Chapter 23

Computational implementations and applications

Martin Forst
Cerence Inc.

Tracy Holloway King
Adobe Inc.

Computational implementations of LFG are computer programs composed of LFG annotated c-structure rules and lexical entries. LFG was designed to be computationally tractable and has a strong history of broad-coverage grammar implementations for diverse languages. As with theoretical LFG, implemented grammars primarily focus on c-structure and f-structure, but the resulting f-structures are used as input to semantics and abstract knowledge representation, and some work has focused on the integration of morphological and phonological information as well as argument structure. From a theoretical linguistic perspective, implemented grammars allow the linguist to test analyses and to see interactions between different parts of the grammar. From an application perspective, applications such as machine translation and question answering take advantage of the abstract f-structures and the ability of LFG grammars to parse and generate as well as to detect (un)grammaticality.

Computational implementations of LFG are computer programs composed of LFG annotated c-structure rules and lexical entries. When parsing, they take as input a natural language sentence and output c-structures and f-structures and potentially other projections such as semantics. When generating, they take as input an f-structure and generate a grammatical natural language sentence. As with theoretical LFG, these implemented grammars obey the fundamental premises of LFG such as completeness, coherence, and uniqueness.[1] LFG was de-

[1]This contrasts with approaches which produce f-structure-like representations but do not use LFG principles or machinery. See Section 3 and Cahill et al. 2002.

Martin Forst & Tracy Holloway King. 2023. Computational implementations and applications. In Mary Dalrymple (ed.), *Handbook of Lexical Functional Grammar*, 1083–1123. Berlin: Language Science Press. DOI: 10.5281/zenodo.10185986

signed from the outset to be computationally tractable and has a strong history of broad-coverage implementations for multiple languages, primarily through the ParGram project (Butt, King, et al. 1999) which is built on the XLE grammar development platform (Crouch et al. 2011).

Grammar engineering involves the implementation of linguistically-motivated grammars so that natural language utterances and text can be processed to produce deep syntactic, and sometimes semantic, structures. As with theoretical LFG, implemented grammars primarily focus on c-structure and f-structure. The resulting f-structures have been used extensively as input to semantics and abstract knowledge representation. Other work has focused on the integration of morphological and phonological information, as well as argument structure, but in general these areas have lagged behind the proposals in the theoretical literature. In addition, implemented LFG grammars have been used to create large-scale tree and dependency banks, mapping a corpus of sentences to a set of f-structures or related dependency structures.

We first introduce the computational implementations of LFG, presenting specific platforms and touching upon aspects such as core components, grammar development tools, modularity, and runtime performance (Section 1). We then discuss implications for theoretical issues (Section 2) and the ParGram grammar resources (Section 3). Finally, we outline existing and potential applications for LFG implemented grammars (Section 4).

1 Computational implementations

Computational implementations of LFG grammars focus on annotated phrase structure rules and lexical entries. These implementations concentrate on creating high-quality f-structures since most applications use f-structures as their input (Section 4). This section first introduces the major platforms that support LFG implementations. The core components provided by these platforms are then outlined, followed by some specific grammar development tools. Finally two computational notions, modularity and performance, are discussed.

1.1 Platforms

Since the inception of LFG as a grammar framework several platforms aimed at processing text according to the LFG formalism have been created. These platforms range from an M.Sc. project (Minos 2014) and introductory French implementation (Zweigenbaum 1991) to an industrially funded grammar development and processing platform which was actively developed for over two decades: the Xerox Linguistic Environment (XLE). In between those in terms of breadth of applicability and technical maturity are systems developed in academic research institutions, in particular XLFG, SxLFG, and the Free Linguistic Environment (FLE). Active development on many of these systems is limited: for current status and documentation the platform owners should be consulted.

1.1.1 XLFG and Elvex

XLFG (Clément & Kinyon 2001) is a parsing platform that was first implemented for didactic purposes.[2] It has been used to verify the soundness of several proposals to handle a variety of linguistic phenomena (Section 2), e.g. zeugmas, particle verbs, and non-constituent coordination (Clément 2019).

XLFG uses an Earley parser (Earley 1970) for context-free parsing, and then resolves the f-structure constraints on packed c-structure representations (Maxwell & Kaplan 1989, 1993). It expects tokenized sentences as input and uses full-form lexicons for lexical lookup (Section 1.2). XLFG does not facilitate the use of external components like finite-state transducers for preprocessing tasks such as tokenization or morphological analysis (Section 1.2). It has primarily been applied to parsing French and English, i.e. analyzing French or English text into f-structures. Recently, work was started on a generator, i.e. mapping f-structures to text, using XLFG-style grammars for the production of surface realizations from f-structures. This generator is named Elvex.[3]

1.1.2 SxLFG

SxLFG (Boullier & Sagot 2005) was also developed with the participation of Lionel Clément, but its main authors are Pierre Boullier and Benoît Sagot of INRIA. The primary focus of SxLFG is on the deep non-probabilistic parsing of large corpora (Sagot & Boullier 2006) by means of robustness techniques for input sentences for which no spanning c-structure can be produced. The underlying context-free parser is the Earley parser of the SYNTAX project. Like XLFG and XLE, SxLFG resolves f-structure constraints on packed c-structure representations. The French broad-coverage LFG implementation that has been used extensively with SxLFG includes a large full-form lexicon for French, the Lefff 2 (Sagot et al. 2006). Like XLFG, SxLFG does not facilitate the use of external components

[2]XLFG is available at http://www.xlfg.org

[3]Elvex is available at https://github.com/lionelclement/Elvex

like finite-state transducers for preprocessing tasks such as tokenization or morphological analysis (Section 1.2). SxLFG was developed for parsing. Generation has not been in the scope of SxLFG.

1.1.3 XLE (and GWW as precursor) and XLE-Web

The Xerox Linguistic Environment (XLE) was developed by the Natural Language Theory and Technology (NLTT) group at the Xerox Palo Alto Research Center (PARC). It started as a reimplementation in C of the earlier Grammar Writer's Workbench (GWW) (Kaplan & Maxwell 1996), which was implemented in Lisp and is still available. XLE was used by several academic and industry teams for the development of LFG implementations for more than a dozen languages (see Section 3 on the ParGram project). XLE, in conjunction with a customized broad-coverage grammar, was used to parse the English Wikipedia in the Powerset search engine (Kaplan 2009).

XLE has mostly been used for parsing, but it includes a generator that can efficiently produce surface realizations from f-structures and even packed f-structure charts (Maxwell 2006). Thanks to this bidirectionality, it has powered applications such as machine translation and sentence condensation (Section 4), and it has been used in research projects on stochastic realization ranking (Cahill & Forst 2009).

From its inception, XLE was designed to use finite-state transducers for low-level processing steps such as (de)tokenization and morphological analysis and generation (Section 1.2). The interface can readily be used with transducers in Xerox's finite-state transducer format including ones converted from the Foma finite-state transducers, and with relatively little programming effort, other external components can be integrated into an XLE grammar (e.g. see Fang & King 2007 on integrating a non-finite state Chinese word breaker into an XLE Chinese grammar).

In addition to the ϕ-projection from c-structures to f-structures, the XLE parser supports further projections from either of those representations. One of them, the optimality structure, is hard-coded to guide the parsing and generation process on the basis of optimality marks (Section 1.5). The use of optimality marks as a robustness mechanism is one of the many extensions of XLE born out of a joint effort of the group at PARC and its ParGram partners.

Other extensions of the parser and generator are aimed at reducing latency and at ranking the (top n) parses or realizations. For the former, the most notable mechanism is c-structure pruning (Cahill et al. 2008). C-structure pruning relies on corpus data annotated with (partial) constituent bracketing and learns to eliminate highly unlikely c-structures before the computationally expensive resolution of f-annotations. For the latter, a component for training[4] and applying maximum-entropy models based on a large variety of features is provided as part of XLE (Riezler et al. 2002).

Beyond the parser and the generator, XLE also contains a term-rewriting component which was first developed for transfer-based machine translation but has been used for a number of other purposes: identifying and deleting modifiers in f-structures that can be deleted without changing the meaning of the corresponding sentences too much (Riezler et al. 2003); treebank (Rosén 2023 [this volume]) conversion from one dependency-oriented format into another (Forst 2003); further normalization of f-structures and/or construction of semantic representations (Crouch & King 2006, Bobrow et al. 2007); extraction of features for parse ranking (Forst 2007) and realization ranking (Cahill & Forst 2009).

Currently XLE is used by the academic members of the ParGram initiative (Section 3) as well as by individual researchers. It can be used online with LFG implementations for a number of languages via XLE-Web,[5] a web interface for XLE developed at the University of Bergen, and is used as part of the INESS infrastructure developed there (Rosén et al. 2009, 2012). See Rosén 2023 [this volume] for details on using INESS for parsebanking and more generally the uses of LFG parsebanks. XLE is available for non-commercial research purposes.[6] Uses beyond that require a license agreement with PARC and Xerox.

1.1.4 FLE

The Free Linguistic Environment (FLE) (Ćavar et al. 2016) aims to create an LFG-oriented grammar-development and parsing environment with a license less restrictive than XLE’s. It is implemented in C++ and uses the same grammar syntax as XLE, but it is subject to the Apache 2.0 license. In addition to the context-free grammar format of XLE, it supports two probabilistic context-free grammar formats. For tokenization and morphological analysis, FLE provides an interface to Foma transducers.[7] FLE uses open-source components when possible. FLE provides basic parsing functionality but does not contain a generator capable of producing surface strings for input f-structures.[8]

[4]Training data comprises sentences with labeled bracketing, which can be derived from treebanks or created manually (Riezler et al. 2002).

[5]XLE-Web is available at http://clarino.uib.no/iness/xle-web

[6]XLE is available at https://ling.sprachwiss.uni-konstanz.de/pages/xle/redmine.html

[7]Foma supports the import from and the export to XFST formats and XFST supports Foma transducers.

