

# 20 years of the Grammar Matrix: cross-linguistic hypothesis testing of increasingly complex interactions

Olga Zamaraeva<sup>1</sup>, Chris Curtis<sup>2</sup>, Guy Emerson<sup>3</sup>, Antske Fokkens<sup>4,5</sup>,  
Michael Wayne Goodman<sup>6</sup>, Kristen Howell<sup>6</sup>, T.J. Trimble,  
and Emily M. Bender<sup>7</sup>

<sup>1</sup> Universidade da Coruña

<sup>2</sup> Firemuse Research

<sup>3</sup> University of Cambridge

<sup>4</sup> Vrije Universiteit Amsterdam

<sup>5</sup> Eindhoven University of Technology

<sup>6</sup> LivePerson Inc.

<sup>7</sup> University of Washington

## ABSTRACT

The Grammar Matrix project is a meta-grammar engineering framework expressed in Head-driven Phrase Structure Grammar (HPSG) and Minimal Recursion Semantics (MRS). It automates grammar implementation and is thus a tool and a resource for linguistic hypothesis testing at scale. In this paper, we summarize how the Grammar Matrix grew in the last decade and describe how new additions to the system have made it possible to study interactions between analyses, both monolingually and cross-linguistically, at new levels of complexity.

*Keywords: HPSG, grammar engineering, hypothesis testing, typology*

## INTRODUCTION

1

From its beginnings in 2001, the Grammar Matrix project (Bender *et al.* 2002, 2010, among others)<sup>1</sup> has investigated how grammar engineer-

---

<sup>1</sup> Olga Zamaraeva contributed the constituent questions and the clausal complements libraries to the Grammar Matrix framework and led the writing of this

ing can support research into cross-linguistic variation and similarity. Key to this approach has been the potential of computational implementation to handle complexity, in both data and analyses. In this paper we take stock of work on and with the Grammar Matrix since 2010, explore how that potential has been leveraged and envision future directions.

The Grammar Matrix is a meta-grammar engineering framework expressed in the Head-driven Phrase Structure Grammar formalism (HPSG; Pollard and Sag 1994), specifically in one particular restricted version of it (Copestake 2002a). Grammar engineering is a discipline and a methodology concerned with a particular empirical approach to modelled linguistic knowledge (Bierwisch 1963; Zwicky *et al.* 1965; Müller 1999; Butt *et al.* 1999; Bender 2008; Müller 2015): namely, grammar *modelling* and *testing*. *Modelling* grammar in this context means coming up with sets of grammar entities (in this case: types, rules and lexical entries) and implementing them as a computer program which can accept or reject strings by attempting (and either succeeding or failing) to find a syntactic structure that can correspond to the input string. *Testing* (analogous to “competence profiling” as defined by Oepen (2002, page 89)) means deploying this grammar program (usually along with a separate parser program) on a list of sentences and then assessing whether or not the grammar indeed *correctly* parsed all grammatical sentences and rejected all the ungrammatical ones – an alternative to doing the testing with pen and paper, performing computations in one’s head. *Correctly* here means that each structure assigned by the grammar to any grammatical string is in fact a correct linguistic representation of it. For the purposes of the

---

paper. Emily M. Bender is the initial developer and the long-time lead of the Grammar Matrix project, and the principal investigator of the National Science Foundation grants for both the Grammar Matrix and AGGREGATION projects. The rest of the authors are in alphabetical order. Chris Curtis contributed the valence change library to the project; Guy Emerson is the author of append-lists and computation types; Antske Fokkens is the author of CLIMB and contributed to the word order library; Michael W. Goodman contributed to the morphotactics library and to the regression test system; Kristen Howell contributed the clausal modifiers and the nominalized clauses libraries and contributed to the AGGREGATION project. T. J. Trimble added the support for adjectives and copulas and made other contributions to the lexicon component of the Grammar Matrix.

Grammar Matrix, this last assessment is done with respect not to the syntactic tree but to the resulting sentence semantics that is directly paired with the syntactic structure. In our case, the semantics is encoded in the Minimal Recursion Semantics formalism (MRS; Copestake *et al.* 2005). In terms of practical set up, the Grammar Matrix allows the linguist-user to enter a typological, lexical, and morphological description of a grammar via a web-based questionnaire, and obtain a grammar fragment implemented in HPSG automatically. This, in turn, allows this linguist to test the set of hypotheses that the grammar encodes against the data stored in text files, in a computer-aided fashion.

We see the Grammar Matrix as a flexible framework for building up, over time and in a data-driven fashion, a set of analyses which are demonstrably useful for describing the repertoire of grammatical variation in the world's languages. Our conviction is rooted in three properties of the framework: (i) the Grammar Matrix design is informed by typological literature (while relying on established HPSG concepts); (ii) the development methodology prioritizes cross-linguistic applicability of the analyses and as such leaves flexibility to define HPSG features motivated by the data; (iii) for any new analyses proposed for inclusion in the Grammar Matrix, there is a system in place which allows one to automatically test the new analyses in integration with the existing ones (Bender *et al.* 2007). Long term, this builds and extends a system of analyses for which there is a demonstrated area of applicability – which also grows over time.

The paper is structured as follows. In Section 2, we describe the syntactic formalism which the Grammar Matrix uses and the grammar engineering philosophy which it follows. Section 3 gives a summary of the additions to the Grammar Matrix since 2010 and describes the development methodologies, including a detailed description of how the analyses which are part of the Grammar Matrix are being tested as a holistic system. In Section 4, we discuss several specific examples of how the Grammar Matrix helps identify tensions between different analyses, while Section 5 gives examples of the analyses which have proven particularly robust over the years. The paper concludes with the discussion of how the Grammar Matrix serves as infrastructure to support other research projects (Section 6) and a look ahead to future directions (Section 7).

This section is intended to give the reader basic background on HPSG (Section 2.1) and grammar engineering (Section 2.2), and on the particular version of the HPSG formalism that the Grammar Matrix project uses (Section 2.3). Section 2.3 also describes several major grammar engineering projects and initiatives, some using HPSG and some using other formalisms.

Head-driven Phrase Structure Grammar (Pollard and Sag 1994) is a syntactic framework characterized by a sign-based approach to grammar,<sup>2</sup> the use of formally precise constraint-based formalisms (Carpenter 2005), and an emphasis (shared with Construction Grammar (Fillmore *et al.* 1988) and especially Sign-Based Construction Grammar (Sag *et al.* 2012)) on modelling both the broad generalizations at play in a given language and the rich details of lexical and constructional idiosyncrasy in a single, coherent grammar. Multiple inheritance hierarchies serve as the central device to capture generalizations in HPSG. Like any grammatical framework, HPSG encompasses a variety of related theoretical proposals and also has multiple competing formalisms (for some discussion, see Richter 2021). Despite this variety, the formalisms used in HPSG are relatively stable over time, making possible the development of software which implements those formalisms and can be used for sustained grammar development (for further discussion, see Bender and Emerson 2021).

HPSG formalisms are based on constraint unification. In constraint unification, variables may be constrained to have a particular value or to be equal to the value of another variable. In order for any two types to unify, there must be a single (unique) type in the hierarchy which represents their combination (Copestake 2002b, page 42). Ultimately, an HPSG parser checks whether, given a tokenized sentence string and a set of types and lexical entries that represent the

---

<sup>2</sup>See e.g. Pollard and Sag 1987, page 2 for a summary of de Saussure's (1916) theory of language signs.

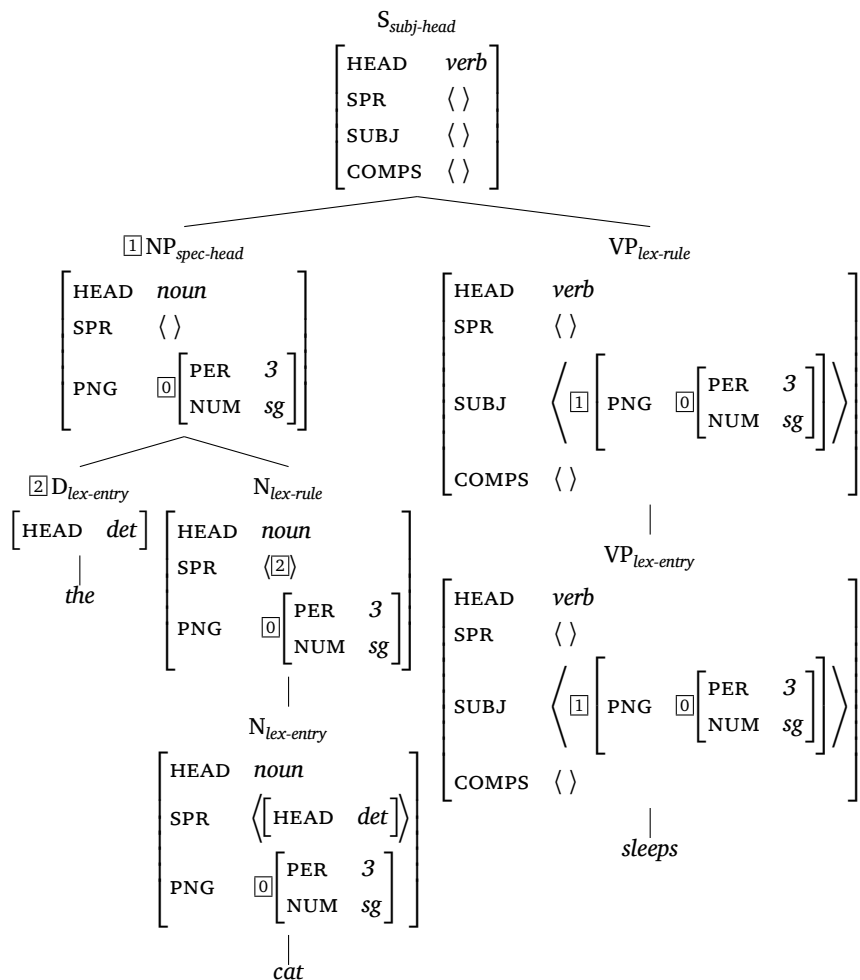


Figure 1: An HPSG derivation visualized as a tree

grammar, it can find a feature structure such that each token corresponds to some lexical entry and together they form some syntactic structure in which all the constraints dictated by the grammar unify.

As a simplified example, consider a tree of feature structures in Figure 1 representing an HPSG parse for the English [eng] (Indo-European) sentence (1).<sup>3</sup> The feature structures in the tree are visu-

<sup>3</sup>This tree is a simplified version of a tree produced by a grammar of English that was output by the Grammar Matrix. In particular, we only show the features

alized as attribute-value matrices (AVMs). This tree includes only selected feature-value pairs and substructure sharing tags ([@] etc.), to illustrate the particular role of each node in the tree that we would like to emphasize here for purposes of exposition.

(1) The cat sleeps. [eng]

Consider the tree in Figure 1 bottom-up, along with related examples (2) and (3) located on page 55. Suppose that the terminal nodes in the tree (corresponding to the lexemes *the*, *cat*, and *sleep*) are provided by some lexicon. The lexical entries in that lexicon are instances of lexical types and specify, among other things, the syntactic category of each word (such as noun or verb) and what arguments they require, if any. For example, the intransitive verb *sleep* requires exactly zero complements and one subject element; furthermore, it requires an NP subject. The noun *cat* has a PNG feature which in turn has PER and NUM features appropriate for it, the values of which in this particular lexical entry are specified to be PER 3 at the lexical entry level, NUM underspecified to just *number* in the lexical type to which the lexical entry belongs (2), and further specified to NUM *sg* after a lexical rule (3) applies. The SYNSEM feature in the lexical rule (3) is the “mother” of the unary rule; the DTR feature is the “daughter”.<sup>4</sup> Note that while the NUM value is identified between the mother and the daughter in the lexical rule, a lexical entry like (2) can unify with the daughter of (3) because its own value is underspecified. In the fully specified tree in Figure 1, the NUM has already been identified with the mother’s NUM in the lexical rule (same with the SUBJ identity between the verb’s lexical entry node and the verb’s lexical rule node).

---

HEAD, SPR (specifier), SUBJ (subject), COMPS (complements), and PNG (person, number, gender), with only NUM and PER within the last of these (whereas in reality, there is also GEN). The tree and the explanation are adapted from Zamaraeva 2021a, page 33.

<sup>4</sup>This lexical rule does not have an overt grammatical marking (these are sometimes called “zero-marking” rules); a lexical rule for plural marking would add the affix *s* to the orthography.

(2)	<table style="border-collapse: collapse; width: 100%;"> <tr> <td colspan="2"><i>noun-lex</i></td> </tr> <tr> <td style="padding-right: 10px;">HEAD</td> <td><i>noun</i></td> </tr> <tr> <td style="padding-right: 10px;">PNG</td> <td> <table style="border-collapse: collapse;"> <tr> <td style="padding-right: 5px;">PER</td> <td>3</td> </tr> <tr> <td style="padding-right: 5px;">NUM</td> <td><i>number</i></td> </tr> </table> </td> </tr> <tr> <td style="padding-right: 10px;">SPR</td> <td> <table style="border-collapse: collapse;"> <tr> <td style="padding-right: 5px;">HEAD</td> <td><i>det</i></td> </tr> </table> </td> </tr> </table>	<i>noun-lex</i>		HEAD	<i>noun</i>	PNG	<table style="border-collapse: collapse;"> <tr> <td style="padding-right: 5px;">PER</td> <td>3</td> </tr> <tr> <td style="padding-right: 5px;">NUM</td> <td><i>number</i></td> </tr> </table>	PER	3	NUM	<i>number</i>	SPR	<table style="border-collapse: collapse;"> <tr> <td style="padding-right: 5px;">HEAD</td> <td><i>det</i></td> </tr> </table>	HEAD	<i>det</i>	(3)	<table style="border-collapse: collapse; width: 100%;"> <tr> <td colspan="2"><i>sg-lex-rule</i></td> </tr> <tr> <td style="padding-right: 10px;">SYNSEM</td> <td> <table style="border-collapse: collapse;"> <tr> <td style="padding-right: 10px;">ORTH</td> <td>0</td> </tr> <tr> <td style="padding-right: 10px;">HEAD</td> <td><i>noun</i></td> </tr> <tr> <td style="padding-right: 10px;">PNG NUM</td> <td>1 <i>sg</i></td> </tr> </table> </td> </tr> <tr> <td style="padding-right: 10px;">DTR</td> <td> <table style="border-collapse: collapse;"> <tr> <td style="padding-right: 10px;">ORTH</td> <td>0</td> </tr> <tr> <td style="padding-right: 10px;">HEAD</td> <td><i>noun</i></td> </tr> <tr> <td style="padding-right: 10px;">PNG NUM</td> <td>1</td> </tr> </table> </td> </tr> </table>	<i>sg-lex-rule</i>		SYNSEM	<table style="border-collapse: collapse;"> <tr> <td style="padding-right: 10px;">ORTH</td> <td>0</td> </tr> <tr> <td style="padding-right: 10px;">HEAD</td> <td><i>noun</i></td> </tr> <tr> <td style="padding-right: 10px;">PNG NUM</td> <td>1 <i>sg</i></td> </tr> </table>	ORTH	0	HEAD	<i>noun</i>	PNG NUM	1  <i>sg</i>	DTR	<table style="border-collapse: collapse;"> <tr> <td style="padding-right: 10px;">ORTH</td> <td>0</td> </tr> <tr> <td style="padding-right: 10px;">HEAD</td> <td><i>noun</i></td> </tr> <tr> <td style="padding-right: 10px;">PNG NUM</td> <td>1</td> </tr> </table>	ORTH	0	HEAD	<i>noun</i>	PNG NUM	1
<i>noun-lex</i>																																			
HEAD	<i>noun</i>																																		
PNG	<table style="border-collapse: collapse;"> <tr> <td style="padding-right: 5px;">PER</td> <td>3</td> </tr> <tr> <td style="padding-right: 5px;">NUM</td> <td><i>number</i></td> </tr> </table>	PER	3	NUM	<i>number</i>																														
PER	3																																		
NUM	<i>number</i>																																		
SPR	<table style="border-collapse: collapse;"> <tr> <td style="padding-right: 5px;">HEAD</td> <td><i>det</i></td> </tr> </table>	HEAD	<i>det</i>																																
HEAD	<i>det</i>																																		
<i>sg-lex-rule</i>																																			
SYNSEM	<table style="border-collapse: collapse;"> <tr> <td style="padding-right: 10px;">ORTH</td> <td>0</td> </tr> <tr> <td style="padding-right: 10px;">HEAD</td> <td><i>noun</i></td> </tr> <tr> <td style="padding-right: 10px;">PNG NUM</td> <td>1 <i>sg</i></td> </tr> </table>	ORTH	0	HEAD	<i>noun</i>	PNG NUM	1  <i>sg</i>																												
ORTH	0																																		
HEAD	<i>noun</i>																																		
PNG NUM	1  <i>sg</i>																																		
DTR	<table style="border-collapse: collapse;"> <tr> <td style="padding-right: 10px;">ORTH</td> <td>0</td> </tr> <tr> <td style="padding-right: 10px;">HEAD</td> <td><i>noun</i></td> </tr> <tr> <td style="padding-right: 10px;">PNG NUM</td> <td>1</td> </tr> </table>	ORTH	0	HEAD	<i>noun</i>	PNG NUM	1																												
ORTH	0																																		
HEAD	<i>noun</i>																																		
PNG NUM	1																																		

All types in the grammar are part of the type hierarchy, with subtypes inheriting all constraints of their supertypes. Each type is specified to have features appropriate for it, and each subtype of a type may set the values for those features (which in turn are constrained to be specific types). HPSG uses the type hierarchy to define feature appropriateness, to constrain which feature structures can unify with each other, and to capture generalizations. In that final function, the type hierarchy supports compactness and elegance and thus maintainability and scalability of grammars.

HPSG theory is characterized by rich lexical types and relatively schematic phrase structure rules. The properties of any given node in a tree are established by combining constraints from lexical entries and rules, including constraints which propagate information through the tree. When information is identified between different parts of feature structure or tree, this is called *structure sharing*. One example is the Head Feature Principle (Pollard and Sag 1994, page 31), which stipulates that the value of the feature HEAD (including all feature-value pairs inside HEAD) in a phrase licensed by a headed rule must be shared between the mother and the head daughter. Accordingly, for a phrase structure rule, the grammarian must indicate if it is a headed rule and, if so, which daughter is the head daughter. In Figure 1, the HEAD category is propagated because the Head Feature Principle is implemented in the grammar. Other information is propagated because the particular phrase structure or lexical rules are defined specifically to do that; for example, the head-specifier rule identifies the non-head daughter with the sole element on the SPR list of the head daughter in Figure 1, and so on.

Structure sharing means some parts of the structure are the same. Any feature structure can also be visualized as a graph (see Pollard and

Sag 1994, pages 16–17), and indeed graph data structures provide a convenient and frequently used implementation of feature structures. In such cases, identity tags in the AVM notation correspond to reentrancies in the graph, meaning the arcs will converge in the exact same place. In other words, in Figure 1 the subject of the head daughter and the entire non-head daughter are not only similar (identical); they are literally the same structure.

The notion of structure-sharing is closely related to the notion of constraint unification generally, and to *unification failure*, which is the mechanism which leads to HPSG grammars rejecting ungrammatical input, or in other words not generating ungrammatical strings. Suppose the same HPSG grammar that licenses sentence (1) by assigning it the structure in Figure 1 is given the string (4) as input instead.

(4) \*The cats sleeps.

In order for the grammar to license the plural orthography *cats*, the instance of the lexical entry for *cat* had to go through a lexical rule which specifies its value as *pl*. This means that if the grammar attempts to use the head-subject rule to license (4), there will be a unification failure between the verb's expected subject's PNG value and the one specified for the noun phrase, as illustrated in Figure 2.

Finally, one other thing about the HPSG formalism that is important for understanding this paper is the notion of *list*, seen in Figure 1 as the value type for SPR, SUBJ, and COMPS (specifier, subject, and complement lists; the list notation being the angle brackets  $\langle \ \rangle$ ). List is a type in the type hierarchy, just like everything else. Lists are convenient for modelling different parts of grammar, most notably the notion of children of a node in the tree, and also arguments (e.g. of a verb).

## 2.2

### *Grammar engineering*

Grammar engineering is the implementation of formal precision grammars as computer programs such that parsing and generation can be done automatically. Precision grammars are machine-readable models of language which encode notions of grammaticality and linguistic knowledge. The concept of precision grammar engineering arises



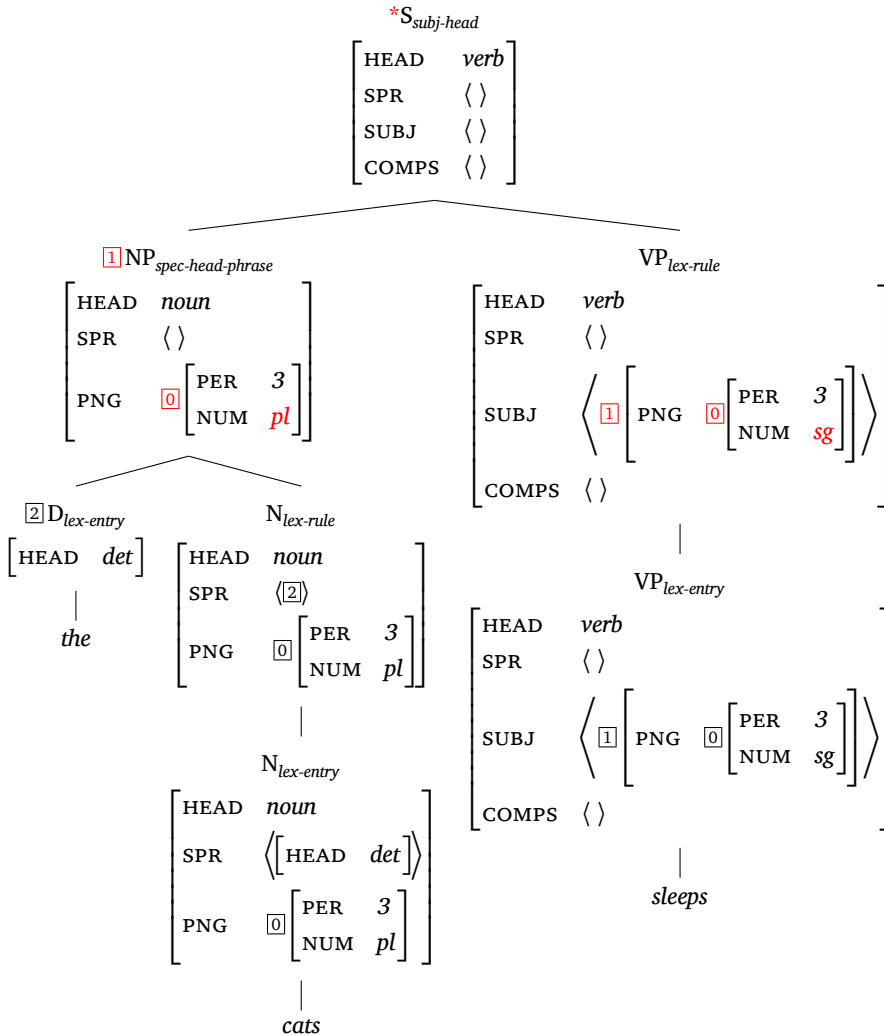


Figure 2: Unification failures (red) visualized as an impossible derivation tree. The asterisk (\*) in the top node signifies that this node is impossible given the constraints

naturally from the idea that modelling grammar is akin to writing a computer program that accepts or rejects strings. An important characteristic of a grammar engineering system is rigor: it actually implements the grammar-program idea on the computer, precluding human mistakes that are due to e.g. human operational memory constraints or inconsistency of attention. It was suggested at least as

early as in Bierwisch 1963 that, without computational aid, the task of tracking how exactly multiple complex analyses interact with each other (and therefore how exactly even a small change in an analysis affects the grammar) becomes virtually impossible.<sup>5</sup>

One of the biggest benefits of grammar engineering projects like the Grammar Matrix, the focus of this paper, is that they allow us to empirically test syntactic theories by creating explicit models of them on a computer and then deploying those models on test suites of data from human languages. Engineered grammars make it much harder for grammarians to fool themselves into thinking that the grammar (a set of syntactic analyses) covers something it actually does not cover. The computer will definitively show which strings from the test set are parsed and which are not, and the grammarians will then be left with the task of investigating any failures. Conversely, at any point the grammarians can be confident in stating that the grammar covers a specific set of strings, namely the ones in the test suites which the grammar actually parsed. A complete system of analyses covering the entire set of human languages remains a very distant goal, and the field proceeds towards it in steps, carefully documenting issues along the way. This can thus be seen as a practical implementation of the Montagovian method of fragments (Montague 1974; Partee 1979; Gazdar *et al.* 1985).

