058. Berlin: Language Science Press. DOI:??

Müller, Stefan & Masood Ghayoomi. 2010. PerGram: A TRALE implementation of an HPSG fragment of Persian. In *Proceedings of 2010 IEEE International Multiconference on Computer Science and Information Technology – Computational Linguistics Applications (CLA'10). Wisła, Poland, 18–20 October 2010*, vol. 5, 461–467. Polnish Information Processing Society.

Müller, Stefan & Walter Kasper. 2000. HPSG analysis of German. In Wolfgang Wahlster (ed.), *Verbmobil: Foundations of speech-to-speech translation* (Artificial Intelligence), 238–253. Berlin: Springer Verlag.

Müller, Stefan & Janna Lipenkova. 2013. ChinGram: A TRALE implementation of an HPSG fragment of Mandarin Chinese. In Huei-ling Lai & Kawai Chui (eds.),

Proceedings of the 27th Pacific Asia Conference on Language, Information, and Computation (PACLIC 27), 240–249. Taipei, Taiwan: Department of English, National Chengchi University.

Müller, Stefan & Bjarne Ørsnes. 2013. *Danish in Head-Driven Phrase Structure Grammar* (Empirically Oriented Theoretical Morphology and Syntax). Berlin: Language Science Press. In preparation.

Muszyńska, Ewa. 2016. Graph- and surface-level sentence chunking. In *Proceedings of the ACL 2016 Student Research Workshop*, 93–99. Berlin, Germany: Association for Computational Linguistics. http://anthology.aclweb.org/P16-3014.

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty & Daniel Zeman. 2016. Universal Dependencies v1: a multilingual treebank collection. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk & Stelios Piperidis (eds.), *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. Portorož, Slovenia: European Language Resources Association (ELRA).

Oepen, Stephan. 2001. *[incr tsdb()] — competence and performance laboratory. user manual*. Technical Report. Saarbrücken, Germany: COLI.

Oepen, Stephan & Daniel P. Flickinger. 1998. Towards systematic grammar profiling: Test suite technology ten years after. *Journal of Computer Speech and Language* 12(4). 411–436. http://www.delph-in.net/itsdb/publications/profiling.ps.gz, accessed 2018-2-25. (Special Issue on Evaluation).

Oepen, Stephan, Daniel P. Flickinger, Kristina Toutanova & Christopher D. Manning. 2004. LinGO Redwoods: A rich and dynamic treebank for HPSG. *Research on Language and Computation* 2(4). 575–596.

Oepen, Stephan, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic & Zdenka Uresova. 2015. SemEval 2015 task 18: broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, 915–926. http://aclweb.org/anthology/S15-2153.

Oepen, Stephan, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova & Yi Zhang. 2014. SemEval 2014 task 8: broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 63–72. http://aclweb.org/anthology/S14-2008.

Oepen, Stephan & Jan Tore Lønning. 2006. Discriminant-based MRS banking. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, 1250–1255. http://www.lrec-conf.org/proceedings/lrec2006/pdf/364_pdf.pdf.

Oepen, Stephan, Klaus Netter & Judith Klein. 1997. TSNLP — Test suites for natural language processing. In John Nerbonne (ed.), *Linguistic databases*, 13–36. Stanford, CA: CSLI Publications.

Oepen, Stephan, Lilja Øvrelid, Jari Björne, Richard Johansson, Emanuele Lapponi, Filip Ginter & Erik Velldal. 2017. The 2017 Shared Task on Extrinsic Parser Evaluation: towards a reusable community infrastructure. In *Proceedings of the 2017 shared task on extrinsic parser evaluation, at the 4th International Conference on Dependency Linguistics and the 15th International Conference on Parsing Technologies*, 1–16. http://svn.nlpl.eu/epe/2017/public/proceedings.pdf.

Oepen, Stephan, Erik Velldal, Jan Tore Lønning, Paul Meurer, Victoria Rosén & Dan Flickinger. 2007. Towards hybrid quality-oriented machine translation: On linguistics and probabilities in MT. In *Proceedings of 11th conference on theoretical and methodological issues in machine translation*, 144–153. Skövde, Sweden.

Oostdijk, Nelleke. 2000. The Spoken Dutch Corpus. overview and first evaluation. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC)*. http://lrec-conf.org/proceedings/lrec2000/pdf/110.pdf.

Packard, Woodley. 2015. *Full forest treebanking*. University of Washington MA thesis.

Packard, Woodley, Emily M Bender, Jonathon Read, Stephan Oepen & Rebecca Dridan. 2014. Simple negation scope resolution through deep parsing: a semantic solution to a semantic problem. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL), long papers*, 69–78. http://aclweb.org/anthology/P14-1007.

Penn, Gerald. 2004. Balancing clarity and efficiency in typed feature logic through delaying. In Donia Scott (ed.), *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), main volume*, 239–246. Barcelona, Spain.