[8]FLE is available at https://gorilla.linguistlist.org/fle/

1.2 Core components

The LFG systems described above allow grammar writers to implement LFG grammars with annotated phrase structure rules and lexical entries similar to those in theoretical LFG. The main difference is that the formatting is specified with easier-to-type variants, e.g. symbols like ↑ and ↓ are replaced with ^ and !.

(1) Example theoretical and implemented annotated c-structure rules:

Theoretical notation:			Implementation (XLE system) notation:	
S ⟶	NP	VP	S -->	NP: (^ SUBJ)=!;
	(↑ SUBJ)=↓	↑=↓		VP: ^=!.

1.2.1 Preprocessing

In order to implement an LFG grammar, it is necessary to preprocess the text that the grammar will parse. Minimally the preprocessing contains a tokenizer which breaks the text into tokens (i.e. words) and canonicalizes the capitalization if necessary (e.g. lowercasing sentence initial capitalized words in English unless they are proper nouns). These canonicalized tokens are then looked up in the lexicon. Implemented lexicons are similar to their theoretical counterparts, comprising the word, its part of speech, and f-structure annotations such as PRED, CASE, and NUMBER. This information is integrated into the grammar via the annotated c-structure rules, as in theoretical LFG. Many implementations integrate a morphological analyzer which associates inflected forms of words with their lemma and morphological information. When using a morphological analyzer, the text is first tokenized and canonicalized for capitalization and then processed by the morphology. The output of the morphology (lemmata and morphological tags) are looked up in the lexicon. This simplifies the lexicon which only has to contain the lemmata and the morphological tags instead of containing all the inflected forms. These morphologies are often finite-state transducers (FSTs; Beesley & Karttunen 2003) which can be used for both parsing and generation.[9] For more details on using FSTs for preprocessing for LFG grammars see Kaplan et al. 2004 and Bögel et al. 2019, for integration of externally developed morphologies and lexicons within LFG grammars see Kaplan & Newman 1997.

A given inflected form can have multiple morphological analyses. Often all the analyses are provided as input to the LFG grammar, and the c-structure rules

[9]Parsing goes from a string (e.g. a natural language sentence) to a c- and f-structure. Generation goes from an f-structure to a natural language string. Most theoretical LFG focuses on parsing, although some accounts, especially OT-LFG ones (see papers in Sells 2001), discuss generation.

Original text:	Dogs barked.		
Tokenization:	Dogs	barked	.
	dogs		
Morphology:	dog +Noun +Pl	bark +Verb +Past	. +Punct
	dog +Verb +3Sg		

Figure 1: Example preprocessing: Tokenization and morphological analysis

and f-structure constraints are used to eliminate analyses which are not feasible in the context of the sentence (e.g. the verbal analysis of *dogs* in figure 1). Preprocessing with a part-of-speech (PoS) tagger marks each word with its part of speech, as in (2). This information can be used to prune the morphological analyses and thus constrain the c-structure built over the sentence. Since PoS taggers are not perfect even for well-edited text, only certain tags are kept, or fall-back techniques are used when no analysis is found. See Kaplan & King 2003 and Dalrymple 2006 for more details on integrating PoS taggers into LFG and other symbolic grammars.

(2) Dogs/Noun barked/Verb and/Conj the/Det cat/Noun left/Verb ./Punct

1.2.2 Projections

Theoretical LFG posits projections beyond the original lexicon, c-structure and f-structure. The exact number and combination of these projections is a subject of lively debate (Belyaev 2023 [this volume]). These include Lexical Mapping Theory (LMT) to map between underlying argument structure and grammatical functions in the lexicon (Findlay et al. 2023 [this volume]), phonological and prosodic projections (Bögel 2023 [this volume]), semantics and semantic structure (Asudeh 2023 [this volume]), and information structure for discourse function information (Zaenen 2023 [this volume]). Most LFG implemented grammars do not include these additional projections because f-structures are sufficient for the applications they target. Even when other projections are included, they are often different from their theoretical counterparts both in their format and in how they are projected or derived. The primary additional component that is included is the semantic component. This component is based on the f-structure and is generally not a projection but instead is a separate post-processing step, although in some stages of its development the Norwegian ParGram grammar NorGram (Dyvik et al. 2016, 2019) included a semantic projection (Halvorsen 1983, Kaplan 1987, Halvorsen & Kaplan 1995) whose representations were in Minimal

Recursion Semantics (Copestake et al. 2005). Semantic components include Glue semantics (Dalrymple et al. 1993, Meßmer & Zymla 2018) and ordered rewriting rules (Crouch & King 2006). The ordered rewriting rules have been extended into abstract knowledge representations (Bobrow et al. 2007). XLE-based implementations have been created for morphological structure (Butt et al. 1996) and prosodic structure and information structure (Butt & King 1998), although none of these are used in large-scale grammars. Instead, they focus on testing theoretical hypotheses and determining the complex interactions among different grammar components (Section 2). The lack of an implementation of LMT has resulted in issues for the parsing of morphologically rich languages like Turkish and Urdu, where interactions between passive and causative constructions cannot be easily captured in LFG implementations (Section 2; Çetinoğlu et al. 2009).

1.2.3 Ambiguity

Implemented grammars often include components to handle ambiguity (see Section 1.5). There are three broad areas around managing ambiguity: computing all the analyses efficiently; representing the ambiguities compactly; resolving the ambiguity so that it does not need to be computed and represented. The first two are discussed in Kaplan & Wedekind 2023 [this volume] and Rosén 2023 [this volume]. Within the grammar writer's control are components including preprocessing by PoS taggers and named entity recognition systems, Optimality-Theory marks to prefer some constructions over others, and stochastic ranking of analyses.

1.2.4 Configuration

The determination of which components (e.g. which tokenizer, morphology, lexicons, and annotated c-structure rules) to use in an implemented grammar need to be specified in a configuration (see Crouch et al. 2011 on how this is done in XLE). These may have default values, e.g. a tokenizer which simply splits sentences at spaces and does not deal with capitalization or punctuation, but large-scale grammars require customized components for the specific language and often the type of text (e.g. newspaper text, tweets). In addition, to allow for rapid extension to specific applications which may have new vocabulary and unusual constructions, these configurations allow the grammar writer to specify lexicons and rules that add to or override those in a standard base grammar (King & Maxwell 2007). For example, to parse English academic biology papers, special lexicons of biological terms as well as special c-structure rules for section titles might be added to a grammar of standard written English.

1.3 Grammar development tools

To aid the grammar writer in managing a large-scale, broad-coverage LFG grammar, specialized variants of standard software development tools are needed. These grammar development tools are part of any LFG platform (Section 1.1). Throughout this chapter we rely on examples from XLE (Crouch et al. 2011), which is the most broadly adopted LFG grammar development framework and is used in the ParGram project (Section 3).

1.3.1 Grammar writer interface

Grammar-development tools for the creation of LFG implementations facilitate the creation of c-structure rules and lexicon entries that are annotated with LFG functional annotations. Some platforms, e.g. Xerox's Grammar Writer's Workbench and XLFG, provide special interfaces for rules and lexicon entries. Others, e.g. XLE, use editors such as Emacs or the Eclipse-based eXLEpse (Rädle et al. 2011). The interfaces provide a way to apply the rules (i.e. the grammar) to a given input string and to output a c-structure and an f-structure graph in human-readable and machine-readable formats. They also generally provide tools to help debug issues such as why a well-formed input sentence does not receive an analysis or why the analysis is incorrect.

1.3.2 Macros and templates

Since grammar engineers want to efficiently encode patterns across lexicon entries and grammar rules, some platforms support additional notations. XLE, for example, supports regular-expression macros that can expand to anything from a piece of f-annotation to an entire rule as well as f-annotation templates, e.g. to allow for like-category coordination over any c-structure category. Using a shared definition of templates across parallel LFG implementations for various languages and domains considerably facilitates the adherence to the agreed-upon f-structure conventions (King et al. 2005). For example, using a template NUMBER wherever number on nouns is assigned ensures that the same attribute (e.g. NUMB) is used and that it only needs to be changed in one place if later another name of the attribute is used (e.g. NUM instead of NUMB). See Section 2 for discussion of the role of macros and templates in theoretical LFG.

1.3.3 Feature table and feature space

In a grammar formalism with untyped attribute-value matrices such as LFG, it is not strictly necessary to declare the valid values for the attributes used in f-structures and potential other levels of representation. However, from an engineering standpoint, it is highly desirable to make sure that only valid values are used; this way, unintended deviations due to typos can be caught easily (Crouch & King 2008). This need to enforce the adherence to a set of conventions is heightened in efforts to develop parallel LFG implementations for various languages such as ParGram (Section 3). XLE therefore supports feature declarations which state all the features, i.e. attributes, and their values that are allowed in the grammar. Multiple feature declarations can be combined to check the grammar code for adherence to them. In ParGram, each grammar combines the common feature declaration with a language-specific one which adds additional language-specific features and declares which subset of values are allowed, e.g. for English the *dual* value of the NUM attribute is removed.

1.3.4 Treebanks as test suites

Treebanks, and more specifically f-structure banks (Rosén 2023 [this volume]), can be used as a form of detailed, LFG-specific test suite for the grammar's coverage. Creating the treebank highlights missing constructions and vocabulary in the grammar. The INESS-based Parsebanker (Rosén 2023 [this volume]) provides infrastructure for rapidly selecting the best parse from an XLE analysis by making use of c- and f-structure discriminants (Rosén et al. 2007). These discriminants are stored as part of the parsebanking to allow for rapid updating as the grammar evolves. The grammar is then enhanced to account for these and the treebank is reparsed with the updated grammar and the new version of the treebank is inspected. This aids both in improving coverage and in ensuring that changes to the grammar do not break constructions that were previously covered. This approach has been used extensively in the development of the Norwegian (Dyvik et al. 2016), Polish (Patejuk & Przepiórkowski 2012), and Wolof (Dione 2014) grammars.