The grammar engineering landscape includes multiple projects carried out in various formalisms. The Grammar Matrix is expressed in one particular version of HPSG developed by the DEep Linguistic Processing with Hpsg INitiative (DELPH-IN, Section 2.3). In addition to DELPH-IN projects, there are other implementations based on the ideas of HPSG, including PAGE (later DISCO) (Uszkoreit *et al.* 1994), ALE (Penn 2000) and its successor TRALE (Meurers *et al.* 2002; Penn 2004; Müller 2007), LIGHT (Ciortuz 2002; Ciortuz and Saveluc 2012), Alpino (Bouma *et al.* 2001b; van Noord 2006, focussing on Dutch), and Enju (Miyao and Tsujii 2008, focussing on probabilistic disambiguation). A grammar engineering project similar in some ways to the Grammar Matrix, called CoreGram (Müller 2015), uses

---

<sup>5</sup>See Fokkens 2014, page 13 for a discussion of Bierwisch 1963 in English and Müller 2015, page 34 for similar discussion and for excerpts translated from German into English.

TRALE's version of HPSG. In Lexical Functional Grammar (LFG; Kaplan and Bresnan 1982), ParGram (Butt and King 2002) is an analogous project. In Minimalism, there is implementation work associated with the strongly lexicalized version of the formalism introduced by Stabler (1997), e.g. Graf and Kostyszyn 2021 and Torr 2018. Candito (1999) proposes a metagrammar for creating French and Italian grammars using the Lexicalized Tree-Adjoining Grammar (LTAG) formalism (Joshi and Schabes 1997). This approach was further developed into the eXtensible MetaGrammar (Crabbé *et al.* 2013, XMG). Clément and Kinyon (2003) propose a metagrammar for generating LFG grammars, inspired by Candito's work. Ranta (2011) implements complex syntactic structures in the multilingual Grammatical Framework Resource Grammar Library. This resource supports the development of grammars for natural language processing (NLP) applications that consist of simple rules that inherit the more complex foundations of the Resource Grammar Library. OpenCCG (Baldrige *et al.* 2007) provides a grammar engineering framework for Combinatory Categorical Grammar (Steedman 2000).<sup>6</sup>

### *DELPH-IN consortium and formalism*

2.3

DELPH-IN<sup>7</sup> is an international consortium of researchers interested in developing implemented grammars with HPSG and MRS and deploying them in the context of practical applications. DELPH-IN produces software support for grammar engineering, grammars, and applications built on grammars, all of which are open source. The software support includes grammar development environments (of which the most widely used is the LKB (Copestake 2002b)), parsing and/or generation engines (the LKB, as well as PET (Callmeier 2000), agree (Slayden 2012), and ACE (Crysmann and Packard 2012)), treebanking

---

<sup>6</sup>Perhaps the strongest current influence of grammar engineering on the rest of the field of NLP is through treebanks. Treebanks are collections of syntactically annotated corpora on which machine learning systems can train. All treebanks were initially either produced by manual annotation, with annotators relying on a linguistic formalism, or using an engineered grammar and manual parse selection.

<sup>7</sup><http://www.delph-in.net>, <https://github.com/delph-in/>

platforms (Oepen *et al.* 2004; Packard 2015),<sup>8</sup> grammar coverage and efficiency profiling facilities (Oepen 2002), Python libraries for a wide variety of data manipulation tasks (Goodman 2019), and the Grammar Matrix meta-grammar engineering toolkit (Bender *et al.* 2002, 2010) which is the focus of this paper.

By far the largest DELPH-IN grammar is the English Resource Grammar (ERG; Flickinger 2000, 2011), but DELPH-IN work has been multilingual from the consortium’s inception in 2002, and the original motivation of the Grammar Matrix was to support the development of grammars for many languages which are interoperable with the same grammar development and application software (Bender *et al.* 2002). Applications developed with DELPH-IN grammatical resources include machine translation (e.g. Oepen *et al.* 2007; Bond *et al.* 2011), computer-assisted language learning (Flickinger and Yu 2013; Suppes *et al.* 2014; Morgado da Costa *et al.* 2016, 2020), and summarization (Fang *et al.* 2016). For further discussion of applications, see Bender and Emerson 2021, Section 4.2.

Important to the success of the DELPH-IN international consortium is the coordination at the level of formalisms. The particular variant of the typed-feature structure formalism used in DELPH-IN (Copestake 2002a) is dubbed the DELPH-IN Joint Reference Formalism (DELPH-IN JRF) and builds on Type Description Language (TDL; Krieger and Schäfer 1994) as its predecessor. A key design decision in the DELPH-IN JRF is to keep the formalism simple by disallowing e.g. set-valued and disjunctive features as well as relational constraints.<sup>9</sup> These restrictions ensure that unification in any grammar will yield a unique well-formed feature structure (if it exists) (Copestake 2002a, page 230), reducing parsing and generation to well-formed unification and allowing for efficient algorithms leading to faster processing times.<sup>10</sup>

---

<sup>8</sup>In the context of precision grammars, treebanking refers to manually selecting and storing the linguistically correct tree(s) from the “forest” of all trees provided for a sentence by the grammar and the parser.

<sup>9</sup>Relational means the value of a feature may be constrained to be the result of an operation over some other features’ values.

<sup>10</sup>For example, eliminating feature value disjunctions in favour of explicit encoding via underspecified types preserves generality (Flickinger 2000, pages 18–24) while allowing unification methods to be optimized to simplify feature

Summary

2.4

This section described the research and engineering landscape in which the Grammar Matrix exists and is being developed. The Grammar Matrix uses the HPSG theory of syntax (Pollard and Sag 1994; Müller *et al.* 2021) with a deliberately restricted version of the formalism (Copestake 2002a) and Minimal Recursion Semantics for semantic representations (Copestake *et al.* 2005). Generally, Grammar Matrix developers subscribe to the grammar engineering philosophy based on the Montagovian method of fragments (Montague 1974) and are accumulating a complex cross-linguistic system of grammatical analyses while maintaining empirical rigor.

THE GRAMMAR MATRIX: TWO DECADES  
OF CONTINUOUS DEVELOPMENT  
AND RESEARCH

3

The Grammar Matrix (Bender *et al.* 2002, 2010)<sup>11</sup> is a DELPH-IN-based meta-grammar engineering framework that includes a web-based questionnaire,<sup>12</sup> a core HPSG grammar, and a grammar customization system programmed in Python.<sup>13</sup> A user fills out a questionnaire with typological, lexical, and morphological information about a language, and, based on the particular combination of their choices, the system applies the customization logic to output a grammar fragment which includes the core as well as additional, custom types, custom lexical entries, and custom rules. This grammar can be used to parse and generate sentences from the language described through the questionnaire. One of the main goals of the Grammar Matrix project is rigor in grammatical hypothesis testing; the system makes more explicit the relationship between a grammar description,

---

structure subsumption and equality checks (Malouf *et al.* 2002, pages 114–122).

<sup>11</sup> <https://github.com/delph-in/matrix#readme>

<sup>12</sup> <http://www.delph-in.net/matrix/customize/matrix.cgi>

<sup>13</sup> <https://www.python.org/>

or hypothesis, and the actual data from the language for which the description is intended.

The Grammar Matrix has been in active development for two decades; the original version, documented in Bender *et al.* 2002, was developed in late 2001. In that work, Bender *et al.* selected portions of the ERG (Flickinger 2000) which they believed would be useful cross-linguistically and put them together in the first version of the Grammar Matrix. The idea was that for a new grammar, this core distilled from the ERG could be included as a foundation, eliminating the need to write the grammar from scratch. Later, it was observed that some portions of the core lexical types and phrase structure rules could be customized to accommodate various typological profiles. This led to future iterations of the Grammar Matrix project which include the customization system (Bender and Flickinger 2005; Drellishak and Bender 2005; Drellishak 2009; Bender *et al.* 2010), which has been used as a starting point for a number of grammars (described in Section 6.3). The main purpose of the customization system is to automate the mapping between a language's typological profile and a particular set of lexical and phrasal HPSG types which serves this typological profile. As such, the Grammar Matrix is a research framework which aims to combine typological breadth with formal-syntactic depth (Bender *et al.* 2010). The relationship between the core and the customization system is such that it can be refined over time, as support for more and more syntactic phenomena is added for more and more typological profiles. For example, once newly considered data makes it obvious that something in the Grammar Matrix core retains any Indo-European (or specifically English) biases, the constraints representing those biases can be removed from the core and added instead to the customization system.<sup>14</sup> Conversely, features and types can be added to the core when a general analysis is developed that is alternative to the one in the ERG.<sup>15</sup>

---

<sup>14</sup>For example, Trimble (2014, pages 60–67) moved all copula types and most adjective types to the customization system to account for languages without copulas and various adjectival phenomena – primarily switching and constrained argument agreement – that required significant reworking of the ERG's analysis.

<sup>15</sup>For example, Zamaraeva (2021a, pages 168–169) added to the core a feature named WH, which is a generalized version of a feature found in the Zhong

Linguistic hypothesis testing has been one of the main goals of the Grammar Matrix project since day one, but the range and the complexity of the hypotheses which can be tested depend directly on the syntactic and typological coverage of the system, which at first was modest. Bender *et al.* (2010) marked a significant milestone of the Grammar Matrix project, with support added for multiple phenomena and a wide range of typological profiles. Since then, another decade of contributions to the system have taken place (Section 3.1). After a brief overview of how the system can be used to generate a grammar (Section 3.2), we discuss the formal (Section 3.3) and methodological (Section 3.4) innovations that have expanded the capabilities of the system since 2010.

### *Grammar Matrix libraries added since 2010*

3.1

Table 1 lists all the Grammar Matrix libraries that are currently available via the web questionnaire. Twelve new libraries have been added since 2010, increasing the system's scope and the complexity of interactions which can be studied. In particular, the libraries for complex clauses (Howell and Zamaraeva 2018; Zamaraeva *et al.* 2019b) enable the Grammar Matrix-derived grammars to parse recursive sentences, meaning much larger test suites can be used for development and evaluation (see Section 3.4.1). The library for information structure (Song 2014) brought in the important potential to associate information structural meanings with a range of syntactic phenomena used to mark information structure in the world's languages. This, in turn, opened up the possibility of modelling aspects of interrogatives in terms of information structure (Zamaraeva 2021a). The revamped morphotactics library (Goodman 2013) and lexicon and morphology extensions for adjectives and copulas (Trimble 2014) in combination with the new libraries for adnominal possession (Nielsen 2018; Nielsen and Bender 2018), evidentials (Haeger 2017), valence change (Curtis 2018a,b), and nominalization (Howell *et al.* 2018) allow us to model

---

grammar of Chinese (Fan 2018), to accommodate cross-linguistic patterns of question word fronting. For the discussion, see Zamaraeva 2021a, page 188, footnote 61.

Table 1: The Grammar Matrix libraries with selected typological sources

Library	Citation(s)	Selected typological sources
Coordination	Drellishak and Bender 2005	Payne 1985; Stassen 2000; Drellishak 2004
Polar Questions	Bender and Flickinger 2005	–
Person, Number, Gender	Drellishak 2009	Cysouw 2003; Siewierska 2004; Corbett 2000
Agreement	Drellishak 2009	Corbett 2006
Case; Direct-Inverse	Drellishak 2008, 2009	Givón 1994
Argument Optionality	Saleem and Bender 2010; Saleem 2010	Ackema et al. 2006; Dryer 2013a
Tense	Poulson 2011	Comrie 1985; Dahl 1985
Aspect	Poulson 2011	Comrie 1976; Bybee et al. 1994
Lexicon	Drellishak and Bender 2005; Trimble 2014	Dixon 2004
Morphotactics	O'Hara 2008; Goodman 2013	–
Sentential Negation	Crowgey 2012, 2013	Dahl 1979; Dryer 2013b
Information Structure	Song 2014	Féry and Krifka 2008; Büring 2009
Adjectives; Copulas	Trimble 2014	Dixon 2004; Stassen 1997, 2013
Evidentials	Haeger 2017	Aikhenvald 2004; Murray 2017
Nominalized Clauses	Howell et al. 2018	Noonan 2007
Clausal Modifiers	Howell and Zamaraeva 2018	Thompson et al. 1985
Valence Change	Curtis 2018b,a	Haspelmath and Müller-Bardley 2001
Adnominal Possession	Nielsen and Bender 2018; Nielsen 2018	Payne and Barshi 1999; Heine 1997
Clausal Complements	Zamaraeva et al. 2019b	Noonan 2007
Constituent Questions	Zamaraeva 2021a	Haspelmath et al. 2013; Hagège 2008



grammars which account for a fairly wide range of data from descriptive sources on languages with very different typological profiles, as discussed in Section 3.4.2 and illustrated in Figure 10.

*How to use the libraries: an example*

3.2

To illustrate how a user would use the Grammar Matrix customization system to model a particular phenomenon, we show an example of how one could fill out the questionnaire for constituent questions (Zamaraeva 2021a) in Paresi-Haliti [pab] (Arawakan). As we will describe in Section 3.4.1, the process of library development and evaluation involves using the customization system to generate grammars and then using those grammars to parse sentences from test suites. In this case, we describe how Zamaraeva (2021a) created a customized grammar for Paresi-Haliti based on the examples and description in Brandão 2014. Later in Section 4.1, we present a case study related to the evaluation of the constituent questions library on this language.

Based on the description in Brandão 2014, the subpage for constituent questions may look as in Figure 3. For example, Figure 3 reflects the hypotheses that Paresi-Haliti fronts one question phrase obligatorily and that the question words may be overtly marked with focus. The reader can see in Figure 3 that the Constituent Questions subpage of the Grammar Matrix web questionnaire references two other subpages, namely Information Structure and Lexicon. Given the specifications shown in Figure 3, at least one question word must be specified on the Lexicon subpage, as shown in Figure 4. Likewise, on the Information Structure subpage (Song 2014), an affix or a clitic which can attach to question words (as well as other words) must be added. In this case, a contrastive focus marker is specified as in Figure 5. In combination with other grammar specifications made through these and other subpages of the Grammar Matrix web questionnaire, it is possible to obtain an implemented grammar of Paresi-Haliti. We can then test its behaviour with respect to a test suite of grammatical and ungrammatical examples, as discussed further in Section 4.1.

The web questionnaire is capable of producing human-readable grammar specifications that can be saved and re-uploaded later or

## Constituent (wh-) Questions [\[documentation\]](#)

Please indicate which strategy your language uses to form constituent (aka wh-) questions. You may leave this section blank, in which case your grammar will not include a wh-question-forming strategy.

If you fill out this page, you must also fill out the Question Words section on the Lexicon page.

### Choices regarding the position of question phrases

Question phrases can appear at the left edge of the sentence regardless of the position the questioned constituent would appear in (*Who did you see? I know who you saw etc.*):

- Only one question phrase can be fronted
- All question phrases can be fronted
- Question phrases cannot be fronted (*stay in situ*)

There is **obligatory** fronting:

- of at least one question phrase
- of all question phrases
- fronting is optional

There is pied piping of:

- noun phrases (*Which book did you read?* is possible),  and it is obligatory (*\*Which did you read book?* is impossible in your language);
- adpositional phrases (*To whom did you speak?* is possible),  and it is obligatory (*\*Who did you speak to?* is impossible in your language).

### Other choices

- Only one question phrase is allowed per sentence.
- Constituent questions are marked morphologically (specify lexical rules on the Morphology page).
- Question words may bear overt focus marking (specify on Information Structure page).
- Interrogative verbs (add appropriate entries on the Lexicon page).

If you specified Auxiliary-Subject inversion for polar questions, does it also happen in constituent questions?

- yes, in matrix clauses
- also in embedded clauses with constituent questions
- but not in questions about subjects.

<a href="#">Main page</a>
<a href="#">Gen Info</a> <a href="#">Word Order</a> <a href="#">Number</a> <a href="#">Person</a> <a href="#">Gender</a> <a href="#">Case</a> <a href="#">Poss</a> <a href="#">Dir-inv</a> <a href="#">TAM</a> <a href="#">Evidentials</a> <a href="#">Features</a> <a href="#">Neg</a> <a href="#">Coord</a> <a href="#">Y/N Qs</a> <a href="#">Wh-Qs</a> <a href="#">Info Str</a> <a href="#">Arg Opt</a> <a href="#">Nmz</a> <a href="#">Embed Claus</a> <a href="#">?Clausal Mod</a> <a href="#">Lexicon</a> <a href="#">?Morph</a> <a href="#">Toolbox Import</a> <a href="#">Test S</a> <a href="#">TbG Options</a>
<a href="#">Choices file</a> <small>(right-click to download)</small> <a href="#">Save &amp; stay</a> <a href="#">Clear current subpage</a> Create grammar: <a href="#">tgz</a> , <a href="#">zip</a>

Figure 3: The Grammar Matrix web questionnaire, Constituent Questions subpage, filled out for Paresi-Haliti [pab]

hand edited, and also acts as the intermediary to the customization system. As an example, the portion of the text specification corresponding to Figures 3 and 5 can be seen in Figure 6.

Based on specifications such as those shown in Figure 6, the customization system applies logic that outputs a customized grammar including the core types as well as language-specific types, rules and

**Noun Types** | [visualize noun hierarchy](#) (experimental)

Some nouns in this language take adjectives as incorporated affixes:

▶ common (noun1)  
▼ wh (noun2)

**Noun type 2:**

Type name:

Supertypes:  ▼

This is a personal pronoun type

This is a question pronoun (like *who/what*)

Features:

Name:  Value:  ▼

For nouns of this type, a determiner is:  obligatory  optional  impossible

Stems:

Spelling:  Predicate:

Figure 4:  
A portion of the Lexicon subpage of the Grammar Matrix web questionnaire, filled out for Paresi-Haliti [pab]

**Contrastive Focus**

My language uses the same position to express contrastive focus as non-contrastive focus.

My language places contrastively focused constituents in a specific position. The position is

clause-initial.

clause-final.

preverbal.

postverbal.

▼ c-focus-marker2

This marker is

an affix (You should create this affix on Morphology.)

an adposition (You should create this adposition on Lexicon.)

a modifier that appears  ▼  ▼ Spelling:

Figure 5:  
A portion of the Information Structure subpage of the Grammar Matrix web questionnaire, filled out for Paresi-Haliti [pab]

lexical entries. For example, specifying an information structure clitic as in Figure 5 will result in the types shown in Figures 7–8 being added to the grammar. These types, in turn, rely on supertypes such as *no-rels-hcons-lex-item* and *one-icons-lex-item* in Figure 7 which are defined in the Grammar Matrix’s core grammar.

The grammar code in Figures 7–8 represents HPSG feature structures in a machine readable form, specifically in TDL, which is compatible with the DELPH-IN JRF. Assuming the grammar files contain

Figure 6:  
Text specification output  
by the Grammar Matrix questionnaire

```

section=wh-q
front-matrix=single
matrix-front-opt=single-oblig
pied-pip=on
oblig-pied-pip-noun=on
focus-marking=on
wh-q-inter-verbs=on

section=info-str
c-focus-pos=clause-initial
c-focus-marker2_type=modifier
c-focus-marker2_pos=after
c-focus-marker2_cat=nouns, verbs
c-focus-marker2_orth=ala

```

```

infostr-marking-mod-lex := no-rels-hcons-lex-item &
                        one-icons-lex-item &
[ SYNSEM [ NON-LOCAL non-local-none,
          LOCAL [ CONT.ICONS.LIST < #icons &
                  [ IARG2 #target ] >,
                  CAT [ VAL [ SUBJ < >,
                              COMPS < >,
                              SPR < >,
                              SPEC < > ],
                              HEAD adv &
                              [ MOD < [ LIGHT luk,
                                          LOCAL [ CONT.HOOK [ INDEX #target,
                                                                ICONS-KEY #icons ],
                                                                CAT [ MKG [ FC na-or--,
                                                                TP na-or-- ],
                                                                WH.BOOL bool ] ] ] > ] ] ] ].

```

Figure 7: Grammar code output by the customization system

```

contrast-focus-marking-mod-lex := infostr-marking-mod-lex &
[ SYNSEM.LOCAL.CAT [ MKG fc,
                    HEAD.MOD < [ L-PERIPH luk,
                                LOCAL [ CAT.HEAD +nv,
                                        CONT.HOOK.ICONS-KEY contrast-focus ] ] >,
                    POSTHEAD + ] ].

```

Figure 8: Grammar code output by the customization system

enough code to constitute a functional grammar, they can be directly used with DELPH-IN parsers/generators such as the LKB or ACE (see Section 2.3).<sup>16</sup> Thus, by adding a new Grammar Matrix library for a particular syntactic phenomenon, we enable the user to obtain a machine-readable HPSG grammar capable of parsing and generating sentences featuring this phenomenon by simply filling out a web questionnaire and without the need to write the grammar by hand (see Section 6.2–6.3).

### *Formal innovations*

3.3

Once a user has created a grammar with the Grammar Matrix, they can continue developing it by adding or revising analyses to cover more phenomena. At this point, they must engage directly with the DELPH-IN JRF. As mentioned in Section 2.3, this formalism is deliberately restricted (Copestake 2002a). This means some constraints that are used in theoretical HPSG cannot be directly expressed using the DELPH-IN JRF. In particular, the formalism does not support relational constraints, where an operation on a specific feature value influences the value of another feature. Examples of such relations are applying logical-OR to feature values, list append (used for semantic composition and the handling of non-local features, among other things), and the shuffle operator (used in some analyses of variable word order).<sup>17</sup>

Emerson (2017, 2019, 2021, and forthcoming) has shown that, without changing the formalism, relational constraints can be mimicked using “computation types” and “wrapper types”. These computation types can be used to trigger operations such as logical-OR, and recursive type constraints can result in several lists being appended.<sup>18</sup>

---

<sup>16</sup>The Grammar Matrix customization system includes a validation component tasked with ensuring that the grammar specification is both complete and consistent enough to produce a functioning grammar. When the validation component detects that this is not the case, it signals this information to the user through warnings and errors on the questionnaire web pages.

<sup>17</sup>For details on non-local features in HPSG see Pollard and Sag 1994 and Ginzburg and Sag 2000; for DELPH-IN list implementation of list-valued features, see Copestake 2002a.

<sup>18</sup>See also Aguila-Multner and Crysmann 2018 for the discussion of application of append-lists in the context of feature resolution in coordination.

A full explanation of the workings of computation types lies beyond the scope of this paper. The main point we wish to make here is that they can ease grammar development. We illustrate this by comparing the classic DELPH-IN implementation of append operations, through difference lists (for an exposition, see Copestake 2002b, Section 4.3), to the new *append-list* type.