Petrick, Stanley Roy. 1965. *A recognition procedure for Transformational Grammars*. Massachusetts Institute of Technology. Dept. of Modern Languages dissertation. http://hdl.handle.net/1721.1/13013, accessed 2018-2-25.

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

Ranta, Aarne. 2009. The GF resource grammar library. *Linguistic Issues in Language Technology* 2. 1–62.

Reape, Mike. 1994. Domain union and word order variation in German. In John Nerbonne, Klaus Netter & Carl J. Pollard (eds.), *German in Head-Driven Phrase Structure Grammar* (CSLI Lecture Notes 46), 151–198. Stanford, CA: CSLI Publications.

Reiplinger, Melanie, Ulrich Schäfer & Magdalena Wolska. 2012. Extracting glossary sentences from scholarly articles: a comparative evaluation of pattern bootstrapping and deep analysis. In *Proceedings of the ACL-2012 special workshop on rediscovering 50 years of discoveries*, 55–65. http://aclweb.org/anthology/W12-3206.

Rohde, Douglas LT. 2005. *Tgrep2 user manual, version 1.15*. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/cmt-55/OldFiles/lti/Courses/722/Spring-08/Penntbank/Tgrep2/tgrep2_manual.pdf.

Schabes, Yves, Anne Abeillé & Aravind K. Joshi. 1988. *Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars*. Technical Report MS-CIS-88-65. University of Pennsylvania Department of Computer & Information Science.

Schäfer, Ulrich, Bernd Kiefer, Christian Spurk, Jörg Steffen & Rui Wang. 2011. The ACL anthology searchbench. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL), systems demonstrations*, 7–13. http://aclweb.org/anthology/P11-4002.

Schäfer, Ulrich, Hans Uszkoreit, Christian Federmann, Torsten Marek & Yajing Zhang. 2008. Extracting and querying relations in scientific papers. In Andreas R. Dengel, Karsten Berns, Thomas M. Breuel, Frank Bomarius & Thomas R. Roth-Berghofer (eds.), *KI 2008: advances in artificial intelligence*, 127–134. Berlin, Heidelberg: Springer.

Schuster, Sebastian, Éric Villemonte de La Clergerie, Marie Candito, Benoît Sagot, Christopher Manning & Djamé Seddah. 2017. Paris and Stanford at EPE 2017: downstream evaluation of graph-based dependency representations. In *Proceedings of the 2017 shared task on extrinsic parser evaluation, at the 4th International Conference on Dependency Linguistics and the 15th International Conference on Parsing Technologies*, 47–59. http://svn.nlpl.eu/epe/2017/public/proceedings.pdf.

Siegel, Melanie, Emily M. Bender & Francis Bond. 2016. *Jacy: An implemented grammar of Japanese* (CSLI Studies in Computational Linguistics). Stanford, CA: CSLI Publications.

Slayden, Glenn C. 2012. *Array TFS storage for unification grammars*. University of Washington MA thesis.

Solberg, Lars Jørgen. 2012. *A corpus builder for Wikipedia*. University of Oslo MA thesis. https://www.duo.uio.no/bitstream/handle/10852/34914/thesis.pdf.

Stabler, Edward. 1997. Derivational minimalism. In C. Retoré (ed.), *Logical aspects of computational linguistics* (Lecture Notes in Computer Science 1328), 68–95. Berlin: Springer Verlag.

Steedman, Mark & Jason Baldridge. 2011. Combinatory Categorial Grammar. In Robert D. Borsley & Kersti Börjars (eds.), *Non-transformational syntax: Formal and explicit models of grammar: A guide to current models*, 181–224. Oxford, UK/Cambridge, MA: Blackwell Publishers Ltd.

Suppes, Patrick, Tie Liang, Elizabeth E Macken & Daniel P Flickinger. 2014. Positive technological and negative pre-test-score effects in a four-year assessment of low socioeconomic status K-8 student learning in computer-based math and language arts courses. *Computers & Education* 71. 23–32.

Sygal, Yael & Shuly Wintner. 2011. Towards modular development of typed unification grammars. *Computational Linguistics* 37. 29–74.

Torr, John, Milos Stanojevic, Mark Steedman & Shay B. Cohen. 2019. Wide-coverage neural A* parsing for minimalist grammars. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2486–2505. Florence, Italy: Association for Computational Linguistics. https://www.aclweb.org/anthology/P19-1238.

Toutanova, Kristina, Christopher D. Manning, Dan Flickinger & Stephan Oepen. 2005. Stochastic HPSG parse disambiguation using the Redwoods corpus. *Research on Language & Computation* 3(1). 83–105.

Toutanova, Kristina, Christopher D. Manning, Stuart M. Shieber, Dan Flickinger & Stephan Oepen. 2002. Parse disambiguation for a rich HPSG grammar. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories (TLT)*. Sozopol, Bulgaria. http://bultreebank.org/wp-content/uploads/2017/05/paper17.pdf.