1.3.5 Version control

Version control is used in software development to track changes to the software being developed. As with software development more generally, version control in grammar development allows the grammar writer to compare two versions of a rule, lexical entry, or any other part of the grammar, to revert to a previous version if needed, and to view conflicting changes. Version control systems also record who made a particular change, which makes it easier for multiple people to work on a grammar simultaneously by highlighting recent changes, especially conflicting ones. To our knowledge, eXLEpse (Rädle et al. 2011) is the only LFG-oriented editor that offers support for a variety of version-control systems. Since eXLEpse is based on Eclipse, all version-control plugins for Eclipse can be used. However, although XLE does not provide a version control system, most large scale grammars use a standard software version control system such as SVN or Git. In addition, regression testing by providing sentences and analyses known to be parsable by the grammar help in determining whether new versions of a grammar function properly (Chatzichrisafis et al. 2007, de Paiva & King 2008).

1.3.6 Documentation

As with any software development project, it is important to document what each part of the implemented grammar does. This takes the form of comments in the lexicon and annotated phrase structure rules, including examples of sentences which that part of the grammar can parse. Dipper 2003 designed a self-documenting grammar system whereby the comments are extracted into proper, stand-alone documentation and example test suites of constructions covered by the grammar.

1.4 Modularity and integration of systems

LFG is an inherently modular linguistic theory, with different representations and components for the lexicon, phrase (constituent) structure, functional structure, semantics, etc. This is highlighted in implemented systems which introduce two other types of modularity: modularity for the grammar components, which correlates with the linguistic modularity, and modularity within those components, which enables better grammar engineering practices. LFG implementations are software systems and hence modularity of the different components is important for developing, scaling, maintaining and debugging the system. This section describes how the modularity of the grammar components helps with grammar implementation.

A core tenet of LFG is that different parts of the grammar require different types of representations. This is echoed in the implementations where the different modules can be created by different people and use different types of technology. As with theoretical LFG, the c-structure is a tree and the f-structure an attribute-value matrix, and the two are related via annotated phrase-structure rules. These phrase-structure rules form one module of the grammar. Similarly, lexicons comprise word forms, parts-of-speech, and f-annotations. These form another module. These lexicons can be custom-created for the LFG grammar or converted from other lexical resources (Kaplan & Newman 1997, Sheil & Ørsnes 2006, Przepiórkowski et al. 2014, Patejuk & Przepiórkowski 2014). The morphological component is often implemented as a finite-state transducer (Kaplan et al. 2004, Bögel et al. 2019) but can be of any form.[10] For example, the ParGram Chinese grammar uses a combined tokenizer and part-of-speech tagger that was externally developed for non-LFG purposes (Fang & King 2007). The importance of modularity is highlighted by the treatment of semantics: there have been many implementational approaches to semantic representations based on the LFG f-structure analyses. These include projecting the semantics as an attribute-value matrix (Halvorsen 1983, Halvorsen & Kaplan 1995, Asudeh 2006, Dyvik et al. 2016, 2019), implementing Glue Semantics (Dalrymple et al. 1993, Meßmer & Zymla 2018), and using ordered rewrite rules (Crouch & King 2006). Without a modular system, this exploration of the best way to capture the semantics would be difficult.

There are three additional reasons to maintain modularity in an implemented grammar. The first is that large scale grammars often have multiple grammar writers. By having different files for the lexicon, templates, and annotated phrase structure rules, the efforts can be divided in such a way that changes can be easily merged. To further aid this, the lexicons and phrase-structure rules often comprise multiple files, e.g. the lexicon might be divided into verbs, closed-class items, and all other entries, and the phrase-structure rules might be divided into clausal and nominal. The second reason is that debugging, i.e. the process of finding and fixing errors in the grammar, is simpler in a more modular system. By having different components and different files within those components, the structure of the grammar is easier to see and the individual rules easier to locate. This debugging is further aided by the use of test suites (Chatzichrisafis et al. 2007, de Paiva & King 2008), including ones based on examples in comments in the grammar rules (Dipper 2003). Even with modularity, the inclusion of OT marks (Section 1.5) can make debugging more complex since an analysis may not surface due to competition with another analysis. A third reason is that as described in Section 1.2 and Section 1.3, in addition to a lexicon and annotated phrase structure rules, LFG implementations can have tokenizers, morphologies, templates, feature tables, etc. These are combined via configuration files that encode the different modules of the system and the way they interact.

[10]The non-FST morphologies are referred to as library transducers in XLE.

1.5 Runtime performance

When implemented grammars are used to test linguistic hypotheses and analyses (Section 2), how quickly the grammar provides an analysis for a sentence, i.e. its latency, is generally not important. However, almost all other uses for implemented grammars (Section 4) have latency considerations. LFG implementations have provided a number of techniques to improve latency, sometimes at the cost of accuracy and coverage, e.g. certain analyses may be lost due to early elimination of possible structures (Kaplan et al. 2004). There are two main issues with runtime performance of LFG grammars: ambiguity and latency. These considerations hold for both parsing and generation; we focus on parsing here.[11]

Ambiguity concerns the multiple analyses (i.e. c- and f-structures) that are assigned to a given sentence. The ambiguity problem is accentuated when there is no semantic or pragmatic processing to guide the choice among the different analyses. The ambiguities fall into three broad categories. First, sentences can have multiple analyses, all of which are correct and equally plausible out of context, e.g. in *I saw her duck* either I saw a bird or I saw a person ducking down. Second, sentences can have correct analyses but even out of context some of them are highly improbable, e.g. in *I saw the child with the telescope* there are two plausible readings where *saw* is the past tense of the verb *see* and one implausible one where *saw* is the present tense of the verb meaning to cut with a saw, which is only plausible in a bizarre magic show. Third, ambiguities can arise when the grammar allows ungrammatical analyses, either intentionally as a fall-back mechanism or unintentionally due to an error in the implementation. Copperman & Segond 1996 provide one of the first detailed expositions of ambiguity in LFG grammars, comparing the ambiguity discussed in the theoretical linguistics literature with that in implemented grammars. King et al. 2004 discuss ambiguity in LFG grammar writing in detail, focusing on the XLE-based LFG implementations.

Language contains ambiguities at many levels, from determining word boundaries in tokenization, to morphological analysis, to syntactic attachment ambiguities, to semantic quantifier scope and beyond. This can result in thousands of analyses even for short sentences and long processing times to compute each analysis. There are two main ways to handle this ambiguity efficiently. One is to handle the ambiguity by "packing" (Maxwell & Kaplan 1989, 1993, Shemtov 1997) and operating at each level efficiently over the packed representations. Packing allows operations to apply just once to shared parts of the representation instead

[11]See Kaplan & Wedekind 2023 [this volume] on the inherent formal and computational properties of LFG.

of enumerating all of the possibilities and processing each of them separately. For example, XLE is designed to maintain packed structures from the tokenization and morphology to the syntactic c- and f-structures and then into an ordered rule writing system that can be used to create semantic representations (Crouch & King 2006). The other way to handle ambiguity is to choose the most likely analysis at each level. For example, if there are multiple morphological analyses for a word (e.g. English *leaves*), the system can choose the most likely one given the information it has at that time (e.g. the words adjacent to *leaves* and their potential morphological analyses). This has the downside that the correct analysis may be lost due to removing information early (Dalrymple 2006).

Optimality Theory (OT) (Kuhn 2023 [this volume]) can be used to allow the grammar writer to prefer certain analyses and even to control which grammar rules are active. Frank et al. 1998, 2001 propose an extension of the classical LFG projection architecture to incorporate a constraint ranking mechanism inspired by OT. A new projection, the o-projection, specifies violable constraints, which are used to determine a "winner" among competing, alternative analyses. Many ambiguities can be filtered from the set of possible analyses for a given sentence by using this constraint ranking mechanism in the XLE system. For example, OT marks can be used to prefer verbal analyses over adjectival ones in copular clauses with passives like *They were eaten*. XLE further provides a way to cut down the search space in parsing, allowing for potentially fewer parses to search through. This is done via a special STOPPOINT feature, which is part of the Optimality Theory preference mechanism incorporated into XLE (King et al. 2000). The OT marks can be grouped with certain groups only applying if no parse is found with the original set of OT marks. That is, XLE will process the input in multiple passes, using larger and larger versions of the grammar in subsequent reparsing phases. These groupings are referred to as STOPPOINTS. STOPPOINTS are useful for eliminating ungrammatical analyses when grammatical analyses are present and for speeding up the parser by only using expensive and rare constructions when no other analysis is available. If a solution can be found with the smaller, restricted grammar, XLE will terminate with this solution. Otherwise, a reparsing phase is triggered. This approach can be used to prefer multi-word expressions, for instance so that XLE will only consider analyses that involve the individual components of the multi-word expression if there is no valid analysis involving the multi-word expression. In addition to the OT marks, c-structure pruning (Cahill et al. 2008) and part-of-speech tagging and named entity recognition (Kaplan & King 2003, Dalrymple 2006, Krasnowska-Kieraś & Patejuk 2015) can be used to eliminate unlikely c-structures before unification.

Even with the use of OT marks, a sentence may have many valid parses. However, downstream applications often expect a single analysis, i.e. a single f-structure, as input. To use LFG grammars as input to such applications, statistical methods can be used to choose the most probable analysis (Riezler et al. 2002). These stochastic models are trained on treebanks or dependency banks of known correct analyses. As a variant of this, Dalrymple 2006 and Krasnowska-Kieraś & Patejuk 2015 investigated using a stochastic part-of-speech tagger to trim potential analyses before constructing the c- and f-structure.

2 Implications for theoretical issues

LFG and HPSG (Pollard & Sag 1994 and, for an implementational perspective, Bender & Emerson 2019) are in the privileged position of having not only a community of theoretical linguists but also of grammar engineers, with significant crossover between the theoretical and grammar-engineering communities. There are four areas in which grammar engineering interacts with theoretical linguistics (King 2011, 2016). These include: using grammar engineering to confirm linguistic hypotheses; linguistic issues highlighted by grammar engineering; implementation capabilities guiding theoretical analyses; and insights into architecture issues. The positive feedback loop between theoretical and implementational efforts is a domain in which LFG and HPSG have a distinct advantage compared to many other linguistic theories, given the strong communities and resources available.