Examples (5)–(7) illustrate how difference lists (*diff-lists*) may be used to represent appending operations.<sup>19</sup> Example (5) provides the basic type definition of a *diff-list* specifying that it consists of two lists. Example (6) shows the definition of a *diff-list*  $\langle !a,b! \rangle$ . Note that the value of LAST is identical to the REST of the list starting with *b*. As such, LAST corresponds to the end of the list.

$$(5) \begin{bmatrix} \textit{diff-list} \\ \text{LIST} \quad \textit{list} \\ \text{LAST} \quad \textit{list} \end{bmatrix} \qquad (6) \begin{bmatrix} \textit{diff-list} \\ \text{LIST} \begin{bmatrix} \textit{nonempty-list} \\ \text{FIRST} \quad a \\ \text{REST} \begin{bmatrix} \textit{nonempty-list} \\ \text{FIRST} \quad b \\ \text{REST} \quad \boxed{1} \textit{list} \end{bmatrix} \end{bmatrix} \\ \text{LAST} \quad \boxed{1} \end{bmatrix}$$

In difference lists, the end of one list can be identified with the beginning of another list. The example below illustrates how this can be used to create the *diff-list* on the left by appending the two lists following it.

$$(7) \begin{bmatrix} \textit{diff-list} \\ \text{LIST} \quad \boxed{1} \\ \text{LAST} \quad \boxed{3} \end{bmatrix} \begin{bmatrix} \textit{diff-list} \\ \text{LIST} \quad \boxed{1} \\ \text{LAST} \quad \boxed{2} \end{bmatrix} \begin{bmatrix} \textit{diff-list} \\ \text{LIST} \quad \boxed{2} \\ \text{LAST} \quad \boxed{3} \end{bmatrix}$$

Using *diff-lists* for such operations requires carefully keeping track of the components of the list. The “end” of a difference list is actually an underspecified list, and for that reason, difference list appends are

<sup>19</sup>These examples correspond to examples (14)–(16) in Zamaraeva and Emerson 2020, pages 162–163. More details and examples can also be found in Zamaraeva 2021a, pages 42–43.

notoriously easy to break when introducing new types to the grammar, leading to such problems as overgeneration, spurious ambiguity, and semantic representations with missing predications. Also, it is difficult to count elements on a difference list.<sup>20</sup> In practice, wrongly defined difference lists are a well-known source of errors in the grammar.<sup>21</sup>

The *append-list* type, illustrated in Example (8) has a feature APPEND, which allows for simple and elegant syntax,<sup>22</sup> thereby making grammars easier to develop and maintain. Example (9) illustrates what appending two lists looks like when using *append-list*. For a more detailed exposition of how the *append-list* type works, see Zamaraeva and Emerson 2020.<sup>23</sup>

$$(8) \left[ \begin{array}{l} \textit{append-list} \\ \text{LIST} \quad \boxed{0} \textit{list} \\ \text{APPEND} \quad \left[ \begin{array}{l} \textit{list-of-list-wrappers-with-append} \\ \text{RESULT} \quad \boxed{0} \end{array} \right] \end{array} \right]$$

$$(9) \left[ \begin{array}{l} \textit{append-list} \\ \text{APPEND} \quad \langle \boxed{1}, \boxed{2} \rangle \end{array} \right] \quad \boxed{1} \left[ \begin{array}{l} \textit{append-list} \\ \text{LIST} \quad \langle a, b \rangle \end{array} \right] \quad \boxed{2} \left[ \begin{array}{l} \textit{append-list} \\ \text{LIST} \quad \langle c \rangle \end{array} \right]$$

Implementing the *append-list* type in the Grammar Matrix allowed for faster development of analyses which relied heavily on manipulating non-local lists, such as the ones developed for the constituent questions library (Zamaraeva 2021a).

### Methodological innovations

3.4

This section describes several important methodological principles characteristic of the Grammar Matrix development (Section 3.4.1) and what innovations took place in the recent years with respect to

<sup>20</sup> See Zamaraeva and Emerson 2020 for details.

<sup>21</sup> We base this claim on our experience as grammar engineers and our experience of teaching grammar engineering to others.

<sup>22</sup> In the sense of programming language syntax, not a branch of linguistics.

<sup>23</sup> Examples (8) and (9) correspond to examples (17) and (18) in Zamaraeva and Emerson 2020, page 164.

those principles. The first innovation is that we extended the practice of testing analyses for generalizability against held-out languages to using held-out language families (Section 3.4.2). The second important development is the evolution of “regression testing” (Section 3.4.3), which ensures an explicit area of applicability for the large and complex system of grammatical hypotheses. The third is CLIMB (Section 3.4.4), which is a methodology that allows one to track, after a starter grammar was created (e.g. with the Grammar Matrix), how one analysis influences subsequent decisions, and what the alternatives could have been. Finally, the last innovation is the “Spring cleaning” algorithm (Section 3.4.5), which allows the identification of portions of the grammar that are in fact unused.<sup>24</sup>

### 3.4.1

#### Data-driven Grammar Matrix development

The overall methodology for Grammar Matrix development was first summarized in Bender *et al.* 2010. The development is driven by typology and data: it starts from aggregating typological descriptions and exemplar sentences for the phenomenon for which support is being added; data is used as a guide throughout the development of analyses; finally evaluation is performed using previously unseen (“held-out”) languages, usually from held-out language families.

Grammar Matrix libraries are developed in a data-driven manner, using a set of illustrative “development” languages and corresponding test sets. When adding support for a syntactic phenomenon, a Grammar Matrix developer typically first compiles test suites from several languages consisting of exemplar grammatical and ungrammatical sentences illustrating the syntactic phenomenon being modelled.<sup>25</sup>

---

<sup>24</sup> Both CLIMB and Spring cleaning methods can be applied to any DELPH-IN grammar but both were developed in the context of Grammar Matrix development (Fokkens 2014).

<sup>25</sup> In the context of Grammar Matrix library development and in cases when exemplar sentences from a descriptive grammar contain phenomena which are not being modelled and are not already present in the Grammar Matrix, the developer may have to simplify/modify the sentences, e.g. remove a greeting, discourse particle, or even a relative clause (as relative clauses are not currently supported by the Grammar Matrix customization system). In such cases, it is ideal to get judgments on the resulting modified sentences from an



Many of these test suites come from natural languages which the developer encountered in their typological literature review. Others are constructed using artificial languages, or “pseudo-languages”, to ensure the testing of typological combinations not illustrated with specific examples in the typological literature.

To illustrate the data-driven development process in more detail, we will use an example that includes both a test suite from an actual language and a test suite made of artificial data representing a hypothesized language type (pseudo-language). It helps to (i) connect the methodology to software engineering practices and (ii) give the reader additional background for the case study presented later in Section 4.3. The process described below is exactly the same for real language data.

The process of library development starts from reading a descriptive source and compiling a test suite of examples, grammatical and ungrammatical, illustrative of the phenomenon for which the library is being added. For example, if the descriptive grammar states that the language has separate morphological paradigms for verbs in declarative (10a) and interrogative sentences (10b–c), the test suite will include examples of each.

- (10) a. oža-va    iche-žee-v  
          track-ACC see-FUT-1SG  
          ‘I will see the tracks.’ [neg] (Khasanova and Pevnov 2002;  
          cited by Hölzl 2018, page 295)
- b. ii-ǰə-m = i?  
          enter-FUT.Q-1SG.Q = Q  
          ‘Shall I come in?’ [neg] (Khasanova and Pevnov 2002; cited  
          by Hölzl 2018, page 295)
- c. eeva iche-ža-m?  
          what see-FUT.Q-1SG.Q  
          ‘What will I see?’ [neg] (Khasanova and Pevnov 2002; cited  
          by Hölzl 2018, page 295)

---

L1 speaker; unfortunately that is not always possible to do. The methodology assumes that any modifications to the original sentences are carefully documented.

Examples (10a–c) come from Negidal [neg] (Tungusik). It uses the same paradigm for polar and constituent questions (e.g. *-m* in (10b–c)). This language represents one typological profile; another includes languages which use three distinct paradigms: one for declaratives, another for polar questions, and yet another for constituent questions. One such language is Makah [myh] (Wakashan). During the development of the constituent questions library, this typological profile was included as a formal experiment; Zamaraeva (2021a) was not aware that Makah has this feature but had hypothesized that such languages may exist and constructed illustrative artificial data (11a–i).

- (11) a. noun tverb-PQ noun?  
b. who iverb-WHQ?  
c. who tverb-WHQ what?  
d. who tverb-WHQ noun?  
e. noun tverb-WHQ what?  
f. \*noun tverb-WHQ noun?  
g. \*who tverb-PQ what?  
h. \*who tverb-PQ noun?  
i. \*noun tverb-PQ what?

At this point, the library developer has the test suite like the one above (11a–i), illustrating the fact that the language uses three separate morphological paradigms, and the appropriately filled out questionnaire which, assuming the questionnaire-customization system interface was already implemented, generates textual grammar specification like in Figure 9.<sup>26</sup>

Given a (complete) specification containing the sections relevant to the separate morphological marking in polar and constituent questions, the goal is for the customization system to output a grammar which behaves correctly with respect to the data in (11a–i), namely one that maps the grammatical strings (11a–e) to their correct linguistic representations and rejects the ungrammatical strings (11f–i).

---

<sup>26</sup>Only the morphology section of the specification is shown.

```
section=morphology
verb-pc1_name=mood
verb-pc1_obligatory=on
verb-pc1_order=suffix
verb-pc1_inputs=verb
verb-pc1_lrt1_name=polar
  verb-pc1_lrt1_feat1_name=question
  verb-pc1_lrt1_feat1_value=polar
  verb-pc1_lrt1_feat1_head=verb
  verb-pc1_lrt1_lri1_inflecting=yes
  verb-pc1_lrt1_lri1_orth=-PQ
  verb-pc1_lrt2_feat1_name=question
  verb-pc1_lrt2_feat1_value=wh
  verb-pc1_lrt2_feat1_head=verb
  verb-pc1_lrt2_lri1_inflecting=yes
  verb-pc1_lrt2_lri1_orth=-WHQ
verb-pc1_lrt3_name=ind
  verb-pc1_lrt3_feat1_name=question
  verb-pc1_lrt3_feat1_value=no
  verb-pc1_lrt3_feat1_head=verb
  verb-pc1_lrt3_lri1_inflecting=no
```

Figure 9:  
Lexical rules specification output  
by the Grammar Matrix questionnaire

At this point, the Grammar Matrix library developer has a clear map of what the finished library should cover, in terms of accepting and rejecting strings. This is the starting point of so-called test-driven development in software engineering (Beck 2003)<sup>27</sup> where first the tests are written and then the code is added to the program until all tests pass.

Building the library entails (i) deciding on target semantic representations, (ii) developing the implemented HPSG analyses that will produce those representations, (iii) developing the customization logic that will output the correct grammar components given a specification, and (iv) writing the questionnaire portions to elicit the specification. Grammar Matrix developers typically proceed by creating starter grammars through the customization system with enough other specifications to cover all other phenomena in the test suites and then using those grammars as test beds to work out analyses of specific variants of the phenomenon targeted by the library under development. Once those analyses are developed, and the semantic representations

---

<sup>27</sup> Beck (2003) is often credited for “rediscovering” the concept of test-driven development, as he popularized the term and the practice. The concept probably long predates his book, though we could find no other canonical citation.

they produce hand-verified, they can be generalized and added to the customization logic.

In the case of Grammar Matrix data-driven development, first we have the data (the test suite), and then we develop an analysis and add logic to the customization system until the system starts outputting a grammar which behaves correctly with respect to this data. It is an iterative process. For example, given only the specification in Figure 9 but no code designed to add something to the grammar based on this specification, the output grammar will not include the lexical rules for morphological marking and therefore will not cover sentences like (11a-i); its coverage over the test suite will be 0%.<sup>28</sup> The developer can then add the programmatic logic to the system such that, upon encountering a specification like the one in Figure 9, appropriate lexical rules, such as the one in (12), are added to the grammar. This is a type which is part of the analysis of distinct morphological marking in polar and constituent questions discussed in Section 4.3.

$$(12) \left[ \begin{array}{l} \textit{polar-lex-rule} \\ \text{SYNSEM|LOCAL|CAT|VAL} \left[ \begin{array}{l} \text{SUBJ} \quad \langle \langle \text{NON-LOCAL|QUE|LIST} \quad \langle \rangle \rangle \rangle \\ \text{COMPS} \quad \textit{non-wh-list} \end{array} \right] \end{array} \right]$$

Even in cases where there is already literature or other grammars containing similar types, developing analyses with the level of rigor and detail that the Grammar Matrix requires is often non-trivial. When building on the theoretical literature, the analyses may be sketched in too general a way, and may assume operations not available in the DELPH-IN JRF. When building on analyses already developed in other DELPH-IN grammars, the types are likely to be overly specific to another language. In many cases, especially characteristic of non-Indo-European typology, the analysis will simply not be found anywhere and will need to be developed from scratch. It is therefore expected that the process of developing a Grammar Matrix library involves many debugging cycles. The developer can then load the imperfect grammar into the software such as the LKB and perform interactive

---

<sup>28</sup>The coverage can be computed automatically using tools such as [incr tsdb()] (Oepen and Flickinger 1998).

debugging. The LKB will show any errors that occur when loading the grammar, unification failures that prevent a sentence from parsing, every possible parse tree for each sentence allowing the developer to identify the cause of any spurious parses, and the semantic representation for each parse tree allowing the developer to inspect the correctness of each parse. The developer can then refine the analysis to ensure that grammatical sentences in the test suite are parsed correctly and ungrammatical sentences are not parsed.

To summarize, data-driven development first lays out the grammatical territory to cover in a set of test suites, and then extends the grammar customization system to cover that territory. As the analysis is developed and refined, the grammar engineer moves back and forth between increasing the system's ability to handle grammatical sentences (coverage) and working to minimize both spurious ambiguity (extraneous extra parses of grammatical sentences) and overgeneration (parses of ungrammatical sentences).<sup>29</sup> After the development is finished, as far as can be tested with the initial test suite, the library is evaluated as described in Section 3.4.2.

#### Evaluation with languages from held-out language families

#### 3.4.2

After a library's development is concluded, the developer performs evaluation on held-out languages from held-out language families. This means each evaluation language must come from a different language family than other evaluation languages, and furthermore, from a family not represented in the languages used in the library's development (Zamaraeva 2021a, page 103).<sup>30</sup> The goal is to assess how well the Grammar Matrix analyses generalize with respect to a randomly selected set of languages, specifically languages which may have different properties compared to the languages that were driving the development. This is meant to approximate what will happen when users approach the Grammar Matrix to model additional languages.

---

<sup>29</sup>In practice, lack of coverage, some spurious ambiguity, and some overgeneration may be unavoidable due to development time constraints, in which case the specific cases are documented and left for future work.

<sup>30</sup>While at this stage we avoid including language families that we worked with directly in library development, we do not necessarily exclude families just because they were included in or informed the typological surveys we build on.

The process of evaluation is very similar to the one described above in Section 3.4.1, starting with descriptive sources and the compilation of a test suite, then filling out the questionnaire, obtaining a grammar from the customization system, and deploying it on the test suite. However, at the evaluation stage no modifications are made to the system and the coverage, the overgeneration, and the spurious ambiguity are expected to not be perfect and are reported as measures of the quality of the newly added library.<sup>31</sup>

The practice of using held-out languages (but not necessarily coming from unseen language families) goes back to at least Saleem 2010. Starting with Haeger 2017, only unseen language families are used as part of the methodology. Ultimately, all the test suites (both development and evaluation) are preserved along with the corresponding language specifications and expected “gold” semantic representations for each sentence in the test suite. These sets of files constitute the content of the regression testing system to ensure continued functioning of existing analyses (Section 3.4.3).

### 3.4.3

#### Automatic testing of all existing analyses

A crucial part of Grammar Matrix development methodology is the automatic testing of all analyses currently implemented in the system with respect to stored test suites containing data from different languages, both actual languages and abstract pseudo-languages (regression testing; Bender *et al.* 2007). Regression testing is what ensures that the combination of analyses that constitute the Grammar Matrix have at least a known, explicit area of applicability. Most importantly, regression testing makes it possible to see precisely whether and how any new analysis affects the previous system of analyses with respect to the previously established area of applicability. In other words, regression testing of the Grammar Matrix allows us to extend the computationally assisted method of fragments (Montague 1974; Partee 1979; Gazdar *et al.* 1985) to a cross-linguistic arena.

The term regression testing comes from software engineering where it describes tests that check functional behaviour of a system over time; i.e. each modification to the system can be tested on

---

<sup>31</sup> Any issues discovered during the evaluation stage are documented such that they can be addressed later.

previously used inputs for which the expected (“gold”) output was recorded. If the system behaves differently after a modification, this is a “regression”, assuming there is no mistake in the stored gold outputs. Regressions need to be addressed, either by fixing the system such that the behaviour is back to what was expected or by updating the expected “gold”, if the new output is in fact correct or closer to correct. The practice of regression testing for monolingual grammar development is elaborated in Oepen and Flickinger 1998, Oepen 2002, and Oepen *et al.* 2002, among others.

In the context of the Grammar Matrix, regression tests are sets of grammar specifications (valid inputs to the customization system), input language strings from languages corresponding to the descriptions (with grammatical and ungrammatical items), and finally the corresponding “gold” semantic structures (one or more correct structures for each grammatical sentence and no structures for ungrammatical ones). For example, the test suite discussed above in (11a–i) along with the corresponding grammar specification and the set of correct MRS representations for all the grammatical sentences in the test suite will constitute a regression test after the development of this portion of the system is completed. Running a Grammar Matrix regression test involves invoking a system which automatically feeds a grammar specification to the customization system to obtain a grammar fragment, uses the grammar to process the input language strings, and compares the output to the stored “gold” results. If there is any difference between the expected output and the actual output, the test is flagged and the developer can investigate what causes the difference.

The regression testing system was in place by 2010 with 130 test suites; in the past decade it has had significant extensions, comprising 527 test suites at time of publication. Re-engineering of the system using modern software engineering methods<sup>32</sup> resulted in a much faster, more robust system. This is crucial in the context of the greatly increased number of tests, as otherwise the time to run all the tests

---

<sup>32</sup>The regression testing system was re-engineered twice in the decade, once by Sanghoun Song to use the faster parser ACE (Crysmann and Packard 2012) instead of the LKB parser (Copestake 2002b), and once by Michael W. Goodman to use the PyDelphin libraries and multiprocessing, both times with input from other DELPH-IN contributors.

often (such as for each small modification) would have become prohibitive. Another key change in Grammar Matrix regression testing practice, supported by the more robust regression testing facilities, is the move to include not only artificial (“pseudo-language”) tests in the regression testing suite, but also tests depicting natural languages used in library development (“illustrative languages”) and testing (“held-out languages”).<sup>33</sup>

The current regression testing system counts 527 test suites, of which 75 are from 60 unique natural languages representing 40 language families. Figure 10 shows these 60 languages on the world map. Table 2 summarizes how many languages represent each family.<sup>34</sup>

The Grammar Matrix regression testing system represents a crucial methodological principle of the project, namely that the analyses can be rigorously tested together, thus allowing Grammar Matrix developers to state confidently that the set of HPSG analyses implemented in the customization system definitely accounts at least for the data stored in the system. Over the years, the system has grown to cover 40 language families from all over the globe. The data compiled from descriptive resources for 60 languages that the Matrix regression testing system currently contains can be reused for research purposes by anyone who is interested in typologically diverse data on any of the syntactic phenomena represented in the Grammar Matrix.<sup>35</sup>

---

<sup>33</sup>Thus, while all of the libraries were tested on natural languages as well as pseudo-languages, there are some natural languages described in the Grammar Matrix literature which never made their way into the regression test suite.

<sup>34</sup>Language family and ISO 639-3 codes are given as in WALS (Dryer and Haspelmath 2013) or, if not found there, as in Glottolog (Hammarström *et al.* 2021), except in cases where we learned that the name listed in those resources contained a slur. The second column in the table is the number of unique languages represented per family, e.g. there are two unique Afro-Asiatic, Niger-Congo, etc., languages represented in the system. The third column shows the total number of test profiles. This last number includes any repetitions, e.g. Japanese is represented in the system by 3 test profiles which may contain different sentences and target different syntactic phenomena. That is why the number in column 3 is not necessarily obtained by multiplying the number in column 2 by the number of items in the corresponding cell in column 1.