Tsuruoka, Yoshimasa, Yusuke Miyao & Jun'ichi Tsujii. 2004. Towards efficient probabilistic HPSG parsing: integrating semantic and syntactic preference to guide the parsing. In *Proceedings of the IJCNLP workshop beyond shallow analyses: formalisms and statistical modeling for deep analyses*.

van der Beek, Leonoor, Gosse Bouma, Rob Malouf & Gertjan van Noord. 2002. The Alpino Dependency Treebank. In *Computational Linguistics in the Netherlands 2001: selected papers from the twelfth CLIN meeting* (Language and Computers 45), 8–22. Rodopi.

van Noord, Gertjan. 2006. At last parsing is now operational. In *Actes de la 13ème conf'erence sur le Traitement Automatique des Langues Naturelles (TALN)*, 20–42. http://talnarchives.atala.org/TALN/TALN-2006/taln-2006-invite-002.pdf.

van Noord, Gertjan, Gosse Bouma, Frank van Eynde, Daniël de Kok, Jelmer van der Linde, Ineke Schuurman, Erik Tjong Kim Sang & Vincent Vandeghinste. 2013. Large scale syntactic annotation of written Dutch: Lassy. In Peter Spyns & Jan Odijk (eds.), *Essential speech and language technology for Dutch: results by the STEVIN programme*, 147–164. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-30910-6_9. DOI:10.1007/978-3-642-30910-6_9

van Noord, Gertjan & Robert Malouf. 2005. Wide coverage parsing with stochastic attribute value grammars. Unpublished draft. An earlier version was presented at the IJCNLP workshop *B*eyond Shallow Analyses: Formalisms and statistical modeling for deep analyses.

Velldal, Erik. 2009. *Empirical realization ranking*. University of Oslo, Department of Informatics dissertation.

Velldal, Erik, Lilja Øvrelid, Jonathon Read & Stephan Oepen. 2012. Speculation and negation: rules, rankers, and the role of syntax. *Computational Linguistics* 38(2). 369–410. https://www.mitpressjournals.org/doi/full/10.1162/COLI_a_00126.

Wahlster, Wolfgang (ed.). 2000. *Verbmobil: Foundations of speech-to-speech translation* (Artificial Intelligence). Berlin: Springer Verlag.

Waldron, Benjamin, Ann Copestake, Ulrich Schäfer & Bernd Kiefer. 2006. Preprocessing and tokenisation standards in DELPH-IN tools. In *Proceedings of the fifth International Conference on Language Resources and Evaluation (LREC)*. Genoa, Italy: European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2006/pdf/214_pdf.pdf.

Wax, David. 2014. *Automated grammar engineering for verbal morphology*. University of Washington MA thesis.

Wechsler, Stephen, Jean-Pierre Koenig & Anthony Davis. 2019. Argument structure and linking. In Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.), *Head-Driven Phrase Structure Grammar: The handbook*, 223–263. Berlin: Language Science Press. DOI:??

Xia, Fei & William D. Lewis. 2007. Multilingual structural projection across interlinear text. In *Proceedings of the 6th Conference of the North American Chapter of the Associaton for Computational Linguistics (NAACL), long papers*, 452–459. Rochester, New York.

Ytrestøl, Gisle, Dan Flickinger & Stephan Oepen. 2009. Extracting and annotating Wikipedia sub-domains: towards a new eScience community resource. In *Pro-*

ceedings of the 7th International Workshop on Treebanks and Linguistic Theories (TLT), 185–197. https://dspace.library.uu.nl/handle/1874/296811.

Zamaraeva, Olga, Kristen Howell & Emily M. Bender. 2019. Handling cross-cutting properties in automatic inference of lexical classes: a case study of Chintang. In *Proceedings of the 3rd Workshop on the Use of Computational Methods in the Study of Endangered Languages (ComputEL)*.

Zamaraeva, Olga, Kristen Howell & Adam Rhine. 2018. Improving feature extraction for pathology reports with precise negation scope detection. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, 3564–3575. http://aclweb.org/anthology/C18-1302.

Zamaraeva, Olga, František Kratochvíl, Emily M Bender, Fei Xia & Kristen Howell. 2017. Computational support for finding word classes: a case study of Abui. In *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages (ComputEL)*, 130–140.

Zhang, Yi & Hans-Ulrich Krieger. 2011. Large-scale corpus-driven PCFG approximation of an HPSG. In *Proceedings of the 12th International Conference on Parsing Technologies*, 198–208. Dublin, Ireland: Association for Computational Linguistics. http://aclweb.org/anthology/W11-2923.

Zwicky, Arnold M., Joyce Friedman, Barbara C. Hall & Donald E. Walker. 1965. The MITRE syntactic analysis procedure for Transformational Grammars. In *Proceedings – FALL Joint Computer Conference*, 317–326. DOI:10.1109/AFIPS.1965.108