2.1 Confirming linguistic hypotheses

Grammar engineering can be used to confirm linguistic hypotheses (Bierwisch 1963, Müller 1999, Butt, Dipper, et al. 1999, Bender 2008, Bender et al. 2011, King 2011, Fokkens 2014, King 2016, Müller 2015). Encoding the hypothesis in an implemented grammar not only highlights details of the analysis that might be missed in a pencil-and-paper version but can also bring to light interesting interactions with other linguistic phenomena, especially when the hypothesized analysis is encoded in a broad-coverage grammar. Two examples of this type include the analysis of determiner agreement systems and the prosody-syntax interaction.

King & Dalrymple 2004 provide an LFG analysis of determiner agreement and noun conjunction, looking particularly at indeterminacy of agreement features. In order to test the proposed system, they implemented a toy grammar with lexical entries of each type and enough syntactic structure to encompass determiner, adjective, and verb agreement with conjoined and non-conjoined nouns. As a result, the authors were able to confirm that their analysis was formally sound and accounted for the known data. This toy grammar was relatively easy to implement in XLE because all of the necessary components, e.g. distributive features, were already available.

Implementing proposals for the prosody-syntax interaction in LFG is more challenging because not all of the mechanisms that have been proposed in the literature are available in systems like XLE. Butt & King 1998 used an existing, non-LFG analysis of Bengali clitics and implemented it in order to test whether p(rosodic)-structure could be used to capture the generalizations proposed in the theoretical analysis, focusing on where mismatches between prosodic and syntactic structure occur. A much different interface approach was pursued in Bögel et al. 2009, which built upon the finite-state transducers used for tokenization and morphological analysis within the grammars (Section 1.2). Finally, a large-scale implementation of certain phonology-syntax interactions was completed for Welsh (Mittendorf & Sadler 2006).

2.2 Implementational devices

Writing large-scale grammars highlights the interaction of different parts of the grammar and the need to be able to formally state certain types of generalizations. These needs have led to the creation of formal devices, some of which have become part of theoretical LFG analyses while others remain implementational devices. Implementation capabilities that guided theoretical analysis include the use of complex categories for auxiliary analysis in English and German, the analysis of Welsh phonology-syntax interactions through the interaction of morphological analysis via finite-state transducers and the LFG c-structure, and the introduction of templates and macros.

Complex categories (Crouch et al. 2011) are a formal c-structure device. They allow for generalizations over c-structure categories by having the category be composed of a fixed component and a variable, where the variable can pass its value to other complex categories on the right-hand side of the rule. In this way, they allow the grammar writer to capture generalizations through notation. This notation is then automatically compiled into standard c-structure rules. Complex categories are used to constrain the order and form of auxiliaries and main verbs in English (e.g. *They will have been promoted.*) by having each auxiliary state its meaning and its form (e.g. *have* is an AUX[perf,base] with perfective meaning and base form while *been* is an AUX[pass,perf] with passive meaning and pefective form) and the VP rules themselves are complex categories that reflect their head and based on that put requirements on their complement.

Welsh consonant mutations are a phenomenon whereby the initial consonant of certain words changes based on its phonological and syntactic environment (Graver 2023 [this volume]). To capture the joint requirements on the morphophonology and the syntax which trigger mutations, Mittendorf & Sadler 2006 used the finite-state morphology capabilities integrated in XLE to control where Welsh consonant mutations occur by encoding the boundary conditions in the morphological tag sequences. The modular nature of LFG combined with the implementational device of finite-state morphology provided a clean solution to the different types of triggers for the mutations.

A long standing debate in the linguistic literature, especially for constraint-based formalisms like HPSG and LFG, is whether a comprehensive and efficient grammatical theory should include a type hierarchy and what role it should play. Historically HPSG has had types as foundational to the theory while LFG has not. However, in grammar engineering, it is important to be able to efficiently capture generalizations as well as exceptions to those generalizations. The introduction of templates into the formal devices available to LFG allows for generalizations and inheritance via notation, without introducing a full type hierarchy into the formalism (Dalrymple, Kaplan & King 2004, Crouch & King 2008) and as a result, the concept of templates has become part of theoretical LFG analyses. Similar to complex categories, templates and macros allow the grammar writer to capture generalizations through notation, which is then automatically compiled into standard LFG c- and f-structure rules.

Two more minor formal devices which are gaining traction in theoretical analyses are instantiation and local variables (a third is the restriction operator discussed in the next section). Since the beginning, predicates (PRED) in LFG have not been unifiable with one another due to their unique lexical index (Kaplan & Bresnan 1982). Certain non-PRED features also need to be non-unifiable (Dalrymple 2001). This can be captured by instantiation, represented by having the value of the feature be followed by an underscore. For example, instantiating the form values of English particles blocks their occurring multiple times in a sentence (e.g. **they threw out the garbage out*) (see Figure 2 for an English example and Forst et al. 2010). Finally, local variables anchor a functional uncertainty equation to a particular f-structure and then refer to that f-structure in other annotations (Dalrymple 2001, Crouch et al. 2011). This is needed when making a set of statements about a particular element of a set or a particular type of governing element. For example Szűcs 2019 uses local variables to state constraints on topic left dislocation constructions in Hungarian.

2.3 Architectural issues

Implementing a wide variety of phenomena, as is necessary for broad-coverage grammars, brings to light architectural issues with the theory. Çetinoğlu et al. 2009 and Bögel et al. 2019 describe issues with the interaction of the passive and causative in Turkish and Urdu. These issues are the result of how lexical rules in LFG interact with complex predicate formation, where the passive is traditionally analyzed as involving a lexical rule while the causative is often analyzed as a complex predicate. The Urdu and Turkish grammars use the restriction operator (Kaplan & Wedekind 1993) in the annotated c-structure rules to model complex predication, including causatives. The restriction operator allows for features of f-structures to be restricted out, i.e. to cause the grammar to function as if these features did not exist. This allows complex predicate-argument structures to be built dynamically (Butt et al. 2003, 2010). In contrast, the passive is handled by lexical rules which apply to the predication frames in the lexicon. This predicts that passivization applies before causativization and that it is not possible to passivize a causative by demoting or suppressing the subject of the causative. However, this is the reverse of the Urdu and Turkish facts. To solve this problem in the ParGram grammars of Urdu and Turkish, both the causative and the passive are handled via restriction in the annotated phrase structure rules. In the theoretical literature, this issue had not been highlighted because for Turkish and Urdu style morphosyntax, the causative was handled in argument-structure, but the interaction between causativization and passives at the morphology-syntax interface highlighted that traditional lexical rules do not allow for the right order of application when causativization is morphological but passivization is part of the syntax.

To conclude this section, the interaction of grammar engineering and theoretical linguistics helps to confirm linguistic hypotheses, to highlight complex linguistic issues, to posit new formal capabilities, and shed light on architecture issues. The positive feedback loop between theoretical and implementational efforts is a domain in which LFG and HPSG have a distinct advantage.

3 Grammar resources: ParGram

The systems described above are used to create small- and large-scale LFG grammars. These can be used as input to applications (Section 4) or to explore theoretical hypotheses (Section 2). The Parallel Grammar (ParGram) project is a consortium of LFG researchers implementing grammars for a typologically varied set of languages in a parallel fashion (Butt, King, et al. 1999, Butt et al. 2002) using the

XLE LFG parser, generator, and grammar development platform. The parallels are most notable in the f-structure space, where common features and analyses are used wherever possible, but differ when required by the syntax of the languages. This parallelism is enabled by LFG theory, by grammar engineering components such as feature declarations, and by semi-annual meetings between the grammar writers.[12]

ParGram began with three languages: English (Riezler et al. 2002), French (Frank 1996), and German (Dipper 2003, Rohrer & Forst 2006). They developed aligned f-structure analyses for a tractor manual which existed as an aligned corpus in all three languages. Even with three closely related languages, it was clear that full f-structure alignment was not possible (Butt, Dipper, et al. 1999) due to fundamental syntactic differences in the languages. Later, the Fuji Xerox Corporate Research Group and the University of Bergen joined the initiative with a Japanese (Masuichi et al. 2003) and a Norwegian grammar (Dyvik et al. 2016, 2019) respectively. Other longer-term academic efforts participating in ParGram concern the development of Urdu (Butt & King 2002, 2007) and Polish (Patejuk & Przepiórkowski 2012) LFG implementations. Finally, further ParGram efforts have given rise to computational LFGs for Arabic (Attia 2006, 2012), Chinese (Fang & King 2007), Danish (Ørsnes 2006), Georgian (Meurer 2009), Hungarian (Laczkó & Rákosi 2008–2019), Indonesian (Arka et al. 2009, Arka 2012), Korean (Kim et al. 2003), Malagasy (Dalrymple et al. 2006), Tamil (Sarveswaran & Butt 2019), Tigrinya (Kifle 2011), Turkish (Çetinoğlu & Oflazer 2018), Welsh (Mittendorf & Sadler 2006), and Wolof (Dione 2014).

The project resulted in the creation of LFG grammars in these multiple languages and hence a greater understanding of the parallelism (or lack thereof) for the LFG analyses of particular constructions. Major issues in LFG analysis and architecture highlighted by the ParGram project included: Copular constructions and in particular whether there is a copular *be* predicate and whether the predicated argument has a subject (XCOMP-like) or not (PREDLINK) (Dalrymple, Dyvik, et al. 2004, Attia 2008); how to handle argument-changing relations such as the passive, causative, benefactives, complex predicates, and interactions thereof, including morphological and syntactic interactions (Bögel et al. 2019; see Section 2); whether auxiliaries have predicates or just supply tense and aspect features to the f-structure (Butt et al. 1996, Dyvik 1999); the interaction of tokenization and morphology with the c- and f-structures, especially around features like Welsh mutations (Mittendorf & Sadler 2006) and Urdu complex predicates (Bögel et al.

[12]A similar approach was subsequently adopted by the HPSG DELPH-IN consortium (Bender et al. 2002).