<sup>35</sup>The median size of the test suites is 17 sentences. The largest test suite is for Umatilla Sahaptin [uma] (Penutian) (Drellishak 2009) and it contains over 6000



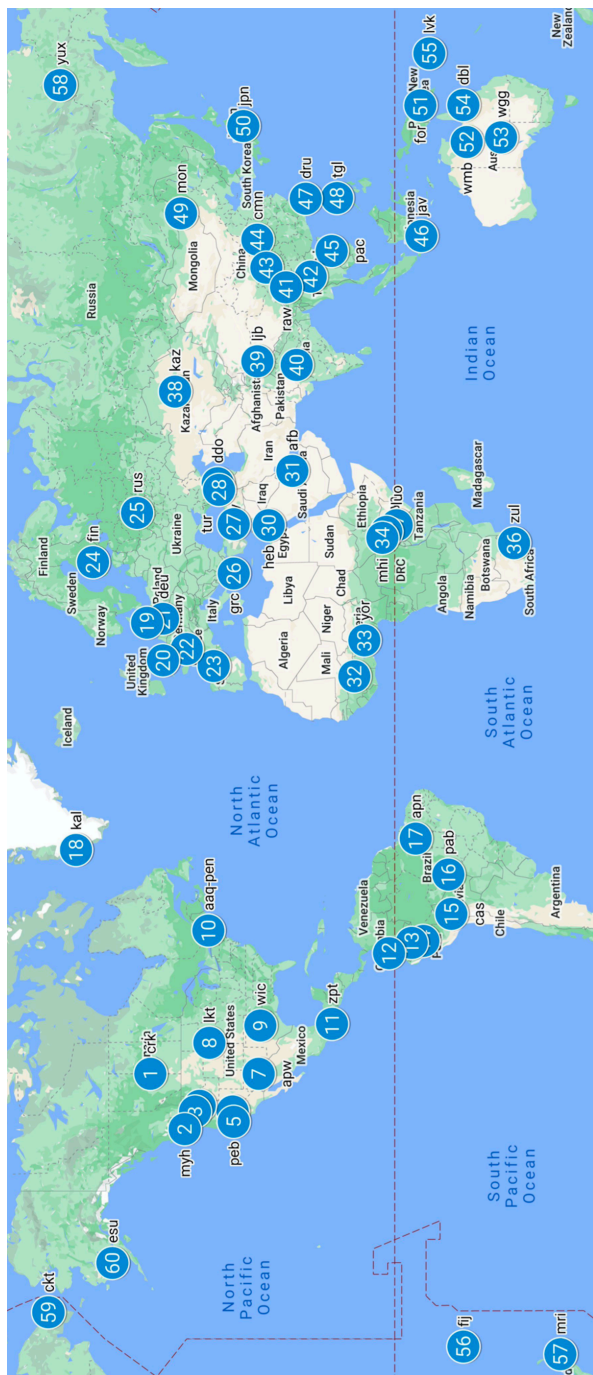


Figure 10: Languages in the Grammar Matrix regression testing system. Map: Google Maps; map data: SIO, NOAA, U.S. Navy, NGA, GEBCO Image Landsat/Copernicus. Locations/spellings as in WALS (Dryer and Haspelmath 2013) or Glottolog (Hammarström *et al.* 2021). Starred\* numbers are not fully visible. 1. Cree [crk]; 2. Makah [myh]; 3. Yakima Sahaptin [yak]; 4\*. Umatilla Sahaptin [umal]; 5. Eastern Pomo [peb]; 6\*. Washo [was]; 7. Western Apache [apw]; 8. Lakota [lkt]; 9. Wichita [wic]; 10. Penobscot [pen]; 11. Zapotec [zpt]; 12. Awa Pit [kwl]; 13. Uranina [lra]; 14\*. Shipibo-Konibo [shp]; 15. Moseten [cas]; 16. Paresi [pab]; 17. Apinaje [apn]; 18. West Greenlandic [kal]; 19. North Frisian [frr]; 20. English [eng]; 21. German [deu]; 22. French [fra]; 23. Basque [eus]; 24. Finnish [fin]; 25. Russian [rus]; 26. Ancient Greek [grc]; 27. Turkish [tur]; 28. Georgian [kat]; 29\*. Tsez [ztl]; 30. Hebrew [heb]; 31. Gulf Arabic [afb]; 32. Jalkunan [bxl]; 33. Yoruba [yor]; 34. Madi [mhi]; 35\*. Lango [laj]; 36. Zulu [zul]; 37\*. Luo [luo]; 38. Kazakh [kaz]; 39. Ladakhi [ljb]; 40. Hindi [hin]; 41. Rawang [raw]; 42. Blang [blr]; 43. Quiang [cng]; 44. Mandarin [cmn]; 45. Pacoh [pac]; 46. Javanese [jav]; 47. Rukai [ruk]; 48. Tagalog [tgl]; 49. Mongolian [mon]; 50. Japanese [jpn]; 51. Fore [for]; 52. Wambaya [wmb]; 53. Wangkangurru [wgg]; 54. Dyrribal [dbl]; 55. Lavukaleve [lvk]; 56. Fijian [fij]; 57. Maori [mri]; 58. Yukaghir [yux]; 59. Chukchi [ckt]; 60. Yup'ik [esu]

Table 2:  
Language families  
represented  
in the regression  
testing system

Language family (ISO 639-3 codes)	N languages	N test suites
Indo-European (deu, eng, fra, frr, grc, hin, rus)	7	13
Austronesian (dru, fij, jav, mri, tgl)	5	6
Sino-Tibetan (cmn, cng, lbj, raw)	4	4
Afro-Asiatic (afb, heb), Algie (aaq-pen, crk), Altaic (kaz, tur), Austro-Asiatic (pac, blr), Eastern Sudanic (luo, laj), Inuit-Yupik-Unangan (esu, kal), Niger-Congo (yor, zul), Pama-Nyungan (dbl, wgg), Penutian (uma, yak)	2	19
Arawakan (pab), Barbacoan (kwi), Basque (eus), Caddoan (wic), Central Sudanic (mhi), Hokan (peb), Japanese (jpn), Kartvelian (kat), Macro-Ge (apn), Mande (bxl), Mirndi (amb), Mongolic-Khitani (mon), Mosestenean (cas), Na-Dene (apw), Nakh-Daghestanian (ddo), Otomanguean (zpt), Panoan (shp), Siouan (lkt), Solomons East Papuan (lvk), Trans-New Guinea (for), Uralic (fin), Uralic (ura), Wakashan (myh), Washo (was), Yukaghir (yux)	1	33

3.4.4

CLIMB

One motivation for implementing precision grammars is that natural languages are complex, consisting of many phenomena that interact. The analyses for these phenomena also interact, which makes it practically impossible to verify whether a newly proposed analysis interacts correctly with existing analyses without the aid of a computer (see also Section 4). Implementing grammars provides the means to test this through systematically adding test sets that represent covered

---

sentences; it was partially computer generated based on the examples found in a descriptive grammar. The largest test suite fully vetted by an L1 speaker is for Russian [rus] (Indo-European) and it contains 273 sentences (Zamaraeva 2021a).

phenomena and applying regression tests as described in Section 3.4.3. Regression testing is also used in the development of individual grammars, as grammar engineers create and continuously update test data, and test the grammar on the full set of test data after each change.

In this way, grammar engineering can contribute to more systematic syntactic research. A challenge that remains, however, is that regression tests only allow grammar engineers to look back. There are often multiple ways in which a phenomenon can be analyzed and the decision for a specific analysis can only be tested on those phenomena that have already been analyzed and not on those that are not covered yet.<sup>36</sup> It is inevitable that decisions are sometimes made based on inconclusive evidence. The order in which phenomena are considered can thus have a major impact on the resulting grammar (Fokkens 2011; Fokkens 2014, page 69).

The CLIMB<sup>37</sup> method (Fokkens *et al.* 2012) aims to address this challenge by extending the idea of grammar customization from providing a mere kick-start to a general methodology of grammar development. The basic idea behind CLIMB is that, in case of inconclusive evidence, alternative analyses are implemented in a *metagrammar* which can generate grammars with either of the solutions. The grammar developer can maintain the alternative analyses and keep on testing their interactions with analyses for other phenomena until sufficient evidence is found. CLIMB uses the Grammar Matrix customization software to continue the development of individual grammars *after* the kick-start from the general customization system has taken place. It is thus *per se* not a method for developing new libraries for the Grammar Matrix customization system itself. It can however also be used when developing new customization libraries. In fact, the method was first developed to compare alternative analyses for V2-word order across languages (Fokkens 2011). In practical terms, CLIMB consists of programmatic scripts which work with the Grammar Matrix files.

There are currently three versions of CLIMB for DELPH-IN grammars described in detail in Fokkens 2014, Chapters 3–4. In its origi-

---

<sup>36</sup> Naturally, linguists are not fully unaware of phenomena that are not covered yet and can take them into account to some extent.

<sup>37</sup> Comparative Libraries of Implementations with Matrix Basis (CLIMB).

nal form, CLIMB continues to use the Grammar Matrix customization system for grammar engineering. The grammar developer includes all analyses in the original customization system (possibly cleared of components that are not relevant for the language in question). The development cycle consists of (i) generating one or multiple versions of the grammar using the customization system; (ii) adding and testing a new analysis in one or more grammars; (iii) adding these analyses to the customization system; (iv) generating and testing grammars with all alternative analyses for previously covered phenomena; and (v) adapting the analysis for proper interaction with alternative analyses if applicable and testing again. The advantage of using this version of CLIMB is that it offers the full flexibility of the customization system.

A disadvantage is that it involves moving back and forth between declarative coding for grammar engineering and procedural coding in the customization system. Moreover, in practice, the full flexibility of the customization system is not likely to be exploited. Notably the morphotactics library offers countless options which are not likely to be considered as alternative options for which more evidence is needed. A second version, called declarative CLIMB, offers an alternative way of using CLIMB without writing procedural code. The grammar engineer can define alternative analyses and the accompanying modifications that are needed to make them work with the rest of the grammar with an indication of the analysis they belong to. The Grammar Matrix customization code is used behind the scenes to create well-formed grammars from a set of selected analyses.

The third version of CLIMB is an adaptation of declarative CLIMB meant to support research on large-scale grammars developed in the traditional way. Declarative CLIMB consists of a shared core and collection of (alternative) analyses from which grammars can be generated. In this third version, CLIMB provides a set of changes that can be applied to a working grammar. The grammar developer can define additions, replacements and components to be removed to adjust the grammar. The customization code is used to generate an adapted grammar based on the original grammar and the changes. The code can also generate the set of changes needed to replace the new alternative analysis with the original analysis. The developer can thus propose an alternative analysis, add new analyses to the grammar with

this alternative and generate a version of the grammar with these new analyses and the original analysis.

In the scenario above, CLIMB is used to test alternative hypotheses during (part of) the trajectory of grammar development for a specific language. The method can also be used for flexible parallel grammar development. For a specific language, CLIMB could support versions of the grammar that exhibit (slightly) different behaviour. This could be versions that are adapted for a specific domain (e.g. one that captures structures typically used on social media, one more aimed at newspaper text), that aim to be more robust rather than precise, or that are designed for a specific application. Fokkens (2014, Section 6.4) illustrates for instance how CLIMB can be used to include alternative rules to spot specific errors of second language learners, as also seen in e.g. Flickinger and Yu 2013, Morgado da Costa *et al.* 2016, and Morgado da Costa *et al.* 2020.

Another natural way of using CLIMB is for multilingual grammar development. In this case, grammar developers truly continue in the spirit of the Grammar Matrix customization system. When a new analysis is developed for one language, code generation can be integrated in grammars of other typologically related languages. The idea of aiming for full typological coverage is abandoned, which allows for more depth. In addition to parallel grammar development, this can result in a significantly larger jump start for a new grammar of an additional related language. For instance, Fokkens (2014, Section 6.2.5) illustrates the increase in coverage of phenomena captured when developing a grammar for Northern Frisian from gCLIMB (a metagrammar for German that also contains variations for Dutch) compared to developing it from the Grammar Matrix customization system alone. In addition to gCLIMB, CLIMB has been used to create a prototype for a Slavic metagrammar that can generate a basic grammar for Russian (Fokkens and Avgustinova 2013).

#### Spring cleaning

#### 3.4.5

One of the questions that arises when using a resource that supports grammar development for typologically diverse languages, such as the Grammar Matrix, is how much of the generic core and provided jump-start implementations end up being used. Though it is straightforward

to automatically check which type definitions have been modified, it is less trivial to find out which type definitions are actually active in a grammar and which are never invoked. When grammar developers implement analyses that take a different approach than the grammar core, for instance, they do not necessarily remove the corresponding components from the core. Likewise, obsolete type definitions are not necessarily removed when analyses are adapted or replaced. The spring cleaning algorithm (Fokkens *et al.* 2011) can be used to identify which components of the grammar actually influence the grammar's behaviour. The algorithm was developed with the specific purpose of identifying components of the Grammar Matrix that are an active part of (larger) grammars and relies on the code from the customization system to process DELPH-IN JRF.

A DELPH-IN grammar defines a hierarchy of typed feature structures. Each type inherits all constraints from its supertypes. A grammar furthermore defines instances: lexical items or rules defined through the type hierarchy. The parser and the generator start with instances: the parser forms syntactico-semantic representations of words based on its lexicon and lexical rules and then combines them using grammar rules. Conversely, the generator generates surface strings using the lexicon and lexical rules and combining them using grammar rules. This means any type that defines (part of) an instance impacts the grammar. These types are referred to as *instantiated types*. Combining components is done through unification. Types that influence whether instantiated types can unify therefore also impact the grammar. Conversely, types which are neither instantiated nor influence unification of instantiated types have no effect on the grammar.

The spring cleaning algorithm starts from the instance definitions. It then goes through the grammar and tags all types that are a supertype of an instance and marks them as instantiated types. It then extracts the feature values from all instantiated type definitions and marks the type definitions of these values and their supertypes as relevant types as well. In the next step, the algorithm checks whether the remaining types (that are not instantiated types nor types that are part of the definition of an instantiated type) enable unification of any relevant type. Any type that influences unification of relevant types is also marked as relevant. Remaining types are flagged as redundant. The algorithm was applied to four Grammar Matrix grammars repre-

senting three languages (two grammars with alternative word order analyses for Wambaya, one for Mandarin Chinese, and one for Bulgarian). The outcome showed that even relatively small grammars contained noise and that identifying superfluous types can help identify errors in the grammar (Fokkens *et al.* 2011). Occasional spring cleanings of grammars are therefore recommended.

### *Summary*

3.5

This section summarized how the support for syntactic phenomena in the Grammar Matrix has grown since 2010 and covered the most important formal and methodological innovations adopted in this context. We listed all current Grammar Matrix libraries (and will later illustrate interactions among some of these). We presented in detail the testing system that currently covers 60 languages from 40 language families and allows for automatic testing of any modification in any of the analyses with respect to data from this wide typological range. In addition, this section summarized some formal innovations particularly relevant to how the system implements non-local dependencies (further discussed in Section 4.3) and described algorithms which allow the grammar engineer to track how analyses in a DELPH-IN grammar influence each other and how to determine whether some parts of the grammar remain unused – which are important for future improvements of the Grammar Matrix project and the grammars it gives rise to.

## MAKING TENSIONS BETWEEN ANALYSES EXPLICIT

4

Grammar engineering allows a grammarian to identify unexpected interactions between analyses which might otherwise be overlooked due to the overall complexity of the grammar. Furthermore, in the case of the Grammar Matrix, this is done with respect to the entire cross-linguistic system that the framework provides. In this section, we present several examples of tensions between syntactic analyses which

were made explicit in the context of the Grammar Matrix's development and testing. We start in Section 4.1 with a case study illustrating the range of problems in the analyses that we are now able to discover and document. The important thing to note here is that documenting such a range of issues only became possible with the recent additions to the Grammar Matrix libraries (Section 3.1), because these issues all have to do with *interactions* among analyses of phenomena such as clausal complementation and modification, nominalization, adnominal possession, information structure, constituent questions, and long-distance dependencies. We then present a particularly intricate formal issue having to do with non-local dependencies and coordination which would probably be impossible to detect without a computational framework but has bearing on very common, seemingly simple sentences (Section 4.2). Finally, in Section 4.3, we discuss tensions that inform decisions of what belongs in the core grammar vs. the libraries, again revealed in the process of evaluating a new library on a held-out language.

#### 4.1 *Two word order hypotheses in Paresi-Haliti: A case study*

In the context of her work on the constituent questions library for the Grammar Matrix, Zamaraeva (2021a, Section 8.5.9) considers two alternative analyses for basic word order in Paresi-Haliti [pab] (Arawakan) based on a descriptive grammar of the language (Brandão 2014). Brandão 2014 features a number of long, complex examples, which is good material for testing the interaction between Grammar Matrix libraries. However, according to Zamaraeva (2021a, page 342), many examples are not yet fully glossed and some phenomena are not yet fully described or understood (as is normal for an underdocumented language). In particular, the word order is said to be mostly V-final (the most common order being SOV), yet personal pronouns can occur after the verb, and indeed if they are taken into account, then, according to another source, da Silva 2013 (cited by Brandão (2014, page 318)), the most frequent word orders in the language are SVO and OSV. All orders in fact occur with some frequency, according to Brandão (2014, page 319). The exact nature of the interaction of information structure with word order is not fully worked out, though



there is a section on focus and topic and many examples are glossed for information structure marking (13).

- (13) aliyakere = ta = la hatyohare  
how = EMPH = FOC this  
'How is this?' (Brandão 2014, page 335)

Due to multiple possible hypotheses for what the basic word order is, this part of the grammatical description of Paresi-Haliti is thus an excellent candidate for computationally assisted hypothesis testing, for example using the Grammar Matrix.

To that end, Zamaraeva (2021a, Section 8.5.9) presents two grammars of Paresi-Haliti, both produced automatically by the Grammar Matrix. The grammars represent two sets of hypotheses, one associated with SOV word order (with some of the other orders accounted for by information structure) and another with free word order (reflecting the fact that all orders are possible). The analyses are evaluated with respect to a Paresi-Haliti test suite containing grammatical and ungrammatical sentences. The test suite consists of 67 items, 64 of them grammatical. Out of those 64, 45 are directly from Brandão 2014 while 19 have modifications or were constructed by Zamaraeva (2021a) based on the information from Brandão 2014. Of the 3 ungrammatical examples, 2 are constructed by Zamaraeva (2021a) and 1 is from Brandão 2014. Table 3 presents the evaluation numbers obtained by running the LKB parser (Copestake 2002b) over the test suite.<sup>38</sup>

While it can be relatively easy to achieve close-to-perfect numbers on a (small) test suite that is driving grammar development, the Paresi-Haliti results from Zamaraeva 2021a, Section 8.5.9 (Table 3) represent the evaluation of the Grammar Matrix system on a held-out language family after the development process was frozen, and so relatively low

---

<sup>38</sup>Raw coverage refers to grammatical sentences for which a grammar can produce any reading at all; validated coverage is for sentences which get a parse with correct semantics; overgeneration is for ungrammatical sentences which nonetheless are accepted by the grammar; and finally ambiguity is the average number of readings per sentence. High overgeneration and ambiguity indicate problems with the grammar, as does low coverage; high raw coverage is not necessarily good unless validated coverage is also high.

Table 3:  
Two word order  
hypotheses  
for Paresi-Haliti

Hypothesis	Raw coverage (%)	Validated coverage (%)	Overgeneration (%)	Ambiguity
SOV	40/64 (62.5)	25/64 (39.0)	2/3 (66.7)	43.95
free	53/64 (82.8)	36/64 (56.0)	2/3 (66.7)	36.17

validated coverage, high overgeneration, and high ambiguity can all be expected. However, after evaluation results are reported, we can still have a good look into which exact problems led to the missing coverage as well as to overgeneration and any spurious ambiguity. For this we use automated DELPH-IN tools, particularly [incr tsdb()] (Oepen and Flickinger 1998) and treebanking tools.

A close examination of the two grammars with respect to the Paresi-Haliti test suite revealed the following issues in the Grammar Matrix system. First of all, we found out that the information structure library was overconstraining SOV grammars such that complements could never be extracted out of VP.<sup>39</sup> Removing that constraint did not lead to any regressions in any of the 527 regression tests, so that problem can easily be fixed at the level of the entire Grammar Matrix system. Note that it took a complex test suite featuring both information structure marking and constituent questions in combination with the SOV word order to discover this problem; without testing the interaction, the problem went unnoticed for years.

Second, the large ambiguity in both grammars was caused in particular by an interaction between the adnominal possession and the constituent questions libraries in which unwanted underspecification led to spurious phrase structure rule application. The adnominal possession library (Nielsen 2018) provides lexical rules for constructions in which possession is marked morphologically (on the possessor, the possessum, or both). At the time this library was developed, only a few analyses within the information structure library exercised non-local features, and (without specific tests for this interaction) it was not apparent that the lexical rules were leaving the non-local features underspecified. A grammar with both adnominal possession lexical rules and an analysis involving, say, head-filler rules for constituent

<sup>39</sup>More specifically, subject-head phrase was constrained to have an empty SLASH list.

questions will produce (nonsensical) “readings” where a filler-gap rule applies to a structure in which there is no gap (or strictly speaking, where the information about whether or not there is a gap was lost).

More sources of spurious ambiguity were discovered thanks to these two grammars. They include interactions between the analyses of constituent questions, information structure, and clausal modifiers and nominalization libraries. For example, we discovered that the customization system was not properly customizing our base analysis of the question pronoun for grammars that also had nominalizers. This led to nominalization lexical rules applying to question pronouns (while they should only apply to verbs) and ultimately to spurious parses. The same need for a nominalization-blocking constraint was found in the filler-gap rule added by the information structure library to grammars which have clausal modifiers and nominalizers.

Fixing the issues that we found in the Paresi-Haliti grammar dramatically reduced ambiguity in both grammars while also raising the validated coverage of the SOV grammar over the test suite from 39% to 50% percent and of the free word order from 56 to 65%, as presented in Table 4. Future work is required for a meaningful comparison of the two different word order hypotheses, though we can observe that the gain in validated coverage is bigger for the SOV grammar.<sup>40</sup>

Hypothesis	Raw coverage (%)	Validated coverage (%)	Overgeneration (%)	Ambiguity
SOV	41/64 (64.1)	36/64 (56.2)	2/3 (66.7)	4.02
free	51/64 (79.7)	42/64 (65.6)	2/3 (66.7)	3.98

Table 4: Improved grammars for Paresi-Haliti

### Long-distance dependencies

4.2

In this section, we discuss a complex interaction between analyses of coordination, adjuncts, and gapped complements. Each analysis has

<sup>40</sup>The higher validated coverage of the free word order grammar does not necessarily mean it is a better hypothesis since the SOV grammar can be developed further such that more orders are covered by the information structure library.

a strong motivation, and they are all relevant for long-distance dependencies. In particular, they all manipulate non-local lists such as SLASH. Taking all the analyses together, non-local lists become over-constrained, incorrectly predicting ungrammaticality for many grammatical sentences. Without computationally implemented grammars, this interaction would almost certainly have gone unnoticed.<sup>41</sup>

Since the early days of HPSG, long-distance dependencies have been analyzed using non-local sets or lists such as SLASH (for a historical overview, see Flickinger *et al.* 2021). The Non-local Feature Principle (Pollard and Sag 1994) states that the value of a non-local feature on the mother is the concatenation of the values on the daughters.

An alternative approach, advocated by Bouma *et al.* (2001a), instead passes (or “threads”) the SLASH values of non-head daughters through the head daughter’s SLASH value. This analysis is particularly attractive for modelling lexical items that take gapped complements (see examples in (15)), as it allows each lexical entry to specify how its SLASH list relates to the SLASH lists of its complements, as in (14).

(14)

$$\left[ \begin{array}{l} \text{lexical threading type} \\ \text{SYNSEM} \left[ \begin{array}{l} \text{NON-LOCAL|SLASH|APPEND} \langle \underline{1}, \underline{2} \rangle \\ \text{LOCAL|CAT|VAL} \left[ \begin{array}{l} \text{SUBJ} \langle \left[ \text{NON-LOCAL|SLASH} \underline{1} \right] \rangle \\ \text{COMPS} \langle \left[ \text{NON-LOCAL|SLASH} \underline{2} \right] \rangle \end{array} \right] \end{array} \right] \end{array} \right]$$

In the majority of cases, the SLASH list of the head is simply the concatenation of the SLASH lists of its arguments, as shown in (14). In cases like *eager* and *easy*, one element is first removed from the complement’s SLASH list. This analysis is known as non-local amalgamation (Bouma *et al.* 2001a; Ginzburg and Sag 2000) or lexical threading.

- (15) a. Kim is eager to please.  
 b. Kim is easy to please.

<sup>41</sup> Specifically, this particular issue was discovered when using the Grammar Matrix in a graduate-level grammar engineering course; see also Section 6.2.

A lexical threading analysis is implemented in the English Resource Grammar (ERG) and was inherited by the Grammar Matrix in its initial construction, using lexical amalgamation of non-local features from arguments and phrasal amalgamation for head-modifier combinations. While integrated into the broad coverage monolingual grammar (ERG), this system was not thoroughly tested in the cross-linguistic Grammar Matrix context until the information structure (Song 2014) and especially the constituent question (Zamaraeva 2021b) libraries were added. On either analysis, the handling of non-local features requires a notion of append (e.g. the SLASH list of the mother is the append of the daughters’). Recall that relational constraints like append are not part of the DELPH-IN variant of the HPSG formalism. The Grammar Matrix initially implemented these appends with difference lists, like the ERG. However, in order to better handle SLASH lists with more than one element as well as for general better maintainability, the Grammar Matrix has moved to using append lists (Zamaraeva and Emerson 2020; see also Section 3.3). We thus had a system which implemented partially lexical and partially phrasal amalgamation of non-local features using append lists, combined with standard HPSG analyses of complementation and modification.

The Grammar Matrix also provides analyses of coordination (Drelshak and Bender 2005), and the interaction between this analysis and the handling of non-local features revealed complexities. To make sure that only compatible phrases can be coordinated, an attractive analysis is to identify large parts of the feature structures for the conjuncts (see Abeillé and Chaves 2021 for a recent review of approaches to coordination in HPSG). However, if these feature structures contain computation types (see Section 3.3), we are identifying not only the outputs of the computation, but the inputs as well. Implementing amalgamation (lexical or phrasal) of non-local features with computation types means that a verb phrase’s SLASH contains not only a list, but also the history of append operations. If we identify not only the lists, but also the computation histories, then we have a much stronger constraint on compatibility for coordination.

For example, consider coordinated intransitive and transitive clauses, as illustrated in (16a). The SLASH list of an intransitive verb like *sleep* is precisely the SLASH list of its subject. However, the SLASH list of a transitive VP like *eat bananas* appends the subject’s SLASH

list with the empty SLASH list of *bananas*. Appending an empty list results in the original list. However, the feature structure associated with a computation type includes all intermediate steps in the computation. With a different number of empty lists being appended, there is a different feature structure.

- (16) a. Monkeys sleep and eat bananas.  
 b. Monkeys sleep soundly and eat bananas.