2019). In addition, the ParGram project resulted in improvements to the grammar development platform (Section 1.2, Section 1.3 and Section 1.5) and in best practices for distributed parallel grammar development.

In addition to the traditional LFG-style ParGram grammars which use annotated phrase structure rules to create the c- and f-structure representations, the ParGram project also includes several automatically induced grammars that create ParGram compatible f-structures, i.e. f-structures using the same feature space as the grammars described above, but which are learned from tree and f-structure banks (Cahill et al. 2002). These grammars are robust in that they produce f-structures for nearly any sentence, at the cost of producing structures which sometimes violate core LFG principles such as completeness and coherence. See Section 4 for applications which require such robustness.

An influential initiative that resembles ParGram is the Universal Dependencies (UD) initiative (McDonald et al. 2013; see also Haug 2023 [this volume]). Like ParGram, it aims at parallel representations across languages, and UD follows LFG concerning many of the distinctions made at the level of syntactic dependencies and grammatical functions respectively (de Marneffe et al. 2014). This being said, surface-oriented dependency structures as used in UD cannot be as parallel as the more abstract f-structures of ParGram. Korsak 2018 and Przepiórkowski & Patejuk 2020 discuss the similarities between LFG and UD and investigate mapping between LFG f-structures and UD. Another noteworthy difference between ParGram and UD is that ParGram has been developing reversible XLE grammars whereas UD focuses solely on parsing.

4 Applications

Some applications integrating natural language processing only require parsing. For these applications, parsing should be robust to typos and grammatical errors, unusual constructions, unknown words, etc. In addition, minor issues in parsing may be unimportant for these applications because systematic errors can be compensated for within the system. Semantic search is an application that requires only parsing, needs to be robust, and can tolerate certain parsing errors.

Other applications, e.g. sentence condensation, transfer-based machine translation (MT) and conversational agents, require both parsing and generation. Applications using generation generally require highly grammatical output since users are sensitive to malformed natural language such as incorrect subject-verb agreement. Since corpus-induced grammars do not lend themselves to refinement in order to control generation, hand-crafted grammar implementations such as LFG grammars are still the means of choice for the generation of high-quality text.

Finally, there are applications that require grammaticality judgments. This is the case of grammar checkers, both general-purpose ones and grammar checkers for computer-assisted language learning (CALL). Parsers trained on general-purpose treebanks cannot be used for this purpose, so these applications are another natural fit for hand-crafted grammar implementations. In our opinion, LFG suits this purpose particularly well because its terminology is relatively close to that used in language instruction.

4.1 Applications requiring deep features and robustness

For applications that require mainly natural language understanding, parsing needs to be robust to unexpected words and constructions. To provide the robustness necessary for these applications, domain-specific grammars can be created based on a general large-scale grammar (Kim et al. 2003, King & Maxwell 2007). However, this is often not enough to cover all use cases. LFG grammars can use morphological guessers to cover unknown vocabulary (Dost & King 2009, Bögel et al. 2019), can parse fragments of the structure, e.g. provide f-structures for all the noun phrases even if they cannot be formed into a sentence (Riezler et al. 2003), and can include fall-back rules (mal-rules Schneider & McCoy 1998, Reuer 2003, Khader 2003, Fortmann & Forst 2004, Bender et al. 2004) explicitly accounting for certain types of ungrammaticality, e.g. incorrect subject-verb agreement.

Semantic search is one application which benefits from the deep LFG representations. As a search application, the goal is to find documents which are relevant to the query and, ideally, to highlight the passage in the document most relevant to the query. Semantic search moves beyond keyword matching to match the relationships between entities in the query. It can include queries that are full interrogatives as well as ones that are phrases. The ParGram XLE English grammar was used in the Powerset Inc. semantic search engine for searching Wikipedia articles. By using LFG representations for the query and the documents it can differentiate between *who acquired PeopleSoft* and *who did PeopleSoft acquire*, where PeopleSoft is the object in the first question and the subject in the second. By using a fragment grammar as a backup, longer sentences could be partially parsed, e.g. the first conjunct of a coordinated sentence could be parsed even if the second failed. This combined with the redundancy across the articles made using an LFG grammar feasible for moving beyond keyword search. The f-structures were mapped to abstract knowledge representations which went beyond grammatical functions to semantic rules, e.g. mapping *Oracle acquired PeopleSoft* and

PeopleSoft was acquired by Oracle and even *Oracle's acquisition of PeopleSoft* to the same abstract representation.

A more complex application than semantic search is question answering. Unlike search, question answering uses a document collection to find the answer to the query, which is generally in the form of a natural-language question, and present it to the user. The PARC Bridge system (Bobrow et al. 2007) used the XLE ParGram English grammar as its base and mapped the query and documents to an abstract knowledge representation using ordered rewrite rules, deep lexical resources such as WordNet (Fellbaum 1998) and VerbNet (Kipper et al. 2000, Levin 1993), and knowledge resources such as Cyc (Lenat 1995). The queries and documents were then matched against one another with a graph-based algorithm. An interesting extension of this was to perform entailment and contradiction detection (ECD) (Bobrow et al. 2007) with a graph-based module that determined whether one sentence entailed or contradicted (or neither) the other. ECD depended on understanding the roles between the entities as determined by the LFG grammar as well as detailed lexical knowledge.

Burton 2006 describes a tutorial system which uses the XLE English grammar for its language-understanding component. The tutorial system is provided by Acuitus and teaches network administration. The coursework includes a set of troubleshooting exercises where students find and fix problems. During these exercises the computer helps the students when they ask for help or based on their actions. The system asks the student a mix of multiple-choice, short-answer, and natural-language questions. The idea behind using natural-language interactions is to encourage students to think beyond what multiple-choice questions provide and to allow more complex questions and answers. The system converts the f-structures from the student input to semantic interpretations via the transfer rule system (Crouch 2006). Both the syntactic parsing and the semantics are adapted to the domain to provide more accurate and robust results.

Historically, hand-crafted LFG implementations have had a hard time competing with machine-learned constituency or dependency parsers in terms of robustness, i.e. providing a parse for all input, and speed for purely understanding-oriented applications, even though they are often superior in terms of systematicity and detail of analysis and despite the fact that machine-learned parsers often produce illogical parses for input where LFG grammars would fail to produce a parse. Because of this speed and perceived robustness, machine-learning-based dependency parsers have become increasingly popular, as is evident from the shared tasks of the Conference on Computational Natural Language Learning (CoNLL) series. Interesting though, the CoNLL tasks now often integrate UD representations (McDonald et al. 2013), which can be seen as less fine-grained f-structures (see Section 3 for more details on UD). The combination of hand-crafted grammars, fall-back techniques, and statistical parser selection as described in this chapter allow LFG and other rule-based grammars to be used in applications requiring robustness (see also Ivanova et al. 2016).

4.2 Applications requiring grammaticality

Certain applications not only aim to map text to representations more amenable to the computation of meaning, but they also take abstract meaning representations, including f-structures, as input and map them to text. Among such applications are sentence condensation and transfer-based machine translation, both applications for which LFG implementations have been used because f-structures are abstract enough to facilitate transformations like the removal of certain adjuncts or the transfer from a source to a target language. Furthermore, since corpus-induced grammars do not lend themselves to refinement in order to control generation, hand-crafted grammar implementations are still the means of choice for the generation of high-quality text.

Sentence condensation is a form of summarization (Knight & Marcu 2000, Jing 2000). It takes a long sentence and produces a shorter sentence which preserves the core meaning of the original sentence. This requires the ability to identify the core part of the original sentence and to generate a grammatical shorter sentence. Riezler et al. 2003 and Crouch et al. 2004 used the ParGram XLE grammar to create a sentence condensation system for English. The LFG f-structure was used to identify the core meaning, e.g. by removing adjuncts other than negation. A new f-structure was created which contained only this core meaning. This new f-structure was then run through the grammar in the generation direction to generate the shorter, condensed sentence. This sentence was guaranteed to be grammatical since it met the well-formedness conditions of the grammar. Since multiple strings (e.g. sentences) can map to the same f-structure, more than one condensed sentence can often be generated from a single f-structure. This can be partially controlled by Optimality-Theory marks in the grammar in XLE (Frank et al. 1998). The choice between the remaining sentences can be done with a language model (Riezler et al. 2003). A related application is note taking where longer texts are condensed into legible notes (Kaplan et al. 2005).

Machine translation (MT) involves automatically translating a text from one language (the source) to another (the target). The resulting translation has to preserve the meaning and to be grammatical. LFG f-structures have been used for MT (Oepen et al. 2004, Riezler & Maxwell 2006, Avramidis & Kuhn 2009, Graham et al. 2009, Graham 2012, Graham & van Genabith 2012, Homola & Coler

2012). The idea is that the f-structure encodes the meaning of the sentence more abstractly than the surface form of the text and so can be used as the level for translation. That is, f-structure enables translation by transfer across structures and not just an interlingua across words (Kaplan et al. 1989). In theory, simply substituting the PRED values in the f-structure could produce an f-structure in the target language and the LFG grammar can then be used to generate the translation. In practice, f-structures still encode enough language-specific syntactic information that additional transfer rules need to be applied before the generation step. For example, one language may use indefinite singular determiners (e.g. English *a*) while the other may not, in which case the determiner would have to be deleted (in the source language) or inserted (in the target language). The LOGON MT project (Oepen et al. 2004) provides an interesting approach with parsing via the LFG Norwegian NorGram grammar, transfer to semantic MRS (Copestake et al. 2005) and generation via an HPSG English grammar. Although LFG-based MT systems can be brittle since there has to be a successful parse, transfer, and generation, when a translation is produced it is generally of high quality both in terms of preserving the meaning and of being grammatically well-formed.

Consider the English and German sentences in (3) and (4), for which the corresponding f-structures are displayed in Figure 2.

(3) Across the city, monuments to prosperity have sprung up.

(4) In der ganzen Stadt sind Denkmäler des Wohlstands entstanden.
in the whole city be monuments of.the prosperity up.spring
Across the city, monuments to prosperity have sprung up.

Apart from the fact that the German analysis of adjunct NPs in the genitive is not parallel to other ParGram implementations and that the German finite-state morphology decomposes the word *Wohlstand*, which gives rise to a MOD dependency under the SUBJ ADJ-GEN, the f-structures are surprisingly parallel. (At first sight, this is obscured by the fact that in the German f-structure, the sub-f-structures under TOPIC and in the ADJUNCT set are the same.) Even though the English sentence is headed by a particle verb while the German one is not, there is a single PRED value for the head verb on either side; even though the subject of the English sentence precedes the verb while the one of the German sentence follows the verb, both appear in the respective f-structure under SUBJ; even though the auxiliary in the English sentence is *have* while the German verb *entstehen* requires the auxiliary *sein* ('to be') for perfect tenses, the auxiliaries contribute the same value for TNS-ASP PERF. As a result, the transfer component can concentrate on word-to-word translation equivalencies while letting the language-specific grammars take care of well-formedness conditions independent of the language pair under consideration. An example of a non-trivial translation equivalency is the one between *across the city* and *in der ganzen Stadt* (literally 'in the entire city'), as the English phrase might also correspond to *durch die Stadt* (literally 'through the city') in other contexts (especially in combination with motion verbs).

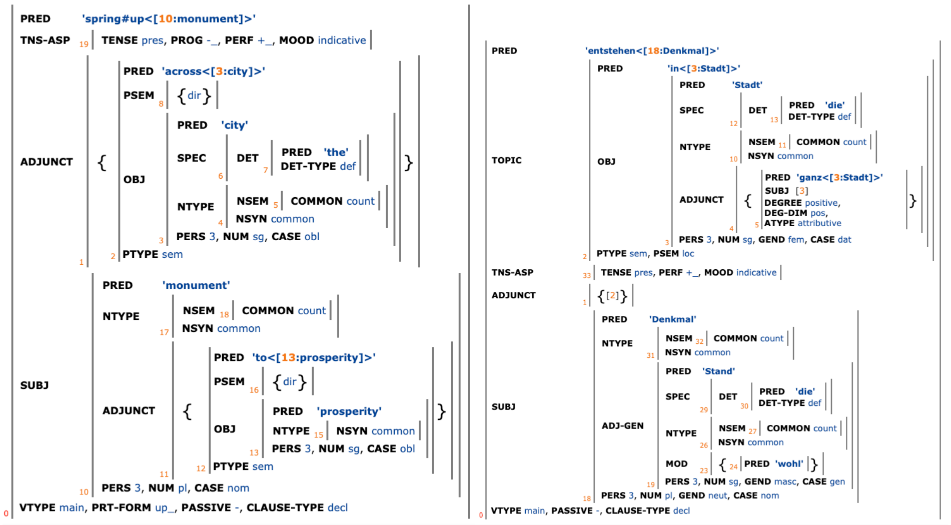
PRED 'spring#up<[10:monument]>'
TNS-ASP TENSE pres, PROG -_, PERF +_, MOOD indicative
ADJUNCT
PRED 'across<[3:city]>'
PSEM {dir}
PRED 'city'
SPEC DET PRED 'the' DET-TYPE def
OBJ
NTYPE NSEM COMMON count NSYN common
PERS 3, NUM sg, CASE obl
PTYPE sem
SUBJ
PRED 'monument'
NTYPE NSEM COMMON count NSYN common
ADJUNCT
PRED 'to<[13:prosperity]>'
PSEM {dir}
OBJ PRED 'prosperity' NTYPE NSYN common PERS 3, NUM sg, CASE obl
PTYPE sem
PERS 3, NUM pl, CASE nom
VTYPE main, PRT-FORM up_, PASSIVE -, CLAUSE-TYPE decl
PRED 'entstehen<[18:Denkmal]>'
TOPIC
PRED 'in<[3:Stadt]>'
OBJ
PRED 'Stadt'
SPEC DET PRED 'die' DET-TYPE def
NTYPE NSEM COMMON count NSYN common
ADJUNCT PRED 'ganz<[3:Stadt]>' SUBJ [3] DEGREE positive, DEG-DIM pos, ATYPE attributive
PERS 3, NUM sg, GEND fem, CASE dat
PTYPE sem, PSEM loc
TNS-ASP TENSE pres, PERF +_, MOOD indicative
ADJUNCT {[2]}
SUBJ
PRED 'Denkmal'
NTYPE NSEM COMMON count NSYN common
ADJ-GEN
PRED 'Stand'
SPEC DET PRED 'die' DET-TYPE def
NTYPE NSEM COMMON count NSYN common
MOD {PRED 'wohl'}
PERS 3, NUM sg, GEND masc, CASE gen
PERS 3, NUM pl, GEND neut, CASE nom
VTYPE main, PASSIVE -, CLAUSE-TYPE decl

Figure 2: F-structures for English and German translation equivalents

Certain other applications do not require semantic representations or grammatical text output but do require the system to have a notion of grammaticality as their purpose is to highlight ungrammatical (or otherwise undesired) passages in text. Such systems can be directed to a general public of people producing texts or explicitly target second-language learners, sometimes even second-language learners with a specific first-language background. The latter application, in the context of Intelligent Computer-Assisted Language Learning (ICALL), has used LFG implementations, typically augmented with mal-rules (Rypa & Feuerman 1995, Reuer 2003, Khader 2003, Fortmann & Forst 2004). Mal-rules are rules or rule extensions that cover ungrammatical constructions typically produced by second-language learners, e.g. NPs where determiners or adjectives do not agree with the head noun, NPs with countable head nouns in the singular that are not preceded by a determiner, or sentences with an ungrammatical order of constituents or a violation of subject-verb agreement. As typical mistakes made by second-language learners depend significantly on their native language as well as on other languages they know, mal-rules can be optimized with respect to their coverage more easily when the linguistic background of the audience is known. A machine-learning-based approach to ICALL exploiting features provided by the English ParGram LFG implementation is described by Berend et al. 2013.

A final application we discuss is natural language understanding (NLU) components used in car computers or in personal assistants on mobile devices. Those components often combine grammar-based analysis and deep-learning-based neural networks or statistical models learned from annotated data. Moreover, machine-learning-based NLU models depend on large amounts of training data from the relevant domain. Since such data is hard to collect and costly to annotate, much of it is generated by means of grammars. For the most part, the grammars used to this end are simple, largely context-free grammars. However, as the semantic representations used for NLU become increasingly sophisticated, the use of more powerful grammar formalisms such as LFG can be used for the generation of high-quality grammatical training data.

5 Conclusion

This chapter provided an overview of computational implementations of LFG. LFG was designed from the outset to be computationally tractable and has a strong history of broad-coverage implementations for multiple languages, primarily through the ParGram project which is built on the XLE grammar development platform. As with theoretical LFG, implemented grammars primarily focus on c-structure and f-structure, but extensive work has been done on using the resulting f-structures as input to semantics and abstract knowledge representation, and some work has focused on the integration of morphological and phonological information as well as argument structure. The ParGram project is based on the theoretical LFG hypothesis that languages are more similar at f-structure, which encodes grammatical functions, than at c-structure. This f-structure similarity can then be exploited in applications such as machine translation. Other applications which take advantage of the more abstract f-structures and the ability of LFG grammars to parse and generate as well as to detect (un)grammaticality include computer-assisted language learning, question answering, and sentence condensation. From a theoretical linguistic perspective, implemented grammars allow the linguist to test analyses and to see interactions between different parts of the grammar.

6 Acknowledgements

We would like to thank Emily Bender, Gerlof Bouma, Ron Kaplan, Agnieszka Patejuk, Victoria Rosén, Annie Zaenen, and two anonymous reviewers for detailed comments on this chapter. All remaining errors are our own.

References

Arka, I Wayan. 2012. Developing a deep grammar of Indonesian within the ParGram framework: Theoretical and implementational challenges. In *Proceedings of the 26th Pacific Asia Conference on Language, Information and Computation*, 19–38.

Arka, I Wayan, Avery D. Andrews, Mary Dalrymple, Meladel Mistica & Jane Simpson. 2009. A linguistic and computational morphosyntactic analysis for the applicative *-i* in Indonesian. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '09 conference*, 85–105. Stanford: CSLI Publications. http://csli-publications.stanford.edu/LFG/14/papers/lfg09arkaetal.pdf.

Asudeh, Ash. 2006. Direct compositionality and the architecture of LFG. In Miriam Butt, Mary Dalrymple & Tracy Holloway King (eds.), *Intelligent linguistic architectures: Variations on themes by Ronald M. Kaplan*, 363–387. Stanford: CSLI Publications.

Asudeh, Ash. 2023. Glue semantics. In Mary Dalrymple (ed.), *Handbook of Lexical Functional Grammar*, 651–697. Berlin: Language Science Press. DOI: 10.5281/zenodo.10185964.

Attia, Mohammed. 2006. Accommodating multiword expressions in an Arabic LFG grammar. In *Advances in Natural Language Processing (FinTAL 2006)* (Lecture Notes in Computer Science 4139), 87–98. DOI: 10.1007/11816508_11.

Attia, Mohammed. 2008. A unified analysis of copula constructions in LFG. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '08 conference*, 89–108. Stanford: CSLI Publications.

Attia, Mohammed. 2012. *Ambiguity in Arabic computational morphology and syntax: A study within the Lexical Functional Grammar framework*. Saarbrücken: LAP LAMBERT Academic Publishing.

Avramidis, Eleftherios & Jonas Kuhn. 2009. Exploiting XLE's finite state interface in LFG-based statistical machine translation. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '09 conference*, 127–145. Stanford: CSLI Publications.

Beesley, Kenneth R. & Lauri Karttunen. 2003. *Finite state morphology*. Stanford: CSLI Publications.

Belyaev, Oleg. 2023. Introduction to LFG. In Mary Dalrymple (ed.), *Handbook of Lexical Functional Grammar*, 3–22. Berlin: Language Science Press. DOI: 10.5281/zenodo.10185934.