If we only had a small finite set of valence frames to consider, we could carefully define the append operations, so that the computation histories are compatible for coordination. However, adjuncts pose a problem here, as illustrated in example (16b), because recursive adjunction creates an unbounded set of possible computation histories.

We can see that the analyses of coordination and non-local dependencies are not fully compatible: combining them, the grammar would fail to parse grammatical sentences like (16a–b). We must therefore revise our system of hypotheses. The current approach in the Grammar Matrix involves two changes. First, we only identify the contents of computation types, without their computation histories. This is illustrated in (17), where the SLASH|LIST values are identified, but not the SLASH values. This is sufficient to resolve the problem noted here. In addition, we have dropped the lexical amalgamation of non-local features (although see Section 4.3). This was done in response to this investigation as well as others where the lexical amalgamation approach has made it very difficult to reason about analyses.

$$(17) \left[ \begin{array}{l} \text{coord-phrase} \\ \text{SYNSEM|NON-LOCAL|SLASH|LIST} \\ \text{LCOORD-DTR|SYNSEM|NON-LOCAL|SLASH|LIST} \\ \text{RCOORD-DTR|SYNSEM|NON-LOCAL|SLASH|LIST} \end{array} \right] \begin{array}{l} \boxed{1} \\ \boxed{1} \\ \boxed{1} \end{array}$$

This example illustrates how the Grammar Matrix methodology allows for discovery and resolution of conflicts between analyses and how considerations of maintainability (preferring analyses which are expected to be robust to changes elsewhere in the grammars) also impact analytical decisions.

The study of morphological marking of interrogatives by Zamaraeva (2021a, Section 6.8) in the context of developing a Grammar Matrix library for constituent questions turned up a tension between morphology and syntax in DELPH-IN HPSG.<sup>42</sup> For the full details of this study, we refer the reader to Zamaraeva 2021b and Zamaraeva 2021a, Section 6.8; here we give a brief summary and contextualize the issue with respect to the Grammar Matrix development.<sup>43</sup>

The tension lies between the analysis of long-distance dependencies (see also Section 4.2) and the analysis required for a particular kind of morphological marking of questions found in e.g. Makah [myh] (Wakashan) which maintains two separate verbal inflection paradigms for polar (18a) and for constituent (18b–c) questions (in addition to the indicative paradigm).

- (18) a. *dudu'k* = 'aλ = qa:k = s  
 sing = TEMP = POLAR = 1SG  
 'Am I singing?' [myh] (Davidson 2002, page 100)
- b. *ʔačaq* = qa:ʔ                      *dudu'k*  
 who = CONTENT.3SG sing  
 'Who is singing?' [myh] (Davidson 2002, page 285)
- c. *baqiq* = qa:ʔ                      *ti'*  
 what = CONTENT.3SG DEM  
 'What is this?' [myh] (Davidson 2002, page 285)

In a sense, this particular tension is related to the one presented in Section 4.2: while the syntax of coordination as modelled in the Grammar Matrix made using the non-local amalgamation principle less attractive, the example of Makah points in the opposite direction and suggests that lexical amalgamation of non-local features can still be

<sup>42</sup>The tension lies not in the implementation of the lexical rules but rather in the treatment of non-local features involved in interrogative constructions.

<sup>43</sup>The data and the AVM examples in this section are taken directly from Zamaraeva 2021b. We refer the reader to that work for a complete exposition of the issue summarized here.

very useful in the Grammar Matrix. Specifically, Zamaraeva (2021b) offers two alternative analyses for data such as in (18a–c) and shows that the one that uses lexical amalgamation of non-local features is a lot simpler than the one she developed without lexical amalgamation.<sup>44</sup>

In particular, the analysis which assumes non-local amalgamation relies on a straightforward distinction between two lexical rule types, one for polar and one for constituent questions (19).<sup>45</sup>

- (19) a. 
$$\left[ \begin{array}{l} \textit{polar-lex-rule} \\ \text{SYNSEM|SF} \qquad \textit{ques} \\ \text{DTR|SYNSEM|NON-LOCAL|QUE|LIST} \quad \langle \rangle \end{array} \right]$$
- b. 
$$\left[ \begin{array}{l} \textit{wh-lex-rule} \\ \text{SYNSEM|SF} \qquad \textit{ques} \\ \text{DTR|SYNSEM|NON-LOCAL|QUE|LIST} \quad \textit{cons} \end{array} \right]$$

The QUE list constraint in (19a) means that an affix which is an instance of this lexical rule type cannot apply to something that has a question word as an argument (e.g. subject or complement). Conversely, a word produced with (19b) must have at least one question word as an argument. This works because, under the lexical threading assumption, lexical heads (e.g. verbs) will all be subtypes of a supertype like (14) and will therefore inherit the constraints stated in (14); their own non-local lists, including the QUE lists, will be implicitly constrained to be a concatenation of their arguments' QUE lists.<sup>46</sup>

<sup>44</sup>Zamaraeva was motivated to develop that version for the Grammar Matrix after lexical amalgamation was removed, in light of the issue discussed in Section 4.2.

<sup>45</sup>Both types are subtypes of a more general lexical rule supertype which in turn is part of the lexical rule hierarchy. The lexical rule hierarchy implements the various ways in which affixes can contribute to the form and/or meaning of a word. Here, only the constraints specific to the subtypes for polar and constituent question lexical rules are shown. The feature name SF stands for sentential force; the possible values for this feature include *question* and *proposition*. The type *cons* stands for non-empty list.

<sup>46</sup>The type shown in (14) focuses on SLASH list but the constraint is exactly the same for all NON-LOCAL lists including QUE.



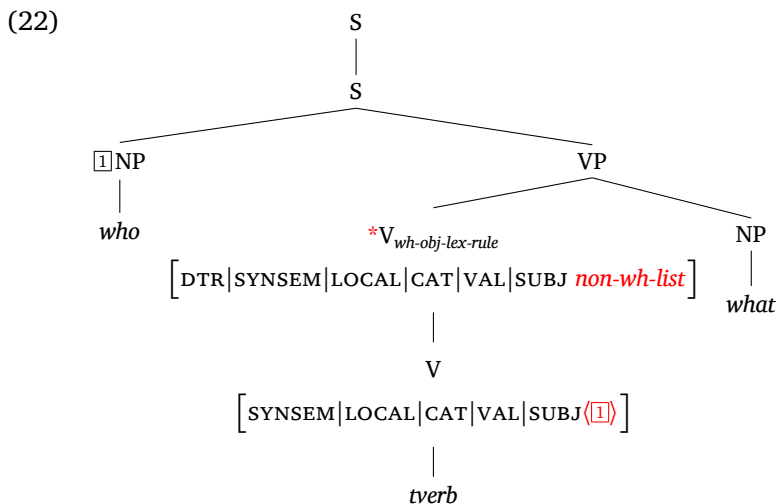
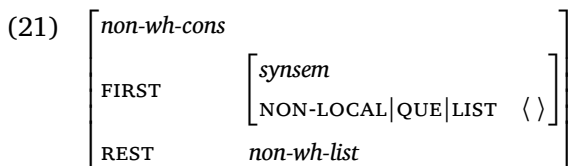
Any question word will have a non-empty QUE list by definition, and thus the head's list will be non-empty also. It is particularly important that for a type to be a subtype of (14) obviates the need to constrain subjects and complements of this type separately as to whether they are question words; it is sufficient to say that the head's QUE list is not empty.

Without the non-local amalgamation assumption, however, heads will not inherit the constraints stated in (14), and it becomes necessary to explicitly constrain the valence lists of heads as to whether they contain question words or not. This necessitates a more complex hierarchy of interrogative lexical rules (shown in (20)) with separate subtypes for cases when a head has a question word as a subject (shown in (20b)) and cases when it has a question word as an object (shown in (20c)), in addition to the subtype for polar questions (shown in (20a)).

- (20) a. 
$$\left[ \begin{array}{l} \text{polar-lex-rule} \\ \text{SYNSEM|LOCAL|CAT|VAL} \left[ \begin{array}{l} \text{SUBJ} \quad \langle \left[ \text{NON-LOCAL|QUE|LIST} \quad \langle \rangle \right] \rangle \\ \text{COMPS} \quad \text{non-wh-list} \end{array} \right] \end{array} \right]$$
- b. 
$$\left[ \begin{array}{l} \text{wh-subj-lex-rule} \\ \text{SYNSEM|LOCAL|CAT|VAL|SUBJ} \quad \langle \left[ \text{NON-LOCAL|QUE|LIST} \quad \text{cons} \right] \rangle \end{array} \right]$$
- c. 
$$\left[ \begin{array}{l} \text{wh-obj-lex-rule} \\ \text{SYNSEM|LOCAL|CAT|VAL} \left[ \begin{array}{l} \text{SUBJ} \quad \text{non-wh-list} \\ \text{COMPS} \quad \langle \left[ \text{NON-LOCAL|QUE|LIST} \quad \text{cons} \right] \rangle \end{array} \right] \end{array} \right]$$

Furthermore, Zamaraeva (2021b) shows that, even with this more complex hierarchy, an additional type is needed to constrain some valence lists either to be empty or to not include any question words (21). This is needed in order to avoid spurious ambiguity in cases where there is more than one question word in the sentence. Without these additional constraints, either lexical rule can apply to license the sole affix needed on the verb. Lacking sufficient data from Makah but based on the description in Davidson 2002, Zamaraeva (2021b) shows how this works on one pseudo-language example (11c), presented here

as (22), where a tree which would otherwise be licensed using (20c) is ruled out, leaving only (20b) as the possibility.



Zamaraeva’s observation points towards the need for further exploration of how much the Grammar Matrix core should cover versus how much should be offloaded to the customization system, or in other words, which parts of the grammatical system we expect to be in every grammar (see also Section 3.4.5). For example, lexical amalgamation of non-local features could be brought back into the Grammar Matrix but not at the level of the core; instead, it would be provided by the customization system only for languages which seem to require it.

Our observation here is that this tension would be hard to notice without the Grammar Matrix which provides the tools to speed up grammar development (e.g. specifying a system of lexical rules, building here in particular on the robustness of the morphotactics library discussed in Section 5) and at the same time embraces a methodology which requires us to examine such a wide range of typological profiles simultaneously and in such formal detail.

*Summary*

4.4

In this section, we presented several examples of how the Grammar Matrix allowed us to discover tensions between different analyses which could have gone unnoticed had we attempted to track all the interactions between all the analyses manually without the means of computer programs for generating grammars and then parsing sentences with those grammars. This ability to computationally verify analyses is what supports the explicitness and empiricism of a grammar engineering approach to linguistic hypothesis testing. Identifying the tensions such as described above efficiently guides future work. Work-in-progress analyses such as the ones described here are thus seen as concrete building blocks which ultimately serve to build a robust system of analyses with an explicit area of applicability.

ACCUMULATING EVIDENCE  
FOR ANALYSES' ROBUSTNESS

5

Fully implemented systems of syntactico-semantic analyses which are deployed in interaction with each other on test suites from diverse languages not only allow us to detect problems in analyses but also to accumulate, over time, a certain confidence that some analyses in fact work well. If a set of hypotheses continues to account for more and more data from more and more languages over time, it is reassuring in terms of the quality of those hypotheses.

In this section, we present some examples of robustness that have been observed in the Grammar Matrix. Section 5.1 gives an example of how a syntactic phenomenon's analysis can be built directly upon an existing analysis of another part of the grammar; Section 5.2 discusses how the system allows for easy reuse of the existing lexical types to introduce new ones; and finally Section 5.3 shows several analyses which relied heavily on the Grammar Matrix's morphotactics library.

## 5.1

*Predicative adjectives in polar questions*

An interaction between the polar questions library and the adjectives library is illustrative of how the Grammar Matrix's analyses, rooted in typological analyses, are robust to unseen interactions between libraries that occur as grammars grow in size.

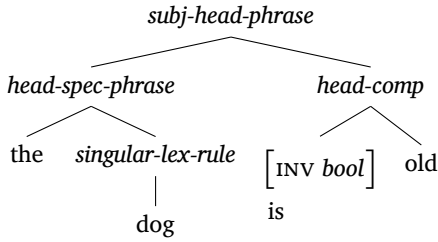
The Grammar Matrix customization system has supported the subject-auxiliary inversion strategy for forming polar questions found in English (e.g. *Is the dog barking?*) since Bender and Flickinger 2005. Poulson's (2011) work on tense/aspect marking and Fokkens's (2010) work on word order further developed the support for auxiliaries. However, the Grammar Matrix did not directly support copulas until Trimble (2014) added them in the context of the adjectives library, as some languages require copulas with predicative adjectives.

Though Trimble's work was done without reference to polar questions in particular, the customization system was able to produce grammars with subject-auxiliary inversion and copulas supporting predicative adjectives with the interaction of these libraries correctly supporting subject-copula inversion with minimal spurious ambiguity that was simple to eliminate. See the examples in (23).

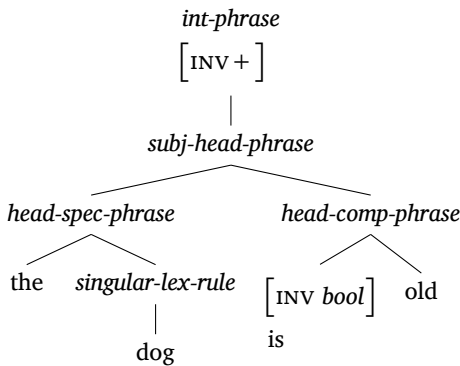
- (23) a. **copula, inverted:** *Is the dog old?* [eng]  
 b. **copula, not inverted:** *The dog is old.* [eng]  
 c. **auxiliary, inverted:** *Is the dog barking?* [eng]  
 d. **auxiliary, not inverted:** *The dog is barking.* [eng]

The copula type introduced by Trimble (2014) was initially underspecified for the feature controlling inversion (INV) and subsequently the inversion phrase structure rule (labelled in (24) as *int* (interrogative)) was spuriously licensed in non-inverted copula phrases, such as *The dog is old*. See both the valid and spurious parses in the simplified schematic set of examples (24). Because the copula is underspecified for INV, even though *int-phrase* bears the constraint [INV +] the spurious parse in (24b) is produced.

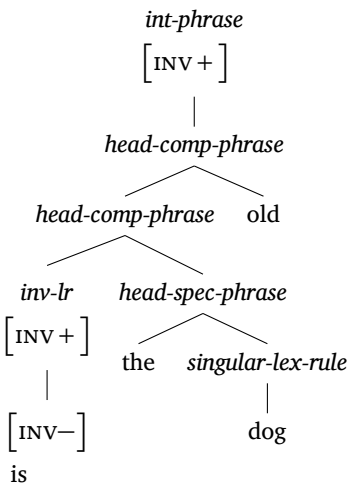
(24) a. valid analysis



b. invalid analysis; spurious application of *int-phrase* rule



c. valid copula inversion analysis



Grammars that licensed the spurious analysis in (24b) were only produced when both polar inversion and copulas were included in the

specification, a scenario not tested during the development of the polar question library, auxiliary support, or copula support. Once the connection between auxiliary inversion and copulas became apparent, it was a simple matter to include copulas in the list of types to which the existing customization logic for the polar question library was adding the [INV –] constraint. The resulting customization process was confirmed to work correctly for languages with inversion; it licensed inverted and non-inverted questions and did not produce the spurious ambiguity in the non-inverted questions.

This sort of analysis of the interaction between libraries with minimal effort demonstrates the robustness of both libraries and their underlying analyses based on the typological and syntactic literature as well as the grammar customization process.

## 5.2 *Reusing and extending the lexical type hierarchy*

The original Grammar Matrix of Bender *et al.* (2002) provided a hierarchy of lexical types, which early users of the Grammar Matrix extended by hand, by either creating lexical entries directly instantiating these types or by creating subtypes and then instantiating them. Drellishak (2009) developed the original web questionnaire and customization logic for the lexicon, which exposed a subset of the core grammar's lexical types through the questionnaire and allowed users to define lexical types in an easier and more abstracted way.

Since then, for over a decade, the lexicon's underlying structure as well as the web interface for adding lexical types and lexical entries have served the development of many new libraries, for the most part requiring only minor extensions. Most extensions have involved merely adding new subtypes (and exposing them via the web interface). From the point of view of HPSG, such developments confirm the generality of the original types. This is not to say that there haven't also been revisions to underlying types, reflecting the ability of the project to refine its analyses over time in a data-driven fashion.

HPSG is a lexicalist theory, which means it assumes a large number of lexical types in any grammar. Over the years, the core lexicon structure (consisting of basic supertypes such as *lex-item*

with a hierarchy of subtypes for different kinds of semantic composition and different valence frames) was successfully extended to support more parts of speech (Trimble 2014; Song 2014; Nielsen 2018).

In addition to refining and extending the lexical type hierarchy in the core grammar, the various libraries have also provided a range of types, accessible via customization at need, which apply in some but not all languages. These include such items as the paired adverbs used in Mandarin [cmn] (Sino-Tibetan) and other languages to express clausal connectives like ‘because’, with one adverb in each clause (Howell and Zamaraeva 2018) and copulas in languages that use them with adjectival predicates (Trimble 2014, Section 4.2).

To take an example of how the existing type hierarchy is reused in more detail, consider the recently added support for question words (Zamaraeva 2021a, Section 6.1) which includes lexical types for question pronouns (such as *who/what* in English), question determiners (such as the English *which*), location in space and time adverbs (like the English *where* and *when*), and morphologically simple question verbs such as the Chukchi *req* which can be roughly translated into English as ‘do what?’.<sup>47</sup> Adding those lexical types was straightforward given the existing Grammar Matrix lexicon framework.

For example, the type for question pronouns is substantially similar to the already existing one for personal pronouns. The constraints shared between the two, shown in (25), model words which are nominal ([HEAD *noun*]), may not serve as modifiers ([MOD < >]), are lexically saturated in their valence (need no dependents as shown by the empty lists as the values of the VAL features), and introduce two predications under the REL feature (the pronoun’s relation and the associated quantifier). The constraints specific to the question pronoun type are shown in (26), and are limited to two: the specific quantifier (*which\_q\_rel*, characteristic of constituent questions) and the non-empty value for QUE, used to detect the presence of these words in both the in-situ and head-filler analyses.

---

<sup>47</sup> Such verbs do not involve incorporation, according to Hagège (2008, page 7).





Section 5.3.1, we give an example of how this analysis straightforwardly interacts with another part of the system, the analysis of adnominal possession.

### *Lexical rules*

5.3

Significant progress has been made since 2010 on the support for syntactic phenomena that can be expressed via morphological marking, including adnominal possession, nominalization, evidentiality, information structure, incorporated adjectives, and more (see Table 1). The Grammar Matrix customization system prompts users to provide position classes (groups of lexical rule types which take the same position within a word), lexical rule types, and lexical rule instances which may specify spelling and, in some cases, contribute to the semantics.<sup>48</sup> The resulting specifications are assembled by the customization system into fairly complex hierarchies of types and sometimes automatically inferred intermediary types for simplifying the type descriptions. The Grammar Matrix's goal in general is to partially automate and therefore speed up grammar development, and its morphotactics library (O'Hara 2008; Goodman 2013) showcases this feature particularly well, as directly writing grammar code for large morphological systems manually would be especially time consuming.

Several significant grammar implementations have tested the limits of the Grammar Matrix morphological systems. Borisova (2010), Bender *et al.* (2014), Crowgey (2019), and the efforts discussed in Section 6.2 have implemented small to large morphological systems using the Grammar Matrix. Bender *et al.* (2012) describe a grammar fragment modelling the lexicon and morphology of Chintang [ctn] (Sino-Tibetan), defined entirely through the customization system, with lexical entries imported automatically from a Toolbox file.<sup>49</sup> This grammar fragment includes 160 lexical rules for verbs and 24 for nouns, hand-defined via the customization system based on the analysis in Schikowski 2012. Crowgey (2019, Section 5) describes a

---

<sup>48</sup> Examples include lexical rules for incorporated adjectives, certain valence changing morphology, and evidentials.

<sup>49</sup> Toolbox is a data management program designed for linguistic work by SIL International (<https://software.sil.org/toolbox/>).

grammar and morphophonological transducer of Lushootseed [lut] (Salishan) where the grammar includes two general position classes with seven lexical rules, two nominal position classes with four lexical rules, and nine verbal position classes with thirty-four lexical rules. Wax (2014, Section 6.1) describes an automatically specified grammar fragment of French [fra] with seventy position classes and twenty-one lexical types described by 938 individual lines in the grammar specification file that the Grammar Matrix outputs.

Specifying such large morphological systems would take significantly longer without the Grammar Matrix morphotactics library with its web-based user interface and abstracted grammar specifications and automatic hierarchy creation. But the morphological support in the Grammar Matrix extends well beyond morphotactics. In this subsection, we briefly survey examples of libraries that build on the morphotactic infrastructure to provide typologically broad support for a range of phenomena: adnominal possession, fine-grained differences between attributive and predicative adjectives, incorporated adjectives, and valence changing morphology.

### 5.3.1

#### Adnominal possession

Nielsen (2018) introduces a Grammar Matrix library for adnominal possession. For languages where the expression of adnominal possession involves affixes indicating features (e.g. person/number) of the possessor, this library makes use of the case library (Drellishak 2009) and the morphotactics library (O'Hara 2008; Goodman 2013). Lexical rule supertypes added to the Grammar Matrix system to support adnominal possession provide an example of how an implemented grammar based on a robust analysis can make correct predictions which humans working without the aid of a computer may overlook.

Zamaraeva (2021a, page 369) describes how, in the process of testing the constituent questions library on a held-out language, Jalkunan [bxl] (Mande), she added to the grammar specification a lexical entry for a question pronoun but was unable to specify an entry for the corresponding possessive pronoun since in the process of development, possessive question pronouns were left unimplemented due to time constraints. However, it turned out that the Grammar Matrix

system already contains everything necessary to produce the correct analysis for the Jalkunan example in question (27), thanks to the adnominal possession library having a lexical rule that can turn a personal pronoun into a possessive pronoun, and thanks to the constituent questions library implementing question pronouns very similarly to how personal pronouns are implemented (see also Section 5.2).

- (27) mā?ā-nĩ    sàá = ∅    nè = ∅  
 who-INDEP house = be there = Q  
 ‘Whose house is that?’ [bxl] (Heath 2017, page 273)

The DMRS<sup>50</sup> artifact in Figure 11 shows the correct semantics for the sentence obtained automatically with the Grammar Matrix-generated grammar: the questioned element is a person (which\_q and person), the person is the possessor of the house (ARG1 of the possession relation is the possessum, ARG2 is the possessor), and the house is located in some space. The importance of this example is that it was not targeted when the grammar specification for Jalkunan was put together, and yet the resulting grammar provided appropriate analyses. This shows that the analysis of adnominal possession (Nielsen 2018) robustly generalizes to constituent questions, even though constituent questions were not part of the Grammar Matrix when the analysis for adnominal possession was developed and tested.

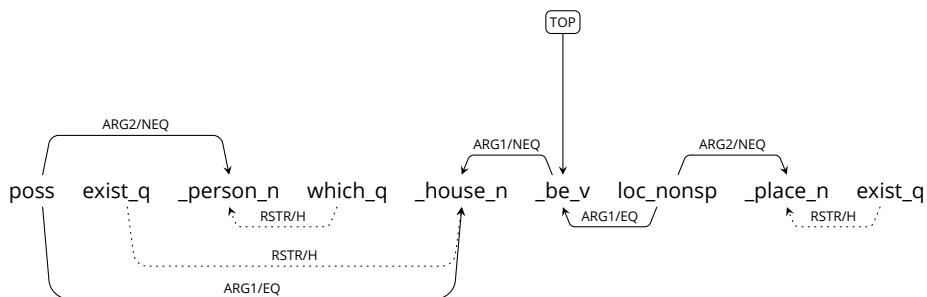


Figure 11: DMRS for sentence (27). The structure can be read as: *Which person possesses the house there?*

<sup>50</sup>DMRS (Copestake 2009) is Dependency MRS, which represents the same information as MRS, but in a dependency graph.

Trimble (2014) introduces a Grammar Matrix library for adjectives in attributive and predicative constructions. The customization system's design to provide custom types in addition to the core grammar proved particularly useful for several adjectival phenomena, enabling a more parsimonious analysis for most languages while accommodating several phenomena that were in conflict with the main analysis, particularly constrained argument agreement and switching adjectives (Stassen 2013).

Languages such as German [deu] (Indo-European) show what Trimble (2014, pages 76–79) calls constrained argument agreement. In these languages, a class of adjectives has one set of morphology in the attributive construction and a different set in a predicative construction. In German, agreement morphology is licensed in the attributive construction and is not licensed in the predicative construction.

- (28) a. Der                    große            Hund bellte.  
 DET.M.NOM.SG big.M.NOM dog bark.PST  
 'The big dog barked.' [deu] (adapted from Hankamer and Lee-Schoenfeld 2005)
- b. Ich bin                    groß.  
 1SG COP.PRES.1SG big  
 'I am tall.' [deu] (adapted from Landman and Morzycki 2003)

gClimb (Fokkens 2014) includes an analysis for this construction where a single position class contains both lexical rules for the agreement morphology used in the attributive construction and a non-inflecting lexical rule which allows the adjective to be the complement of a copula (this rule specifies [PRD +] on the adjective). This way, adjectives can either undergo the predicative lexical rule and be licensed as copula complements or undergo one of the agreement rules and be licensed in the attributive position.

Trimble (2014, pages 74–79) finds a related behaviour of adjectival morphology that Stassen (2013) calls *switching* in languages like Maori [mri] (Austronesian). A single adjective class may be licensed with different sets of syntax and morphology under different circum-

stances, such as one set similar to nouns (see (29b)) and one to verbs (see 29d)).

(29) a. **Nominal predicate:**

he            kiwi teera manu  
DET.INDF kiwi this bird  
'This bird is a kiwi.' [mri] (Biggs 1969, page 90, Stassen 2013)

b. **Noun-like adjectival predicate:**

he    pai    te            koorero  
INDF good DET.DEF talk  
'The talk is good.' [mri] (Biggs 1969, page 14, Stassen 2013)

c. **Verbal predicate:**

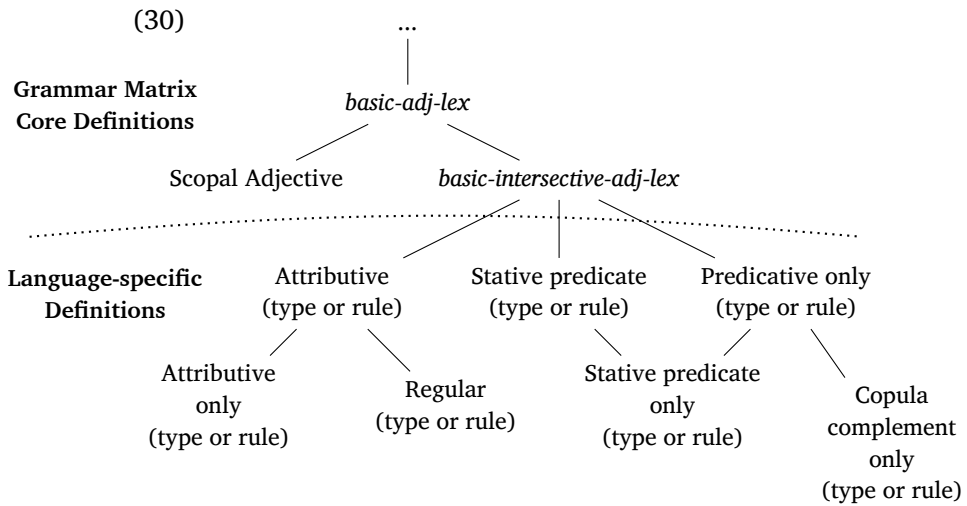
ka    oma    te            kootiro  
INCEP run DET.DEF girl  
'The girl runs.' [mri] (Biggs 1969, page 18, Stassen 2013)

d. **Verb-like adjectival predicate:**

ka    pai    te            whare nei  
INCEP good DET.DEF house this  
'This house is good.' [mri] (Biggs 1969, page 6, Stassen 2013)

The cross-linguistic analysis that is fundamental to the Grammar Matrix leads to the juxtaposition of constrained argument agreement in German and switching adjectives in Maori, offering the opportunity for the same analysis to work in both situations. Like in Fokkens 2014, Trimble's (2014, pages 74–79) analyses of adjectives in languages like German and Maori define adjective types without specifics about their syntactic licensing, such as whether or not the types are licensed as a copula complement or as a matrix predicate, and the syntactic behaviour and morphology are defined through lexical rules. A single required position class is defined with two non-inflecting lexical rule types, one which acts as the base analysis (such as tense marking or being licensed as a copula complement) and one for the other behaviour (such as agreement or being licensed in the attributive construction).

Given the need to support languages like German and Maori, as well as any languages which don't have any morphological reflection of the syntactic distribution of adjectives, the question arises: what should go into the cross-linguistic core grammar? Making a position class like the one posited for German and Maori necessary for all languages would push unwanted complexity into languages where it is not necessary. The solution adopted is to leverage the type hierarchy and the possibility of spreading information between cross-linguistically shared supertypes and language-specific subtypes.



In order to accommodate both constrained argument agreement and switching adjectives, the customization system provides a type hierarchy (shown in (30)) with basic adjectival type definitions (HEAD type, scopal versus intersective adjectives, intersective adjectives modify nouns, etc.) in the core grammar and more specific types (attributive versus predicative, copula complement versus stative predicate, etc.) in the language-specific subtypes. Subsequently, the customization system can provide additional lexical types in the language-specific subtypes if neither constrained argument agreement nor switching adjectives are present; and if either of these phenomena are present, it provides lexical rules which specify the correct syntactic behaviour. The source of the robustness of this approach lies in the combination of the expressive power of the Grammar Matrix's

lexical rule infrastructure alongside the basic architecture of splitting the core grammar and the language-specific subtypes to build custom analyses based on the typological facts of the language.

Adjectives as affixes on nouns

5.3.3

Yup'ik [esu] (Inuit-Yupik-Unangan), Penobscot [aaq-pen] (Algic), and other languages have attributive adjectives that appear as affixes on nouns (see (31a)) usually in addition to adjectives that appear as words (see (31b)) (Miyaoka 2012, page 101, Quinn 2006, pages 28–29). Whereas most morphology either does not add predicates or adds grammatical predicates (such as negation), adjectives are an open class of morpheme with one-to-one morpheme to predicate mapping.

- (31) a. qayar-pa-ngqer-tuq  
kayak-big-have-IND.3SG  
'He has a big kayak.' [esu] (Miyaoka 2012, page 136)
- b. nutaraq                    angyaq            ang'-uq  
new.thing.ABS.SG boat.ABS.SG big-IND.3SG  
'The new boat is big.' [esu] (Miyaoka 2012, page 466)

While implementing adjectival lexemes in the Grammar Matrix required reworking of the lexical hierarchy to support adjectives cross-grammatically (Trimble 2014, pages 76–79), the analysis of incorporated adjectives introduced by Trimble (2014, page 79) required neither changes to the core grammar nor the development of new kinds of questionnaire logic. Rather, it was possible to build it by combining already existing functionality. The existing lexical rule infrastructure (O'Hara 2008; Goodman 2013), both in terms of the core grammar and the customization system, was able to handle this added functionality with no extensions required. Existing functionality in the user interface for specifying predicates on lexical items was used to allow specifying predicates for adjectives as affixes on nouns; and existing customization logic and core grammar functionality resulted in grammar fragments that correctly captured the facts of the language (Trimble 2014, pages 128–129).

Curtis (2018a) describes an implementation of valence-changing morphological operations, including passives, causatives, benefactives, and applicatives found in many language families. These analyses were built upon the existing foundation of argument structure in the Grammar Matrix, primarily the separation of syntactic roles in valence lists and semantic roles in argument slots. Starting from the broad typological groupings of valence-changing morphology as described in Haspelmath and Müller-Bardey 2001, pages 4–14, the implementation focused on decomposing the analyses of these operations into fine-grained atomic rule components. For example, in Zulu the benefactive and motive affixes are homonymous and differ only in the semantic predicate contributed and noun class constraint on the added object (Buell 2005):

(32) a. **Benefactive:**

Ngilahlela	uThandi	udoti
Ngi-lahl-el-a	u-Thandi	u-doti
1S.SBJ-dispose.of-APPL-FV	1-1.Thandi	1-1.trash

‘I’m taking out the trash for Thandi.’ [zul] (Buell 2005, page 189)

b. **Motive:**

Ngilahlela	imali	udoti
Ngi-lahl-el-a	i-mali	u-doti
1S.SBJ-dispose.of-APPL-FV	9-9.money	1-1.trash

‘I’m taking out the trash for money.’ [zul] (Buell 2005, page 189)

The library generates distinct, atomic lexical rule components for these operations, for example specifying *only* the PRED value of the added predicates as in (33a) and (33b). These are then assembled by the customization system – along with the common applicative rule (33c) and added argument rule (33d), which together add the new syntactic argument and connect it to the new semantic predicate – into lexical rule hierarchies that lexical rule instances then inherit from directly.



- (33) a. 
$$\left[ \begin{array}{l} \textit{benefactive-pred-lex-rule} \\ \text{C-CONT | RELS | LIST } \langle \left[ \text{PRED } \textit{benefactive\_rel} \right] \rangle \end{array} \right]$$
- b. 
$$\left[ \begin{array}{l} \textit{motive-pred-lex-rule} \\ \text{C-CONT | RELS | LIST } \langle \left[ \text{PRED } \textit{motive\_rel} \right] \rangle \end{array} \right]$$
- c. 
$$\left[ \begin{array}{l} \textit{basic-applicative-lex-rule} \\ \text{C-CONT } \left[ \begin{array}{l} \text{RELS | LIST } \langle \left[ \textit{event-relation} \right] \rangle \\ \text{HCONS | LIST } \langle \rangle \end{array} \right] \\ \text{DTR | SYNSEM | LOCAL | CONT | HOOK | INDEX } \boxed{1}$$
- d. 
$$\left[ \begin{array}{l} \textit{added-arg3of3-lex-rule} \\ \text{SYNSEM | LOCAL | CAT | VAL | COMPS } \langle \boxed{2}, \left[ \text{LOCAL } \left[ \begin{array}{l} \text{CAT | VAL } \left[ \begin{array}{l} \text{SPR } \langle \rangle \\ \text{COMPS } \langle \rangle \end{array} \right] \\ \text{CONT | HOOK | INDEX } \boxed{1} \end{array} \right] \right] \rangle \\ \text{C-CONT | RELS } \langle \left[ \text{ARG2 } \boxed{1} \right] \rangle \\ \text{DTR | SYNSEM | LOCAL | CAT | VAL | COMPS } \langle \boxed{2} \rangle \end{array} \right]$$

Similar decompositions into blocks were implemented for object-removing, subject-adding, and subject-removing operations, supporting implementation of e.g. (respectively) deobjective, causative, and passive morphology. These phenomena were added to the Grammar Matrix customization system without any additional theoretical additions or machinery: valence-changing operations are specified on position classes and lexical rule types using the existing morphotactics library, and the rule components operate on valence lists, semantic predicates, case marking, and scopal arguments using the existing Grammar Matrix mechanisms. Using these existing mechanisms and stored analyses to create new abstractions at intermediate levels of detail enables grammar engineers to more directly model the commonalities and variations in how valence change is expressed.

5.4

*Summary*

The Grammar Matrix represents a large and complex system of HPSG analyses accounting for data in a wide typological range. Crucially, by rigorously testing said analyses against data from many languages over the years, the project has demonstrated the robustness of the analyses that it houses, particularly of the implementation of the lexical types and the lexical rule types generally and the analyses of specific phenomena relying on those types. In this section, we talked about how it was possible to elegantly extend the system to include various types of adjectives, question pronouns, and valence-changing morphology, to name just several of the phenomena that were successfully added over the years. Crucially, the analyses for all these phenomena demonstrably work together as a system, even in cases when the system engineers did not deliberately consider some of the possible interactions.

6

DISCOVERING COMPLEXITY  
THROUGH USE

Where the data driven methodology described in Section 3.4.1 and the attention to interacting phenomena such as described in Section 4 and Section 5 rigorously test the robustness of the Grammar Matrix’s analyses, the use of the Grammar Matrix outside its own development offers the opportunity to vet these analyses and corresponding implementations at scale. External grammars have the potential to extend the vetting of the Grammar Matrix’s analyses and reveal further unforeseen interactions, by incorporating more phenomena and larger and more complex lexical and morphological systems. Three examples of such use of the Grammar Matrix are: (i) the AGGREGATION project (Bender *et al.* 2014; Zamaraeva *et al.* 2019a; Howell 2020; Howell and Bender 2022, among others), which uses the Grammar Matrix to generate grammars automatically on the basis of linguistic corpora; (ii) an annually offered graduate-level grammar engineering course at University of Washington, in which students use the Grammar Matrix as a starting point and build out the grammars by hand;

and (iii) the broader use of the Grammar Matrix by the research community to develop larger DELPH-IN grammars for linguistic analysis. The sizes of these grammars and their associated test suites, compared with typical Grammar Matrix development grammars, allow the testing of analyses for more interacting phenomena at once. Furthermore, both AGGREGATION and the grammar engineering course place a focus on under-documented languages; and in doing so, provide further evaluation of the Grammar Matrix's analyses in terms of cross-linguistic generalizability.

### *The AGGREGATION project*

6.1

The configurable nature of the Grammar Matrix's grammar specifications makes this toolkit particularly well-suited to support grammar inference. Grammar inference (Bender *et al.* 2014; Zamaraeva *et al.* 2019a) is the practice of automatically generating grammars from partially annotated text and some external source of linguistic knowledge (Howell 2020, page 6; Howell and Bender 2022, page 2).<sup>51</sup> The AGGREGATION Project leverages two sources of linguistic knowledge – interlinear glossed text (IGT) and the Grammar Matrix customization system – to automatically create machine-readable grammars by inferring grammar specifications for the latter from the former.

Unlike grammars created by linguists using the Grammar Matrix customization system directly, grammars inferred by the AGGREGATION Project's morphological inference system (MOM; Wax 2014; Zamaraeva 2016; Zamaraeva *et al.* 2017) and syntactic inference system (BASIL; Howell 2020; Howell and Bender 2022) are brought to scale much more quickly. The rapid scaling offered by automatic inference allows for a degree of complexity that is much more difficult to reach when defining a grammar by hand.

In particular, the MOM morphological inference system infers lexical entries and morphological rules for each form attested in the corpus of IGT data. These lexical entries and rules are merged into larger

---

<sup>51</sup> Howell (2020) and Howell and Bender (2022) define the term *grammar inference* in the context of the AGGREGATION project; while earlier work such as Bender *et al.* 2014 and Zamaraeva *et al.* 2019a refers to grammar inference or describes it as a practice but does not provide a definition.

classes, resulting in grammars that account for a variety of previously unseen strings in a language. However, as Howell (2020, pages 123–128) and Howell and Bender (2022, pages 29–30) observe, such a large set of morphological forms can lead to unforeseen sources of ambiguity. Ambiguity in Howell and Bender’s grammars could be traced back both to inference and to the Grammar Matrix and progress towards reducing this ambiguity was made by Conrad (2021). The discovery of these sources of ambiguity was possible because of the AGGREGATION project’s ability to build large-scale grammars quickly and in turn reveal complexity in language or models of language that would otherwise be difficult to find.

## 6.2 *Grammar Matrix-based grammar development in graduate-level curriculum*

The Grammar Matrix has been used in a graduate-level multilingual grammar engineering course that has been taught annually at the University of Washington since 2004 (see Bender 2007). In this course, students begin by customizing grammars using descriptive resources and the Grammar Matrix customization system; and then continue to build those grammars beyond the customization system’s functionality to achieve greater coverage over the ten week course. Since 2004, over 125 languages have been studied in this course, including many that are the subject of active language documentation projects. The course has served as an important testbed for new Grammar Matrix libraries as they are developed and as a test of the robustness of the system. Each grammar tests at minimum the lexicon and morphological systems; and most exercise the lexicon, morphology, agreement, case, tense-aspect-mood, and coordination libraries. Over the years, each of the other libraries have been tested by multiple languages in connection with the other libraries included in that year’s curriculum. The result has been thorough debugging of Grammar Matrix libraries using combinations of phenomena that were not necessarily considered or tested during library development (such as what we describe in Section 4.2).

To document the analysis that has come from this course, many of the resulting grammar specifications, test suites, and grammar frag-

ments are available via Language CoLLAGE (Bender 2014). In addition, some grammars from the class have resulted in publications focussed on testing specific hypotheses, such as analyzing Kolyma Yukaghir [yux] (isolate) as having focus case (Zamaraeva and Bender 2014) and suggesting a previously unconsidered negation strategy in Nanti [kox] (Arawakan) (Inman and Morrison 2014).

Since 2019, students have used grammar specifications produced by the AGGREGATION system (Section 6.1) as input to the customization system (as opposed to creating the specification from scratch using the Grammar Matrix's web-based questionnaire). The incorporation of the AGGREGATION project in the grammar engineering course enables grammars to include both the lexical and morphological complexity offered by inference from corpora and consequently increases the number of phenomena that can be covered in the course. As a result the potential scale of these grammars increases, allowing for even more testing as the Grammar Matrix continues to grow.

### *Larger DELPH-IN grammars*

6.3

In addition to the relatively small grammars generated by AGGREGATION or developed by students, the Grammar Matrix has given rise to a number of larger grammars and resource grammars. Mid-sized grammars have been developed for the following languages using the Grammar Matrix customization system as a starting point and refining and adding analyses for additional phenomena: Bulgarian [bul] (BURGER; Osenova 2010, 2011), Dutch [nld] (gCLIMB Dutch; Fokkens 2011), Hausa [hau] (HaG; Crysmann 2012), Hebrew [heb] (HeGram; Greshler *et al.* 2015), Indonesian [ind] (INDRA; Moeljadi *et al.* 2015), Lushootseed [lut] (Crowgey 2019), Mandarin [cmn] and Cantonese [yue] Chinese (ManGO; Chunlei and Flickinger 2014, Zhong; Fan *et al.* 2015; Fan 2018), Nuuchahnulth [nuk] (Inman 2019), Thai [tha] (Slayden 2011), and Wambaya [wmb] (Bender 2010). The Grammar Matrix has also given rise to even larger grammars that have undergone long-term development, including for German [deu] (gCLIMB; Fokkens 2011), Korean [kor] (KRG; Kim and Yang 2003; Song *et al.* 2010), Modern Greek [ell] (MGRG; Kordoni and Neu 2005), Norwegian [nob] (NorSource; Hellan and Haugereid 2003), Portuguese [por]

(LXGram; Costa and Branco 2010), and Spanish [spa] (SRG; Marimon 2010).

In addition to the benefits of kick-starting development, these grammars have also benefited from standardization to Minimal Recursion Semantics provided by the Grammar Matrix. DELPH-IN grammars generate MRS representations for sentences which can be used in applications which benefit from semantic representations, as well as to create rich, consistent semantic annotations (see Bender *et al.* 2015, among others). The consistency of MRS representations generated by DELPH-IN grammars is useful for such applications as machine translation (Copestake *et al.* 1995; Bond *et al.* 2011). Even some grammars that did not start by using the Grammar Matrix – such as the Japanese resource grammar (Jacy; Siegel *et al.* 2016) – have incorporated analyses from the Grammar Matrix to maintain consistency of MRS representations with other DELPH-IN grammars. Examination of these grammars and how they have diverged from Grammar Matrix analyses during development poses an interesting area for further work to understand how those analyses hold up at scale.

## 7 CONCLUSION AND FUTURE DIRECTIONS

In this paper, we presented a resource, a system of methodologies, and multiple specific examples of how syntactic hypotheses can be studied rigorously in complex interactions using the HPSG formalism and the Grammar Matrix meta-grammar engineering framework (Bender *et al.* 2002, 2010). We looked back at some of the various contributions made to the Grammar Matrix project since 2010, summarizing the range of phenomena that can currently be modelled and the typological range that the system demonstrably covers.

The first ten years of Grammar Matrix development were characterized by the initial abstraction of the core grammar from the English Resource Grammar, the innovation of the idea of libraries of analyses of cross-linguistically variable phenomena and the orientation to linguistic typology as a key source of data and analyses, and the

development of a regression testing regime for those artifacts. Drelshak's 2009 project began with organizing imagined Grammar Matrix libraries into a (software) dependency tree, leading him to focus on core libraries that others would necessarily depend on (such as case, agreement, and support for the lexicon).

After another decade plus of development, the Grammar Matrix has grown in complexity, with more libraries covering more complex phenomena, allowing for testing more interactions, beginning to realize the promise of linguistic hypothesis testing at a large scale, across languages and ranges of phenomena within those languages. The CLIMB (e.g. Fokkens 2014) and AGGREGATION (e.g. Howell 2020) projects have pushed the boundaries on linguistic hypothesis testing in two further directions: (i) maintaining alternative hypotheses over the course of grammar development and (ii) producing working grammar fragments from the products of field linguistic research.

Where can we expect this project to go in the next decades? There are still many linguistic phenomena to be modelled in Grammar Matrix libraries, including adverbs (other than adverbial clauses and question adverbs), noun compounds, relative clauses, a broader range of valence types (including raising and control phenomena), as well as phenomena which are high frequency in naturally occurring speech collected in language documentation projects such as reported speech, greetings, and discourse markers. There is also near-term work to be done on interactions discovered (and not yet resolved) such as the ones described in Section 4. The addition of computation types to the grammar engineer's toolkit also opens up the possibility for streamlining some existing analyses and making the resulting grammars accordingly easier to maintain. An important direction for future work is seeing how the Grammar Matrix can be informed by its descendant grammars once they are developed beyond the start that the Grammar Matrix provided.

The Grammar Matrix has shown the value of computational implementation to reproducibility in the study of grammar and the concomitant possibility of building directly on the results of others. As the system grows, it also opens up further avenues for study not previously accessible, such as seeking to differentiate which aspects of model complexity are inherent to complexity in the modelled domain and which are consequences of formal or analytical choices.

## ACKNOWLEDGMENTS

Olga Zamaraeva's work is supported by Prof. Gómez Rodríguez's funding from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), from ERDF / MICINN-AEI (TIN2017-85160-C2-1-R, PID2020-113230RB-C21), from Xunta de Galicia (ED431C 2020/11), and from Centro de Investigación de Galicia "CITIC", funded by Xunta de Galicia and the European Union (ERDF – Galicia 2014–2020 Program), by grant ED431G 2019/01.

Grammar Matrix development has been supported by the US National Science Foundation under grants No BCS-0644097, BCS-1160274, and BCS-561833.

We are grateful for the efforts and input of the broader community of Grammar Matrix developers, which includes, beyond the authors of this paper: Elizabeth Conrad, Joshua Crowgey, Laurie Dermer, Scott Drellishak, Chris Evans, Dan Flickinger, Varya Gracheva, Mike Haeger, Scott Halgrim, Joshua Hou, Tom Liu, Daniel P. Mills, Elizabeth Nielsen, Kelly O'Hara, Stephan Oepen, Laurie Poulson, Safiyyah Saleem, Sanghoun Song, David Wax, and Yi Zhang. The work reported here has benefitted greatly from the effort of the past students of Ling 567 at the University of Washington as well as our colleagues in the DELPH-IN Consortium.