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

Bender, Emily M. & Guy Emerson. 2019. Computational linguistics and grammar engineering. In Stefan Müller, Anne Abeillé, Robert D. Borsley & Jean-Pierre Koenig (eds.), *Head-Driven Phrase Structure Grammar: The handbook* (Empirically Oriented Theoretical Morphology and Syntax), 1105–1153. Berlin: Language Science Press. DOI: 10.5281/zenodo.5599868.

Bender, Emily M., Dan Flickinger & Stephan Oepen. 2002. The Grammar Matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In John Carroll, Nelleke Oostdijk & Richard Sutcliffe (eds.), *COLING-GEE '02: Proceedings of the 2002 workshop on Grammar Engineering and Evaluation*, 8–14. Taipei: Association for Computational Linguistics. DOI: 10.3115/1118783.1118785.

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

Bender, Emily M., Dan Flickinger, Stephan Oepen, Annemarie Walsh & Tim Baldwin. 2004. Arboretum: Using a precision grammar for grammar checking in CALL. In *Proceedings of the InSTIL/ICALL Symposium: NLP and speech technologies in advanced language learning systems*. Venice.

Berend, Gabor, Veronika Vincze, Sina Zarrieß & Richárd Farkas. 2013. LFG-based features for noun number and article grammatical errors. In *Proceedings of the 17th Conference on Computational Natural Language Learning: Shared task*, 62–67.

Bierwisch, Manfred. 1963. *Grammatik des deutschen Verbs* (Studia Grammatica II). Berlin: Akademie Verlag.

Bobrow, Daniel G., Bob Cheslow, Cleo Condoravdi, Lauri Karttunen, Tracy Holloway King, Rowan Nairn, Valeria de Paiva, Charlotte Price & Annie Zaenen. 2007. PARC's bridge and question answering system. In Tracy Holloway King & Emily M. Bender (eds.), *Proceedings of the GEAF07 workshop*. Stanford: CSLI Publications. http://csli-publications.stanford.edu/GEAF/2007/geaf07-toc.html.

Bögel, Tina. 2023. Prosody and its interfaces. In Mary Dalrymple (ed.), *Handbook of Lexical Functional Grammar*, 779–821. Berlin: Language Science Press. DOI: 10.5281/zenodo.10185970.

Bögel, Tina, Miriam Butt, Ronald M. Kaplan, Tracy Holloway King & John T. III Maxwell. 2009. Prosodic phonology in LFG: A new proposal. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '09 conference*, 146–166. Stanford: CSLI Publications.

Bögel, Tina, Miriam Butt & Tracy Holloway King. 2019. Urdu morphology and beyond: Why grammars should not live without finite-state methods. In Cleo Condoravdi & Tracy Holloway King (eds.), *Tokens of meaning: Papers in honor of Lauri Karttunen*, 417–438. Stanford: CSLI Publications.

Boullier, Pierre & Benoît Sagot. 2005. Efficient and robust LFG parsing: SxLFG. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT 2005)* (Parsing '05), 1–10. Stroudsburg, PA: Association for Computational Linguistics. DOI: 10.3115/1654494.1654495.

Burton, Richard R. 2006. Using XLE in an intelligent tutoring system. In Miriam Butt, Mary Dalrymple & Tracy Holloway King (eds.), *Intelligent linguistic architectures: Variations on themes by Ronald M. Kaplan*, 75–90. Stanford: CSLI Publications.

Butt, Miriam, Stefanie Dipper, Anette Frank & Tracy Holloway King. 1999. Writing large-scale parallel grammars for English, French and German. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '99 conference*. Stanford: CSLI Publications.

Butt, Miriam, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi & Christian Rohrer. 2002. The Parallel Grammar Project. In John Carroll, Nelleke Oostdijk & Richard Sutcliffe (eds.), *COLING-GEE '02: Proceedings of the 2002 workshop on Grammar Engineering and Evaluation*, 1–7. Taipei: Association for Computational Linguistics. DOI: 10.3115/1118783.1118786.

Butt, Miriam & Tracy Holloway King. 1998. Interfacing phonology with LFG. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '98 conference*. Stanford: CSLI Publications.

Butt, Miriam & Tracy Holloway King. 2002. Urdu and the Parallel Grammar project. In N. Calzolari, K.-S. Choi, A. Kawtrakul, A. Lenci & T. Takenobu (eds.), *Proceedings of the 3rd workshop on Asian Language Resources and International Standardization, 19th International Conference on Computational Linguistics (COLING '02)*, 39–45. DOI: 10.3115/1118759.1118762.

Butt, Miriam & Tracy Holloway King. 2007. Urdu in a parallel grammar development environment. *Language Resources and Evaluation* 41(2). Special Issue on

Asian Language Processing: State of the Art Resources and Processing, 191–207. DOI: 10.1007/s10579-007-9042-8.

Butt, Miriam, Tracy Holloway King & John T. III Maxwell. 2003. Complex predication via restriction. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '03 conference*, 92–104. Stanford: CSLI Publications. http://csli-publications.stanford.edu/LFG/8/pdfs/lfg03buttetal.pdf.

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

Butt, Miriam, Tracy Holloway King & Gillian Ramchand. 2010. Complex predication: Who made the child pinch the elephant? In Linda Ann Uyechi & Lian-Hee Wee (eds.), *Reality exploration and discovery: Pattern interaction in language and life*, 231–256. Stanford: CSLI Publications.

Butt, Miriam, María-Eugenia Niño & Frederique Segond. 1996. Multilingual processing of auxiliaries in LFG. In D. Gibbon (ed.), *Natural language processing and speech technology: Results of the 3rd KONVENS conference*, 111–122. Berlin: Mouton de Gruyter.

Cahill, Aoife & Martin Forst. 2009. Human evaluation of a German surface realisation ranker. In *Proceedings of the 12th conference of the European chapter of the ACL (EACL 2009)*, 112–120. Association for Computational Linguistics. DOI: 10.3115/1609067.1609079.

Cahill, Aoife, John T. III Maxwell, Paul Meurer, Christian Rohrer & Victoria Rosén. 2008. Speeding up LFG parsing using c-structure pruning. In *COLING 2008: Proceedings of the workshop on Grammar Engineering Across Frameworks*, 33–40. Manchester. DOI: 10.3115/1611546.1611551.

Cahill, Aoife, Mairéad McCarthy, Josef van Genabith & Andy Way. 2002. Parsing with PCFGs and automatic f-structure annotation. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '02 conference*, 76–95. Stanford: CSLI Publications.

Ćavar, Damir, Lwin Moe, Hai Hu & Kenneth Steimel. 2016. Preliminary results from the Free Linguistic Environment project. In Doug Arnold, Miriam Butt, Berthold Crysmann, Tracy Holloway King & Stefan Müller (eds.), *Proceedings of the joint 2016 conference on Head-Driven Phrase Structure Grammar and Lexical Functional Grammar*, 161–181. Stanford: CSLI Publications.

Çetinoğlu, Özlem, Miriam Butt & Kemal Oflazer. 2009. Mono/bi-clausality of Turkish causatives. In Sıla Ay, Özgur Aydın, İclâl Ergenç, Seda Gökmen, Selçuk İşsever & Dilek Peçenek (eds.), *Essays on Turkish linguistics: Proceedings of the 14th International Conference on Turkish Linguistics*. Wiesbaden: Harrassowitz Verlag.

Çetinoğlu, Özlem & Kemal Oflazer. 2018. Deep parsing of Turkish with Lexical-Functional Grammar. In Kemal Oflazer & Murat Saraçlar (eds.), *Turkish natural language processing* (Theory and Applications of Natural Language Processing), 175–206. Dordrecht: Springer. DOI: 10.1007/978-3-319-90165-7_9.

Chatzichrisafis, Nikos, Richard Crouch, Tracy Holloway King, Rowan Nairn, Manny Rayner & Marianne Santaholma. 2007. Regression testing for grammar-based systems. In Tracy Holloway King & Emily M. Bender (eds.), *Proceedings of the GEAF07 workshop*, 128–143. Stanford: CSLI Publications. http://csli-publications.stanford.edu/GEAF/2007/geaf07-toc.html.

Clément, Lionel. 2019. Une étude de la coordination des propositions avec ellipse en français : Formalisation et application avec XLFG. *Langue française* 203. 35–52. DOI: 10.3917/lf.203.0035.

Clément, Lionel & Alexandra Kinyon. 2001. XLFG: An LFG parsing scheme for French. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '01 conference*, 47–65. Stanford: CSLI Publications.

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

Copperman, Max & Frederique Segond. 1996. Computational grammars and ambiguity: The bare bones of the situation. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '96 conference*. Stanford: CSLI Publications.

Crouch, Richard. 2006. Packed rewriting for mapping text to semantics and KR. In Miriam Butt, Mary Dalrymple & Tracy Holloway King (eds.), *Intelligent linguistic architectures: Variations on themes by Ronald M. Kaplan*. Stanford: CSLI Publications.

Crouch, Richard, Mary Dalrymple, Ronald M. Kaplan, Tracy Holloway King, John T. III Maxwell & Paula S. Newman. 2011. *XLE Documentation*. Xerox Palo Alto Research Center. Palo Alto, CA. https://ling.sprachwiss.uni-konstanz.de/pages/xle/doc/xle_toc.html.

Crouch, Richard & Tracy Holloway King. 2006. Semantics via f-structure rewriting. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '06 conference*, 145–165. Stanford: CSLI Publications.

Crouch, Richard & Tracy Holloway King. 2008. Type-checking in formally non-typed systems. In *Proceedings of the ACL workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, 3–4. Association for Computational Linguistics. DOI: 10.3115/1622110.1622112.

Crouch, Richard, Tracy Holloway King, John T. III Maxwell, Stefan Riezler & Annie Zaenen. 2004. Exploiting f-structure input for sentence condensation. In

Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '04 conference*, 167–187. Stanford: CSLI Publications.

Dalrymple, Mary. 2001. *Lexical Functional Grammar* (Syntax and Semantics 34). New York: Academic Press. DOI: 10.1163/9781849500104.