We thank the three anonymous reviewers for their constructive criticism and detailed comments.

## REFERENCES

- Anne ABELLÉ and Rui P. CHAVES (2021), Coordination, in Stefan MÜLLER, Anne ABELLÉ, Robert D. BORSLEY, and Jean-Pierre KOENIG, editors, *Head-Driven Phrase Structure Grammar: The handbook*, pp. 723–774, Language Science Press, Berlin, doi:10.5281/zenodo.5599848.
- Peter ACKEMA, Patrick BRANDT, Maaïke SCHOORLEMMER, and Fred WEERMAN, editors (2006), *Arguments and agreement*, Oxford University Press, Oxford.



- Gabriel AGUILA-MULTNER and Berthold CRYSMANN (2018), Feature resolution by lists: The case of French coordination, in Annie FORET, Greg KOBELE, and Sylvain POGODALLA, editors, *Proceedings of International Conference on Formal Grammar*, pp. 1–15, Springer, New York.
- Alexandra AIKHENVALD (2004), *Evidentiality*, Oxford University Press, Oxford.
- Jason BALDRIDGE, Sudipta CHATTERJEE, Alexis PALMER, and Ben WING (2007), DotCCG and VisCCG: Wiki and programming paradigms for improved grammar engineering with OpenCCG, in Tracy Holloway KING and Emily M. BENDER, editors, *Proceedings of the 2007 Workshop on Grammar Engineering Across Frameworks*, pp. 5–25, CSLI Publications, Stanford, CA.
- Kent BECK (2003), *Test-driven development: by example*, Addison-Wesley, Boston, MA.
- Emily M. BENDER (2007), Combining research and pedagogy in the development of a crosslinguistic grammar resource, in Tracy Holloway KING and Emily M. BENDER, editors, *Proceedings of the 2007 Workshop on Grammar Engineering Across Frameworks*, pp. 26–45, CSLI Publications, Stanford, CA.
- Emily M. BENDER (2008), Grammar engineering for linguistic hypothesis testing, in Nicholas GAYLORD, Alexis PALMER, and Elias PONVERT, editors, *Computational Linguistics For Less-studied Languages*, pp. 16–36, CSLI Publications, Stanford, CA.
- Emily M. BENDER (2010), Reweaving a grammar for Wambaya: A case study in grammar engineering for linguistic hypothesis testing, *Linguistic Issues in Language Technology*, 3(3):1–34, doi:10.33011/lilt.v3i.1215.
- Emily M. BENDER (2014), Language CoLLAGE: Grammatical description with the LinGO Grammar Matrix, in Nicoletta CALZOLARI, Khalid CHOUKRI, Thierry DECLERCK, Hrafn LOFTSSON, Bente MAEGAARD, Joseph MARIANI, Asuncion MORENO, Jan ODIJK, and Stelios PIPERIDIS, editors, *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)*, pp. 2447–2451, Paris, [http://www.lrec-conf.org/proceedings/lrec2014/pdf/639\\_Paper.pdf](http://www.lrec-conf.org/proceedings/lrec2014/pdf/639_Paper.pdf).
- Emily M. BENDER, Joshua CROWGEY, Michael Wayne GOODMAN, and Fei XIA (2014), Learning grammar specifications from IGT: A case study of Chintang, in Jeff GOOD, Julia HIRSCHBERG, and Owen RAMBOW, editors, *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pp. 43–53, Baltimore, MD, doi:10.3115/v1/W14-2206, <https://aclanthology.org/W14-2206>.
- Emily M. BENDER, Scott DRELLISHAK, Antske FOKKENS, Laurie POULSON, and Safiyah SALEEM (2010), Grammar customization, *Research on Language and Computation*, 8:1–50, doi:10.1007/s11168-010-9070-1.
- Emily M. BENDER and Guy EMERSON (2021), Computational linguistics and grammar engineering, in Stefan MÜLLER, Anne ABEILLÉ, Robert D. BORSLEY,

and Jean-Pierre KOENIG, editors, *Head-Driven Phrase Structure Grammar: The handbook*, pp. 1103–1150, Language Science Press, Berlin.

Emily M. BENDER and Dan FLICKINGER (2005), Rapid prototyping of scalable grammars: Towards modularity in extensions to a language-independent core, in *International Joint Conference on Natural Language Processing*, pp. 203–208, Jeju, <https://aclanthology.org/I05-2035>.

Emily M. BENDER, Dan FLICKINGER, and 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, and Richard SUTCLIFFE, editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pp. 8–14, Taipei.

Emily M. BENDER, Dan FLICKINGER, Stephan OEPEN, Woodley PACKARD, and Ann COPESTAKE (2015), Layers of interpretation: On grammar and compositionality, in Matthew PURVER, Mehrnoosh SADRADEH, and Matthew STONE, editors, *Proceedings of the 11th International Conference on Computational Semantics*, pp. 239–249, Paris, <https://aclanthology.org/W15-0128>.

Emily M. BENDER, Laurie POULSON, Scott DRELLISHAK, and Chris EVANS (2007), Validation and regression testing for a cross-linguistic grammar resource, in Timothy BALDWIN, Mark DRAS, Julia HOCKENMAIER, Tracy Holloway KING, and Gertjan VAN NOORD, editors, *ACL 2007 Workshop on Deep Linguistic Processing*, pp. 136–143, Prague, doi:10.3115/1608912.1608934, <https://aclanthology.org/W07-1218>.

Emily M. BENDER, Robert SCHIKOWSKI, and Balthasar BICKEL (2012), Deriving a lexicon for a precision grammar from language documentation resources: A case study of Chintang, in Martin KAY and Christian BOITET, editors, *Proceedings of the 24th International Conference on Computational Linguistics*, pp. 247–262, Mumbai, <https://aclanthology.org/C12-1016>.

Manfred BIERWISCH (1963), *Grammatik des deutschen Verbs (Grammar of German verbs)*, Akademie Verlag, Berlin, doi:10.2307/411946.

Bruce BIGGS (1969), *Let's learn Maori*, Reed, Wellington, doi:10.1086/465322.

Francis BOND, Stephan OEPEN, Eric NICHOLS, Dan FLICKINGER, Erik VELLDAL, and Petter HAUGEREID (2011), Deep open-source machine translation, *Machine Translation*, 25(2):87–105.

Irina BORISOVA (2010), *Implementing Georgian polypersonal agreement through the LinGO Grammar Matrix*, Master's thesis, Saarland University.

Gosse BOUMA, Robert MALOUF, and Ivan A. SAG (2001a), Satisfying constraints on extraction and adjunction, *Natural Language and Linguistic Theory*, 19(1):1–65, doi:10.1023/A:1006473306778.

- Gosse BOUMA, Gertjan VAN NOORD, and Robert MALOUF (2001b), Alpino: Wide-coverage computational analysis of Dutch, in Walter DAELEMANS, Khalil SIMA'AN, Jorn VEENSTRA, and Jakub ZAVREL, editors, *Selected papers from the Eleventh Meeting for Computational Linguistics in the Netherlands*, pp. 45–59, Rodopi, Amsterdam, doi:10.1163/9789004333901\_004.
- Ana Paula Barros BRANDÃO (2014), *A reference grammar of Paresi-Haliti (Arawak)*, Ph.D. thesis, University of Texas at Austin.
- Leston Chandler BUELL (2005), *Issues in Zulu verbal morphosyntax*, Ph.D. thesis, University of California, Los Angeles.
- Miriam BUTT and Tracy Holloway KING (2002), Urdu and the Parallel Grammar Project, in *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization*, Taipei, doi:10.3115/1118759.1118762.
- Miriam BUTT, Tracy Holloway KING, María-Eugenia NIÑO, and Frédérique SEGOND (1999), *A grammar writer's cookbook*, CSLI Publications, Stanford, CA.
- Joan L. BYBEE, Revere Dale PERKINS, William PAGLIUCA, et al. (1994), *The evolution of grammar: Tense, aspect, and modality in the languages of the world*, University of Chicago Press, Chicago, IL.
- Daniel BÜRING (2009), Towards a typology of focus realization, in Malte ZIMMERMANN and Caroline FÉRY, editors, *Information Structure: Theoretical, Typological, and Experimental Perspectives*, pp. 177–205, Oxford University Press, Oxford, doi:10.1093/acprof:oso/9780199570959.001.0001.
- Ulrich CALLMEIER (2000), PET – A platform for experimentation with efficient HPSG processing techniques, *Natural Language Engineering*, 6:99–108.
- Marie-Hélène CANDITO (1999), *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien (Modular and parameterized organisation of lexicalised electronic grammars. Applications to French and Italian)*, Ph.D. thesis, Université Paris 7.
- Robert L. CARPENTER (2005), *The logic of typed feature structures: With applications to unification grammars, logic programs and constraint resolution*, Cambridge University Press, Cambridge.
- Yang CHUNLEI and Dan FLICKINGER (2014), ManGO: Grammar engineering for deep linguistic processing, *Data Analysis and Knowledge Discovery*, 30(3):57–64.
- Liviu CIORTUZ and Vlad SAVELUC (2012), *Fluid Construction Grammar and feature constraint logics*, pp. 289–311, Springer, Berlin, doi:10.1007/978-3-642-34120-5\_12.
- Liviu-Virgil CIORTUZ (2002), LIGHT – A constraint language and compiler system for typed-unification grammars, in Matthias JARKE, Jana KOEHLER, and Gerhard LAKEMEYER, editors, *Proceedings of the 25th Annual German Conference on AI*, pp. 3–17, Springer, Berlin, doi:10.1007/3-540-45751-8\_1.

Lionel CLÉMENT and Alexandra KINYON (2003), Generating parallel multilingual LFG-TAG grammars from a MetaGrammar, in Erhard W. HINRICHS and Dan ROTH, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 184–191, Sapporo, doi:10.3115/1075096.1075120.

Bernard COMRIE (1976), *Aspect: An introduction to the study of verbal aspect and related problems*, Cambridge University Press, Cambridge.

Bernard COMRIE (1985), *Tense*, Cambridge University Press, Cambridge.

Elizabeth CONRAD (2021), *Tracing and reducing lexical ambiguity in automatically inferred grammars*, Master's thesis, University of Washington.

Ann COPESTAKE (2002a), Definitions of typed feature structures, in Stephan OEPEN, Dan FLICKINGER, Jun-ichi TSUJII, and Hans USZKOREIT, editors, *Collaborative Language Engineering: A Case Study in Efficient Grammar-Based Processing*, pp. 227–230, CSLI Publications, Stanford, CA.

Ann COPESTAKE (2002b), *Implementing typed feature structure grammars*, CSLI Publications, Stanford, CA.

Ann COPESTAKE (2009), Slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go, in Alex LASCARIDES, Claire GARDENT, and Joakim NIVRE, editors, *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 1–9, Athens, doi:10.3115/1609067.1609167.

Ann COPESTAKE, Dan FLICKINGER, Robert MALOUF, Susanne RIEHEMANN, and Ivan SAG (1995), Translation using minimal recursion semantics, in *Proceedings of the Sixth Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, pp. 15–32, Leuven.

Ann COPESTAKE, Dan FLICKINGER, Carl POLLARD, and Ivan A. SAG (2005), Minimal recursion semantics: An introduction, *Research on Language and Computation*, 3(4):281–332.

Greville G CORBETT (2000), *Number*, Cambridge University Press, Cambridge.

Greville G. CORBETT (2006), *Agreement*, Cambridge University Press, Cambridge.

Francisco COSTA and António BRANCO (2010), LXGram: A deep linguistic processing grammar for Portuguese, in António TEIXEIRA, Vera Lúcia Strube LIMA, Luís Caldas OLIVEIRA, and Paulo QUARESMA, editors, *Computational Processing of the Portuguese Language*, volume 6001 of *Lecture Notes in Artificial Intelligence*, pp. 86–89, Springer, Berlin, doi:10.1007/978-3-642-12320-7\_11.

Benoît CRABBÉ, Denys DUCHIER, Claire GARDENT, Joseph Le ROUX, and Yannick PARMENTIER (2013), XMG: eXtensible MetaGrammar, *Computational Linguistics*, 39(3):591–629, doi:10.1162/COLI\_a\_00144.

- Joshua CROWGEY (2012), An a priori typology of sentential negation from an HPSG perspective, in Stefan MÜLLER, editor, *Proceedings of the 19th International Conference on Head-Driven Phrase Structure Grammar*, pp. 107–122, CSLI Publications, Stanford, CA, doi:10.21248/hpsg.2012.7.
- Joshua CROWGEY (2013), *The syntactic exponence of sentential negation: A model for the LinGO Grammar Matrix*, Master's thesis, University of Washington.
- Joshua CROWGEY (2019), *Braiding language (by computer): Lushootseed grammar engineering*, Ph.D. thesis, University of Washington.
- Berthold CRYSMANN (2012), HaG: A computational grammar of Hausa, in Michael R. MARLO, Nikki B. ADAMS, Christopher R. GREEN, Michelle MORRISON, and Tristan M. PURVIS, editors, *Selected Proceedings of the 42nd Annual Conference on African Linguistics*, pp. 321–337, Cascadilla Press, Somerville, MA.
- Berthold CRYSMANN and Woodley PACKARD (2012), Towards efficient HPSG generation for German, a non-configurational language, in Martin KAY and Christian BOITET, editors, *Proceedings of the 24th International Conference on Computational Linguistics*, pp. 695–710, Mumbai, <http://www.aclweb.org/anthology/C12-1043>.
- Chris CURTIS (2018a), *Valence change library for the Grammar Matrix*, Master's thesis, University of Washington.
- Chris CURTIS (2018b), Valence-changing morphology via lexical rule composition, in Stefan MÜLLER and Frank RICHTER, editors, *Proceedings of the 25th International Conference on Head-Driven Phrase Structure Grammar*, pp. 36–48, CSLI Publications, Stanford, CA, doi:10.21248/hpsg.2018.3.
- Michael CYSOUW (2003), *The paradigmatic structure of person marking*, Oxford University Press, Oxford.
- Glauber Romling DA SILVA (2013), *Morfossintaxe da língua Paresi-Haliti (Arawak) (Morphosyntax of the Paresi-Haliti language (Arawakan))*, Ph.D. thesis, Universidade Federal do Rio de Janeiro.
- Östen DAHL (1979), Typology of sentence negation, *Linguistics*, 17(1–2):79–106.
- Östen DAHL (1985), *Tense and aspect systems*, Basil Blackwell, Oxford.
- Matthew DAVIDSON (2002), *Studies in Southern Wakashan (Nootkan) grammar*, Ph.D. thesis, University of New York at Buffalo.
- Ferdinand DE SAUSSURE (1916), *Cours de linguistique générale (Course in general linguistics)*, Payot, Lausanne–Paris.
- Robert M.W. DIXON (2004), Adjective classes in typological perspective, in Robert M.W. DIXON and Alexandra Y. AIKHENVALD, editors, *Adjective Classes: A Cross-Linguistic Typology*, pp. 1–49, Oxford University Press, Oxford.
- Scott DRELLISHAK (2004), *A survey of coordination strategies in the world's languages*, Master's thesis, University of Washington.

- Scott DRELLISHAK (2008), Complex Case Phenomena in the Grammar Matrix, in Stefan MÜLLER, editor, *Proceedings of the 15th International Conference on Head-Driven Phrase Structure Grammar*, pp. 67–86, CSLI Publications, Stanford, CA.
- Scott DRELLISHAK (2009), *Widespread but not universal: Improving the typological coverage of the Grammar Matrix*, Ph.D. thesis, University of Washington.
- Scott DRELLISHAK and Emily M. BENDER (2005), A coordination module for a crosslinguistic grammar resource, in Stefan MÜLLER, editor, *Proceedings of the 12th International Conference on Head-driven Phrase Structure Grammar*, pp. 108–128, CSLI Publications, Stanford, CA, doi:10.21248/hpsg.2005.6.
- Matthew S. DRYER (2013a), Expression of Pronominal Subjects, in Matthew S. DRYER and Martin HASPELMATH, editors, *The World Atlas of Language Structures Online*, Max Planck Institute for Evolutionary Anthropology, Leipzig, <https://wals.info/chapter/101>.
- Matthew S. DRYER (2013b), Negative Morphemes, in Matthew S. DRYER and Martin HASPELMATH, editors, *The World Atlas of Language Structures Online*, Max Planck Institute for Evolutionary Anthropology, Leipzig, <https://wals.info/chapter/112>.
- Matthew S. DRYER and Martin HASPELMATH, editors (2013), *The World Atlas of Language Structures Online*, Max Planck Institute for Evolutionary Anthropology, Leipzig, <https://wals.info/>.
- Guy EMERSON (2017), (Diff)List appends in TDL, <http://www.delph-in.net/2017/append.pdf>, DELPH-IN summit, Oslo, Norway.
- Guy EMERSON (2019), Wrapper types: Relational constraints without relational constraints, <http://users.sussex.ac.uk/~johnca/summit-2019/wrapper-types.pdf>, DELPH-IN summit, Cambridge, UK.
- Guy EMERSON (2021), On the Turing completeness of typed feature structure unification, <https://raw.githubusercontent.com/delph-in/docs/main/summits/2021/turing.pdf>, DELPH-IN summit, online.
- Guy EMERSON (forthcoming), Computation as subtyping: On the Turing completeness of type systems, with applications to formal grammars.
- Zhenzhen FAN (2018), *Building an HPSG Chinese grammar (Zhong)*, Ph.D. thesis, Nanyang Technological University.
- Zhenzhen FAN, Sanghoun SONG, and Francis BOND (2015), An HPSG-based shared-grammar for the Chinese languages: ZHONG [], in Emily M. BENDER, Lori LEVIN, Stefan MÜLLER, Yannick PARMENTIER, and Arne RANTA, editors, *Proceedings of the 2015 Workshop on Grammar Engineering Across Frameworks*, pp. 17–24, Beijing, doi:10.18653/v1/W15-3303.

Yimai FANG, Haoyue ZHU, Ewa MUSZYŃSKA, Alexander KUHNLE, and Simone TEUFEL (2016), A Proposition-based abstractive summariser, in Yuji MATSUMOTO and Rashmi PRASAD, editors, *Proceedings of the 26th International Conference on Computational Linguistics*, pp. 567–578, Osaka, <https://aclanthology.org/C16-1055>.

Charles J. FILLMORE, Paul KAY, and Mary Catherine O’CONNOR (1988), Regularity and idiomaticity in grammatical constructions: The case of Let Alone, *Language*, 64(3):501–538.

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

Dan FLICKINGER (2011), Accuracy v. robustness in grammar engineering, in Emily M. BENDER and Jennifer E. ARNOLD, editors, *Language from a Cognitive Perspective: Grammar, Usage and Processing*, pp. 31–50, CSLI Publications, Stanford, CA.

Dan FLICKINGER, Carl POLLARD, and Tom WASOW (2021), The evolution of HPSG, in Stefan MÜLLER, Anne ABEILLÉ, Robert D. BORSLEY, and Jean-Pierre KOENIG, editors, *Head-Driven Phrase Structure Grammar: The handbook*, 9, pp. 47–87, Language Science Press, Berlin.

Dan FLICKINGER and Jiye YU (2013), Toward more precision in correction of grammatical errors, in Julia HOCKENMAIER and Sebastian RIEDEL, editors, *Proceedings of the 17th Conference on Computational Natural Language Learning*, pp. 68–73, Sofia.

Antske FOKKENS (2010), Documentation for the Grammar Matrix word order library, [https://github.com/delph-in/docs/wiki/MatrixDoc\\_WordOrder](https://github.com/delph-in/docs/wiki/MatrixDoc_WordOrder).

Antske FOKKENS (2011), Metagrammar engineering: Towards systematic exploration of implemented grammars, in Dekang LIN, editor, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pp. 1066–1076, Portland, OR.

Antske FOKKENS (2014), *Enhancing empirical research for linguistically motivated precision grammars*, Ph.D. thesis, Saarland University.

Antske FOKKENS and Tania AVGUSTINOVA (2013), SlaviCLIMB: Combining expertise for Slavic grammar development using a metagrammar, in *Proceedings of High-level Methodologies for Grammar Engineering*, pp. 87–92, Orleans.

Antske FOKKENS, Tania AVGUSTINOVA, and Yi ZHANG (2012), CLIMB grammars: Three projects using metagrammar engineering, in Nicoletta CALZOLARI, Khalid CHOUKRI, Thierry DECLERCK, Mehmet Uğur DOĞAN, Bente MAEGAARD, Joseph MARIANI, Asuncion MORENO, Jan ODIJK, and Stelios PIPERIDIS, editors, *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pp. 1672–1679, Istanbul.

Antske FOKKENS, Yi ZHANG, and Emily M. BENDER (2011), Spring cleaning and grammar compression: Two techniques for detection of redundancy in HPSG grammars, in Minghui Dong HELENA HONG GAO, editor, *Proceedings of the 25th Pacific Asia Conference on Language, Information and Computation*, pp. 236–244, Tokyo.

Caroline FÉRY and Manfred KRIFKA (2008), Information structure: Notional distinctions, ways of expression, in Piet VAN STERKENBURG, editor, *Unity and Diversity of Languages*, pp. 123–136, John Benjamins, Amsterdam, doi:10.1075/z.141.

Gerald GAZDAR, Ewan KLEIN, Geoffrey PULLUM, and Ivan SAG (1985), *Generalized Phrase Structure Grammar*, Harvard University Press, Cambridge, MA.

Jonathan GINZBURG and Ivan SAG (2000), *Interrogative investigations*, CSLI Publications, Stanford, CA.

Talmy GIVÓN (1994), The pragmatics of de-transitive voice: Functional and typological aspects of inversion, in Talmy GIVÓN, editor, *Voice and Inversion*, pp. 3–44, John Benjamins, Amsterdam.

Michael Wayne GOODMAN (2013), Generation of machine-readable morphological rules from human readable input, *University of Washington Working Papers in Linguistics*, 30, [http://depts.washington.edu/uwupl/vol30/goodman\\_2013.pdf](http://depts.washington.edu/uwupl/vol30/goodman_2013.pdf).

Michael Wayne GOODMAN (2019), A Python library for deep linguistic resources, in Jieh HSIANG and Michael STANLEY-BAKER, editors, *2019 Pacific Neighborhood Consortium Annual Conference and Joint Meetings*, pp. 1–7, Singapore.

Thomas GRAF and Kalina KOSTYSZYN (2021), Multiple wh-movement is not special: The subregular complexity of persistent features in minimalist grammars, in *Proceedings of the Society for Computation in Linguistics*, volume 4, pp. 275–285, ScholarWorks@UMass Amherst, Amherst, MA.

Tali Arad GRESHLER, Livnat HERZIG SHEINFUX, Nurit MELNIK, and Shuly WINTNER (2015), Development of maximally reusable grammars: Parallel development of Hebrew and Arabic grammars, in Stefan MÜLLER, editor, *Proceedings of the 22nd International Conference on Head-Driven Phrase Structure Grammar*, pp. 27–40, CSLI Publications, Stanford, CA, <https://proceedings.hpsg.xyz/article/view/318>.

Michael HAEGER (2017), *An evidentiality library for the LinGO Grammar Matrix*, Master's thesis, University of Washington.

Claude HAGÈGE (2008), Towards a typology of interrogative verbs, *Linguistic Typology*, 12:1–44, doi:10.1515/LITY.2008.031.



Harald HAMMARSTRÖM, Robert FORKEL, Martin HASPELMATH, and Sebastian BANK (2021), Glottolog database 4.5, doi:10.5281/zenodo.5772642, <https://doi.org/10.5281/zenodo.5772642>.

Jorge HANKAMER and Vera LEE-SCHOENFELD (2005), What moves in German VP-fronting, [https://www.researchgate.net/publication/254070408\\_What\\_Moves\\_in\\_German\\_VP-Fronting](https://www.researchgate.net/publication/254070408_What_Moves_in_German_VP-Fronting), handout of presentation at Berkeley Syntax/Semantics Circle.

Martin HASPELMATH, Michael MEEUWIS, APiCS CONSORTIUM, *et al.* (2013), Position of interrogative phrases in content questions, in Susanne Maria MICHAELISC, Philippe MAURER, Martin HASPELMATH, and Magnus HUBER, editors, *The Atlas of Pidgin and Creole Language Structures*, pp. 44–47, Oxford University Press, Oxford.

Martin HASPELMATH and Thomas MÜLLER-BARDEY (2001), Valence change, in G. BOOIJ, C. LEHMANN, and J. MUGDAN, editors, *A Handbook on Inflection and Word Formation*, Walter de Gruyter, Berlin.

Jeffrey HEATH (2017), *A grammar of Jalkunan (Mande, Burkina Faso)*, Language Description Heritage Library, doi:10.17617/2.2346932.

Bernd HEINE (1997), *Possession: Cognitive sources, forces, and grammaticalization*, 83, Cambridge University Press, Cambridge, doi:10.1017/CBO9780511581908.

Lars HELLAN and Petter HAUGEREID (2003), The NorSource grammar – an exercise in the Matrix grammar building design, in Emily M BENDER, Dan FLICKINGER, Frederik FOUVRY, and Melanie SIEGEL, editors, *Proceedings of Workshop on Multilingual Grammar Engineering, ESSLLI 2003*.

Andreas HÖLZL (2018), *A typology of questions in Northeast Asia and beyond: An ecological perspective*, Language Science Press, Berlin.

Kristen HOWELL (2020), *Inferring grammars from Interlinear Glossed Text: Extracting typological and lexical properties for the automatic generation of HPSG grammars*, Ph.D. thesis, University of Washington.

Kristen HOWELL and Emily M. BENDER (2022), Building analyses from syntactic inference in local languages: An HPSG grammar inference system, *Northern European Journal of Language Technology*, 8(1), doi:10.3384/nejlt.2000-1533.2022.4017, <https://nejlt.ep.liu.se/article/view/4017/3515>.

Kristen HOWELL and Olga ZAMARAEVA (2018), Clausal modifiers in the Grammar Matrix, in Emily M. BENDER, Leon DERCZYNSKI, and Pierre ISABELLE, editors, *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2939–2952, Santa Fe, NM.

Kristen HOWELL, Olga ZAMARAEVA, and Emily M. BENDER (2018), Nominalized clauses in the Grammar Matrix, in Stefan MÜLLER and Frank RICHTER, editors, *The 25th International Conference on Head-Driven Phrase Structure Grammar*, CSLI Publications, Stanford, CA, doi:10.21248/hpsg.2018.5.

David INMAN (2019), *Multi-predicate constructions in Nuuchahnulth*, Ph.D. thesis, University of Washington.

David INMAN and Ruth MORRISON (2014), Negation in Nanti: Syntactic evidence for head and dependent negators, in Stefan MÜLLER, editor, *Proceedings of the 21st International Conference on Head-Driven Phrase Structure Grammar*, pp. 103–113, CSLI Publications, Stanford, CA, doi:10.21248/hpsg.2014.6, <https://proceedings.hpsg.xyz/article/view/306>.

Aravind K JOSHI and Yves SCHABES (1997), Tree-Adjoining Grammars, in Grzegorz ROZENBERG and Arto SALOMAA, editors, *Handbook of Formal Languages*, pp. 69–123, Springer, New York.

Ronald M. KAPLAN and Joan BRESNAN (1982), Lexical-Functional Grammar: A formal system for grammatical representation, in Joan BRESNAN, editor, *The Mental Representation of Grammatical Relations*, pp. 173–281, MIT Press, Cambridge, MA.

Marina KHASANOVA and Alexander PEVNOV (2002), *Myths and tales of the Negidals*, Nakanishi, Kyoto.

Jong-Bok KIM and Jaehyung YANG (2003), Korean phrase structure grammar and its implementations into the LKB system, in Dong Hong JI and Kim Teng LUA, editors, *Proceedings of the 17th Pacific Asia Conference on Language, Information and Computation*, pp. 88–97, COLIPS Publications, Singapore.

Valia KORDONI and Julia NEU (2005), Deep analysis of Modern Greek, in Keh-Yih SU, Oi Yee KWONG, Jn'ichi TSUJII, and Jong-Hyeok LEE, editors, *Proceedings of the 2004 International Joint Conference on Natural Language Processing*, pp. 674–683, Springer, Berlin.

Hans-Ulrich KRIEGER and Ulrich SCHÄFER (1994), TDL – A type description language for HPSG, in Makoto NAGAO and Yorick WILKS, editors, *Proceedings of the 15th International Conference on Computational Linguistics*, pp. 893–899, Kyoto.

Meredith LANDMAN and Marcin MORZYCKI (2003), Event-kinds and the representation of manner, in Nancy Mae ANTRIM, Grant GOODALL, Martha SCHULTE-NAFEH, and Vida SAMIAN, editors, *Proceedings of the Western Conference in Linguistics*, volume 11, pp. 1–12, Fresno, CA.

Rob MALOUF, John CARROLL, and Ann COPESTAKE (2002), Efficient feature structure operations without compilation, in Stephan OEPEN, Dan FLICKINGER, Jun-ichi TSUJII, and Hans USZKOREIT, editors, *Collaborative Language Engineering: A Case Study in Efficient Grammar-Based Processing*, pp. 105–125, CSLI Publications, Stanford, CA.

Montserrat MARIMON (2010), The Spanish Resource Grammar, in Nicoletta CALZOLARI, Khalid CHOUKRI, Bente MAEGAARD, Joseph MARIANI, Jan

ODIJK, Stelios PIPERIDIS, Mike ROSNER, and Daniel TAPIAS, editors, *Proceedings of the 7th International Conference on Language Resources and Evaluation*, Valletta, [http://www.lrec-conf.org/proceedings/lrec2010/pdf/602\\_Paper.pdf](http://www.lrec-conf.org/proceedings/lrec2010/pdf/602_Paper.pdf).

Detmar MEURERS, Gerald PENN, and Frank RICHTER (2002), A web-based instructional platform for constraint-based grammar formalisms and parsing, in Dragomir RADEV and Chris BREW, editors, *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics*, pp. 19–26, Philadelphia, PA.

Yusuke MIYAO and Jun'ichi TSUJII (2008), Feature forest models for probabilistic HPSG parsing, *Computational Linguistics*, 34(1):35–80.

Osahito MIYAOKA (2012), *A grammar of Central Alaskan Yupik (CAY)*, Walter de Gruyter, Berlin, doi:10.1515/9783110278576.

David MOELJADI, Francis BOND, and Sanghoun SONG (2015), Building an HPSG-based Indonesian resource grammar (INDRA), in Emily M. BENDER, Lori LEVIN, Stefan MÜLLER, Yannick PARMENTIER, and Aarne RANTA, editors, *Proceedings of the 2015 Workshop on Grammar Engineering Across Frameworks*, pp. 9–16, Beijing, doi:10.18653/v1/W15-3302.

Richard MONTAGUE (1974), *Formal philosophy: Selected papers of Richard Montague*, Yale University Press, New Haven, CT.

Luís MORGADO DA COSTA, Francis BOND, and Xiaoling HE (2016), Syntactic well-formedness diagnosis and error-based coaching in computer assisted language learning using machine translation, in Hsin-Hsi CHEN, Yuen-Hsien TSENG, Vincent NG, and Xiaofei LU, editors, *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications*, pp. 107–116, Osaka.

Luis MORGADO DA COSTA, Roger WINDER, Shu Yun LI, Benedict Christopher Lin Tzer LIANG, Joseph MACKINNON, and Francis BOND (2020), Automated writing support using deep linguistic parsers, in Nicoletta CALZOLARI, Frédéric BÉCHET, Philippe BLACHE, Khalid CHOUKRI, Christopher CIERI, Thierry DECLERCK, Sara GOGGI, Hitoshi ISAHARA, Bente MAEGAARD, Joseph MARIANI, Hèlène MAZO, Asuncion MORENO, Jan ODIJK, and Stelios PIPERIDIS, editors, *Proceedings of the 12th Language Resources and Evaluation Conference*, pp. 369–377, Marseille.

Sarah E. MURRAY (2017), *The semantics of evidentials*, Oxford University Press, Oxford, doi:10.1093/oso/9780199681570.001.0001.

Stefan MÜLLER (1999), *Deutsche Syntax deklarativ: Head-Driven Phrase Structure Grammar für das Deutsche (Declarative German Syntax: Head-Driven Phrase Structure Grammar for German)*, Max Niemeyer Verlag, Tübingen, doi:10.1515/9783110915990.

- Stefan MÜLLER (2007), The Grammix CD-ROM: A software collection for developing typed feature structure grammars, in Tracy Holloway KING and Emily M. BENDER, editors, *Proceedings of the 2007 Workshop on Grammar Engineering Across Frameworks*, pp. 259–266, CSLI Publications, Stanford, CA, <http://csli-publications.stanford.edu/GEAF/2007/>.
- Stefan MÜLLER (2015), The CoreGram project: Theoretical linguistics, theory development and verification, *Journal of Language Modelling*, 3(1):21–86, doi:10.15398/jlm.v3i1.91.
- Stefan MÜLLER, Anne ABEILLÉ, Robert D. BORSLEY, and Jean-Pierre KOENIG, editors (2021), *Head Driven Phrase Structure Grammar: The handbook*, Language Science Press, Berlin, doi:10.5281/zenodo.5543318.
- Elizabeth NIELSEN and Emily M. BENDER (2018), Modeling adnominal possession in multilingual grammar engineering, in Stefan MÜLLER and Frank RICHTER, editors, *Proceedings of the 25th International Conference on Head-Driven Phrase Structure Grammar*, pp. 140–153, CSLI Publications, Stanford, CA, doi:10.21248/hpsg.2018.9.
- Elizabeth K NIELSEN (2018), *Modeling adnominal possession in the linGO Grammar Matrix*, Master’s thesis, University of Washington.
- Michael NOONAN (2007), Complementation, in Timothy SHOPEN, editor, *Language Typology and Syntactic Description*, pp. 52–150, Cambridge University Press, Cambridge, 2nd edition, doi:doi.org/10.1017/CBO9780511619434.
- Stephan OEPEN (2002), *Competence and performance profiling for constraint-based grammars: A new methodology, toolkit, and applications*, Ph.D. thesis, Universität des Saarlandes.
- Stephan OEPEN, Emily M. BENDER, Ulrich CALLMEIER, Dan FLICKINGER, and Melanie SIEGEL (2002), Parallel distributed grammar engineering for practical applications, in *Proceedings of the Workshop on Grammar Engineering and Evaluation*, Taipei.
- Stephan OEPEN, Dan FLICKINGER, Kristina TOUTANOVA, and Christopher D. MANNING (2004), LinGO Redwoods: A rich and dynamic treebank for HPSG, *Research on Language and Computation*, 2(4):575–596, doi:10.1007/s11168-004-7430-4.
- Stephan OEPEN and Daniel P. FLICKINGER (1998), Towards systematic grammar profiling: Test suite technology ten years after, *Journal of Computer Speech and Language*, 12(4):411–435, doi:10.1006/csla.1998.0105.
- Stephan OEPEN, Erik VELLDAL, Jan Tore LØNNING, Paul MEURER, Victoria ROSÉN, and Dan FLICKINGER (2007), Towards hybrid quality-oriented machine translation. On linguistics and probabilities in MT, in Andy WAY and Barbara GAWRONSKA, editors, *The 11th International Conference on Theoretical and Methodological Issues in Machine Translation*, Skövde, Sweden, <https://aclanthology.org/2007.tmi-papers>.

- Kelly O'HARA (2008), *A morphotactic infrastructure for a grammar customization system*, Master's thesis, University of Washington.
- Petya OSENOVA (2010), BURGER: Bulgarian Resource Grammar – Efficient and Realistic, <http://bultreebank.org/en/bulgarian-resource-grammar-burger/>, technical report, Stanford, CSLI.
- Petya OSENOVA (2011), Localizing a core HPSG-based grammar for Bulgarian, in Hanna HEDELAND, Thomas SCHMIDT, and Kai WÖRNER, editors, *Proceedings of the Conference of the German Society for Computational Linguistics and Language Technology*, pp. 175–182, Hamburg.
- Woodley PACKARD (2015), *Full forest treebanking*, Master's thesis, University of Washington.
- Barbara H. PARTEE (1979), Constraining Montague grammar: A framework and a fragment, in S. DAVIS and M. MITHUN, editors, *Linguistics, Philosophy, and Montague Grammar*, pp. 51–101, University of Texas Press, Austin, TX, doi:10.7560/746251-004.
- Doris L. PAYNE and Immanuel BARSHI, editors (1999), *External Possession*, John Benjamins Publishing Co., Amsterdam, doi:10.1075/tsl.39.
- John R PAYNE (1985), Complex phrases and complex sentences, in Timothy SHOPEN, editor, *Language typology and syntactic description*, volume 2, pp. 3–41, Cambridge University Press, Cambridge, 1st edition, doi:doi.org/10.1017/CBO9780511619434.
- Gerald PENN (2000), Applying constraint handling rules to HPSG, in Thom FRÜHWIRTH, editor, *Proceedings of the Workshop on Rule-Based Constraint Reasoning and Programming*, <http://www.informatik.uni-ulm.de/pm/fileadmin/pm/home/fruehwirth/cl2000r.html>.
- Gerald PENN (2004), Balancing clarity and efficiency in typed feature logic through delaying, in Donia SCOTT, editor, *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pp. 239–246, Barcelona, doi:10.3115/1218955.1218986.
- Carl POLLARD and Ivan A. SAG (1987), *Information-based syntax and semantics*, CSLI Publications, Stanford, CA.
- Carl POLLARD and Ivan A. SAG (1994), *Head-Driven Phrase Structure Grammar*, University of Chicago Press, Chicago, IL.
- Laurie POULSON (2011), Meta-modeling of tense and aspect in a cross-linguistic grammar engineering platform, in Sanghoun SONG and Joshua CROWGEY, editors, *University of Washington Working Papers in Linguistics*, volume 28, pp. 1–62.
- Conor QUINN (2006), *Referential-access dependency in Penobscot*, Ph.D. thesis, Harvard University.

Aarne RANTA (2011), *Grammatical framework: Programming with multilingual grammars*, CSLI Publications, Stanford, CA.

Frank RICHTER (2021), Formal Background, in Stefan MÜLLER, Anne ABEILLÉ, Robert D. BORSLEY, and Jean-Pierre KOENIG, editors, *Head-Driven Phrase Structure Grammar: The handbook*, pp. 89–124, Language Science Press, Berlin, doi:DOI:10.5281/zenodo.5599822.

Ivan A. SAG, Hans C. BOAS, and Paul KAY (2012), Introducing Sign-Based Construction Grammar, in Hans C. BOAS and Ivan A. SAG, editors, *Sign-Based Construction Grammar*, pp. 1–29, CSLI Publications, Stanford, CA.

Safiyah SALEEM (2010), *Argument optionality: A new library for the Grammar Matrix customization system*, Master's thesis, University of Washington.

Safiyah SALEEM and Emily M. BENDER (2010), Argument optionality in the LinGO Grammar Matrix, in Chu-Ren HUANG and Dan JURAFSKY, editors, *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 1068–1076, Beijing.

Robert SCHIKOWSKI (2012), Chintang morphology, unpublished manuscript, University of Zürich.

Melanie SIEGEL, Emily M. BENDER, and Francis BOND (2016), *Jacy: An implemented grammar of Japanese*, CSLI Publications, Stanford, CA, <http://web.stanford.edu/group/cslipublications/cslipublications/site/9781684000180.shtml>.

Anna SIEWIERSKA (2004), *Person*, Cambridge University Press, Cambridge.

Glenn C SLAYDEN (2011), Thai Grammar, available at <http://agree-grammar.com/src/grammars/thai>. Accessed 05-20-2022.

Glenn C SLAYDEN (2012), *Array TFS storage for unification grammars*, Master's thesis, University of Washington.

Sanghoun SONG (2014), *A grammar library for information structure*, Ph.D. thesis, University of Washington.

Sanghoun SONG, Jong-Bok KIM, Francis BOND, and Jaehyung YANG (2010), Development of the Korean resource grammar: Towards grammar customization, in *Proceedings of the 8th Workshop on Asian Language Resources*, pp. 144–152.

Edward STABLER (1997), Derivational minimalism, in Christian RETORÉ, editor, *Proceedings of the International Conference on Logical Aspects of Computational Linguistics*, pp. 68–95, Springer, New York.

Leon STASSEN (1997), *Intransitive predication*, Clarendon Press, Oxford.

Leon STASSEN (2000), AND-languages and WITH-languages, *Linguistic Typology*, 4:1–54.

Leon STASSEN (2013), Predicative adjectives, in Matthew S. DRYER and Martin HASPELMATH, editors, *The World Atlas of Language Structures Online*, Max Planck Institute for Evolutionary Anthropology, Leipzig, <http://wals.info/chapter/118>.

Mark STEEDMAN (2000), *The syntactic process*, MIT Press, Cambridge, MA.

Patrick SUPPES, Tie LIANG, Elizabeth E. MACKEN, and Dan 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 and Education*, 71:23–32, doi:10.1016/j.compedu.2013.09.008.

Sandra A THOMPSON, Robert E LONGACRE, and Shin Ja J HWANG (1985), Adverbial clauses, in Timothy SHOPEN, editor, *Language Typology and Syntactic Description*, pp. 171–234, Cambridge University Press, Cambridge, 1st edition, doi:doi.org/10.1017/CBO9780511619434.

John TORR (2018), Constraining MGBank: Agreement, L-selection and supertagging in Minimalist Grammars, in Iryna GUREVYCH and Yusuke MIYAO, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 590–600, Melbourne, doi:10.18653/v1/P18-1055, <https://aclanthology.org/P18-1055>.

Thomas TRIMBLE (2014), *Adjectives in the LinGO Grammar Matrix*, Master's thesis, University of Washington.

Hans USZKOREIT, Rolf BACKOFEN, Stephan BUSEMANN, Abdel Kader DIAGNE, Elizabeth A. HINKELMAN, Walter KASPER, Bernd KIEFER, Hans-Ulrich KRIEGER, Klaus NETTER, Gunter NEUMANN, Stephan OEPEN, and Stephen P. SPACKMAN (1994), DISCO – An HPSG-based NLP system and its application for appointment scheduling, in Makoto NAGAO and Yorik WILKS, editors, *Proceedings of the 15th International Conference on Computational Linguistics*, Kyoto, <https://aclanthology.org/C94-1072>.

Gertjan VAN NOORD (2006), At Last Parsing Is Now Operational (APLINO), in Piet MERTENS, Cédric FAIRON, Anne DISTER, and Patrick WATRIN, editors, *Actes de la 13ème Conférence sur le Traitement Automatique des Langues Naturelles*, pp. 20–42, l'Association pour Traitement Automatique des Langues, Leuven, <http://talnarchives.atala.org/TALN/TALN-2006/taln-2006-invite-002.pdf>.

David WAX (2014), *Automated grammar engineering for verbal morphology*, Master's thesis, University of Washington.

Olga ZAMARAEVA (2016), Inferring morphotactics from interlinear glossed text: Combining clustering and precision grammars, in Micha ELSNER and Sandra KUEBLER, editors, *Proceedings of the 14th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pp. 141–150, Berlin, doi:10.18653/v1/W16-2021.

Olga ZAMARAEVA (2021a), *Assembling syntax: Modeling constituent questions in a grammar engineering framework*, Ph.D. thesis, University of Washington.

Olga ZAMARAEVA (2021b), Morphological marking of constituent questions: A case for nonlocal amalgamation, in Stefan MÜLLER and Nurit MELNIK, editors, *Proceedings of the 28th International Conference on Head-Driven Phrase Structure Grammar*, pp. 241–261, Frankfurt/Main, doi:10.21248/hpsg.2021.13.

Olga ZAMARAEVA and Emily M. BENDER (2014), Focus case outside of Austronesian: An analysis of Kolyma Yukaghir, in Stefan MÜLLER, editor, *Proceedings of the 21st International Conference on Head-Driven Phrase Structure Grammar*, pp. 176–196, CSLI Publications, Stanford, CA, doi:10.21248/hpsg.2014.10, <https://proceedings.hpsg.xyz/article/view/310>.

Olga ZAMARAEVA and Guy EMERSON (2020), Multiple question fronting without relational constraints: An analysis of Russian as a basis for cross-linguistic modeling, in Stefan MÜLLER and Anke HOLLER, editors, *Proceedings of the 27th International Conference on Head-Driven Phrase Structure Grammar*, pp. 157–177, CSLI Publications, Stanford, CA, doi:10.21248/hpsg.2020.9.

Olga ZAMARAEVA, Kristen HOWELL, and Emily M. BENDER (2019a), 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*, Honolulu, HI.


Olga ZAMARAEVA, Kristen HOWELL, and Emily M. BENDER (2019b), Modeling clausal complementation for a grammar engineering resource, in Gaja JAROSZ, Max NELSON, Brendan O’CONNOR, and Joe PATER, editors, *Proceedings of the 2nd Meeting of the Society for Computation in Linguistics*, pp. 39–49, ScholarWorks@UMass Amherst, Amherst, MA, doi:10.7275/dygn-c796.

Olga ZAMARAEVA, František KRATOCHVÍL, Emily M. BENDER, Fei XIA, and Kristen HOWELL (2017), Computational support for finding word classes: A case study of Abui, in Antti ARPPE, Jeff GOOD, Mans HULDEN, Jordan LACHLER, Alexis PALMER, and Lane SCHWARTZ, editors, *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pp. 130–140, Honolulu, HI, doi:10.18653/v1/W17-0118.

Arnold M. ZWICKY, Joyce FRIEDMAN, Barbara C. HALL, and Donald E. WALKER (1965), The MITRE syntactic analysis procedure for Transformational Grammars, in Robert W. RECTOR, editor, *AFIPS Conference Proceedings*, volume 27, pp. 317–326, Spartan Books, Washington, D.C., doi:10.1145/1463891.1463928.




*Olga Zamaraeva*

 0000-0001-9969-058X


Facultade de Informática  
Universidade da Coruña  
Rúa da Maestranza 9,  
15001 A Coruña, Spain

*Chris Curtis*

 0000-0002-3499-466X


Firemuse Research  
Rockville, Maryland, USA

*Guy Emerson*

 0000-0002-3136-9682


University of Cambridge  
Cambridge, UK

*Antske Fokkens*

 0000-0002-6628-6916


Vrije Universiteit & Eindhoven  
University of Technology  
Amsterdam & Eindhoven, The  
Netherlands

*Michael W. Goodman*

 0000-0002-2896-5141


LivePerson Inc.  
Seattle, Washington, US

*Kristen Howell*

 0000-0002-4564-055X


LivePerson Inc.  
Seattle, Washington, US  
  
University of Washington  
Seattle, Washington, USA

*T.J. Trimble*

 0000-0002-1605-2177


Independent scholar

*Emily M. Bender*


 0000-0001-5384-6227

University of Washington  
Seattle, Washington, USA

Olga Zamaraeva, Chris Curtis, Guy Emerson, Antske Fokkens, Michael Wayne Goodman, Kristen Howell, T.J. Trimble, and Emily M. Bender (2022), *20 years of the Grammar Matrix: cross-linguistic hypothesis testing of increasingly complex interactions*, *Journal of Language Modelling*, 10(1):49–137

 <https://dx.doi.org/10.15398/jlm.v10i1.292>

This work is licensed under the *Creative Commons Attribution 4.0 Public License*.

 <http://creativecommons.org/licenses/by/4.0/>