Dalrymple, Mary. 2006. How much can part-of-speech tagging help parsing? *Natural Language Engineering* 12. 373–389. DOI: 10.1017/s1351324905004079.

Dalrymple, Mary, Helge Dyvik & Tracy Holloway King. 2004. Copular complements: Closed or open? In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '04 conference*, 188–198. Stanford: CSLI Publications. http://csli-publications.stanford.edu/LFG/9/pdfs/lfg04ddk.pdf.

Dalrymple, Mary, Ronald M. Kaplan & Tracy Holloway King. 2004. Linguistic generalizations over descriptions. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '04 conference*, 199–208. Stanford: CSLI Publications.

Dalrymple, Mary, Ronald M. Kaplan, John T. III Maxwell & Annie Zaenen (eds.). 1995. *Formal issues in Lexical-Functional Grammar*. Stanford: CSLI Publications.

Dalrymple, Mary, John Lamping & Vijay Saraswat. 1993. LFG semantics via constraints. In *Proceedings of the 6th conference of the European chapter of the ACL (EACL 1993)*, 97–105. Association for Computational Linguistics. DOI: 10.3115/976744.976757.

Dalrymple, Mary, Maria Liakata & Lisa Mackie. 2006. Tokenization and morphological analysis for Malagasy. *International Journal of Computational Linguistics & Chinese Language Processing* 11(4). 315–332.

de Marneffe, Marie-Catherine, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre & Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik.

de Paiva, Valeria & Tracy Holloway King. 2008. Designing testsuites for grammar-based systems in applications. In *COLING 2008: Proceedings of the workshop on Grammar Engineering Across Frameworks*, 49–56. Manchester. DOI: 10.3115/1611546.1611553.

Dione, Cheikh M. Bamba. 2014. LFG parse disambiguation for Wolof. *Journal of Language Modelling* 2(1). 105–165. DOI: 10.15398/jlm.v2i1.81.

Dipper, Stefanie. 2003. Implementing and documenting large-scale grammars – German LFG. *Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung (AIMS)* 9(1).

Dost, Ascander & Tracy Holloway King. 2009. Using large-scale parser output to guide grammar development. In Tracy Holloway King & Marianne Santaholma (eds.), *Proceedings of the 2009 workshop on Grammar Engineering Across Frameworks (GEAF 2009)*, 63–70. Association for Computational Linguistics. DOI: 10.3115/1690359.1690367.

Dyvik, Helge. 1999. The universality of f-structure: Discovery or stipulation? The case of modals. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '99 conference*, 1–11. Stanford: CSLI Publications.

Dyvik, Helge, Gyri Smørdal Losnegaard & Victoria Rosén. 2019. Multiword expressions in an LFG grammar for Norwegian. In Yannick Parmentier & Jakub Waszczuk (eds.), *Representation and parsing of multiword expressions*. Berlin: Language Science Press. DOI: 10.5281/zenodo.2579037.

Dyvik, Helge, Paul Meurer, Victoria Rosén, Koenraad De Smedt, Petter Haugereid, Gyri Smørdal Losnegaard, Gunn Inger Lyse & Martha Thunes. 2016. NorGramBank: A 'deep' treebank for Norwegian. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk & Stelios Piperidis (eds.), *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC'16)*, 3555–3562. Portorož. http://www.lrec-conf.org/proceedings/lrec2016/summaries/943.html.

Earley, Jay. 1970. An efficient context-free parsing algorithm. *Communications of the ACM* 13(2). 94–102. DOI: 10.1145/362007.362035.

Fang, Ji & Tracy Holloway King. 2007. An LFG Chinese grammar for machine use. In Tracy Holloway King & Emily M. Bender (eds.), *Proceedings of the GEAF07 workshop*, 144–160. Stanford: CSLI Publications. http://csli-publications.stanford.edu/GEAF/2007/geaf07-toc.html.

Fellbaum, Christiane (ed.). 1998. *Wordnet: An electronic lexical database*. Cambridge, MA: The MIT Press. DOI: 10.7551/mitpress/7287.001.0001.

Findlay, Jamie Y., Roxanne Taylor & Anna Kibort. 2023. Argument structure and mapping theory. In Mary Dalrymple (ed.), *Handbook of Lexical Functional Grammar*, 699–778. Berlin: Language Science Press. DOI: 10.5281/zenodo.10185966.

Fokkens, Antske Sibelle. 2014. *Enhancing empirical research for linguistically motivated precision grammars*. Saarbrücken: Universität des Saarlandes. (Doctoral dissertation).

Forst, Martin. 2003. Treebank conversion – Establishing a testsuite for a broad-coverage LFG from the TIGER treebank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03) at EACL 2003*, 205–

216. Association for Computational Linguistics. https://www.aclweb.org/anthology/W03-2404.

Forst, Martin. 2007. Disambiguation for a linguistically precise German parser. *Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung (AIMS)* 13(3).

Forst, Martin, Tracy Holloway King & Tibor Laczkó. 2010. Particle verbs in computational LFGs: Issues from English, German, and Hungarian. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '10 conference*, 228–248. Stanford: CSLI Publications.

Fortmann, Christian & Martin Forst. 2004. An LFG grammar checker for CALL. In *Proceedings of the InSTIL/ICALL symposium: NLP and Speech Technologies in Advanced Language Learning Systems*.

Frank, Anette. 1996. A note on complex predicate formation: Evidence from auxiliary selection, reflexivization, passivization and past participle agreement in French and Italian. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '96 conference*. Stanford: CSLI Publications.

Frank, Anette, Tracy Holloway King, Jonas Kuhn & John T. III Maxwell. 1998. Optimality Theory style constraint ranking in large-scale LFG grammars. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '98 conference*, 1–16. Stanford: CSLI Publications.

Frank, Anette, Tracy Holloway King, Jonas Kuhn & John T. III Maxwell. 2001. Optimality Theory style constraint ranking in large-scale LFG grammars. In Peter Sells (ed.), *Formal and empirical issues in Optimality Theoretic syntax*, 367–397. Stanford: CSLI Publications.

Graham, Yvette. 2012. Deep syntax in statistical machine translation. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '12 conference*, 240–253. Stanford: CSLI Publications.

Graham, Yvette, Anton Bryl & Josef van Genabith. 2009. F-structure transfer-based statistical machine translation. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '09 conference*, 317–337. Stanford: CSLI Publications.

Graham, Yvette & Josef van Genabith. 2012. Exploring the parameter space in statistical machine translation via f-structure transfer. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '12 conference*, 254–270. Stanford: CSLI Publications.

Graver, Jenny. 2023. LFG and Celtic languages. In Mary Dalrymple (ed.), *Handbook of Lexical Functional Grammar*, 1369–1406. Berlin: Language Science Press. DOI: 10.5281/zenodo.10186004.

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

Halvorsen, Per-Kristian & Ronald M. Kaplan. 1995. Projections and semantic description in Lexical-Functional Grammar. In Mary Dalrymple, Ronald M. Kaplan, John T. III Maxwell & Annie Zaenen (eds.), *Formal issues in Lexical-Functional Grammar*, 279–292. Stanford: CSLI Publications.

Haug, Dag. 2023. LFG and Dependency Grammar. In Mary Dalrymple (ed.), *Handbook of Lexical Functional Grammar*, 1829–1859. Berlin: Language Science Press. DOI: 10.5281/zenodo.10186040.

Homola, Petr & Matt Coler. 2012. Machine translation using dependency representation. Presented at the LFG '12 Conference.

Ivanova, Angelina, Stephan Oepen, Rebecca Dridan, Dan Flickinger, Lilja Øvrelid & Emanuele Lapponi. 2016. On different approaches to syntactic analysis into bi-lexical dependencies: An empirical comparison of direct, PCFG-based, and HPSG-based parsers. *Journal of Language Modelling* 4(1). 113–144. DOI: 10.15398/jlm.v4i1.101.

Jing, Hongyan. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP'00)*. DOI: 10.3115/974147.974190.

Kaplan, Ronald M. 1987. Three seductions of computational psycholinguistics. In Peter Whitelock, Mary McGee Wood, Harold L. Somers, Rod Johnson & Paul Bennett (eds.), *Linguistic theory and computer applications*, 149–188. London: Academic Press. Reprinted in Dalrymple, Kaplan, Maxwell & Zaenen (1995: 339–367).

Kaplan, Ronald M. 2009. Deep natural language processing for web-scale search. Presented at the LFG '09 Conference.

Kaplan, Ronald M. & Joan Bresnan. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In Joan Bresnan (ed.), *The mental representation of grammatical relations*, 173–281. Cambridge, MA: The MIT Press. Reprinted in Dalrymple, Kaplan, Maxwell & Zaenen (1995: 29–130).

Kaplan, Ronald M., Richard Crouch, Tracy Holloway King, Michael Tepper & Danny Bobrow. 2005. A note-taking appliance for intelligence analysts. In *Proceedings of the International Conference on Intelligence Analysis.*

Kaplan, Ronald M. & Tracy Holloway King. 2003. Low-level markup and large-scale LFG grammar processing. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '03 conference*, 238–249. Stanford: CSLI Publications.

Kaplan, Ronald M. & John T. III Maxwell. 1996. *LFG Grammar Writer's Workbench.* Xerox Palo Alto Research Center. Palo Alto, CA. https://www.researchgate.net/profile/John_Maxwell5/publication/2760068_Grammar_Writer's_Workbench/links/0c96052405e97928e9000000.pdf.

Kaplan, Ronald M., John T. III Maxwell, Tracy Holloway King & Richard Crouch. 2004. Integrating finite-state technology with deep LFG grammars. In *Proceedings of the Workshop on Combining Shallow and Deep Processing for NLP at the European Summer School on Logic, Language, and Information (ESSLLI)*.

Kaplan, Ronald M., Klaus Netter, Jürgen Wedekind & Annie Zaenen. 1989. Translation by structural correspondences. In *Proceedings of the 4th conference of the European chapter of the ACL (EACL 1989)*, 272–281. Association for Computational Linguistics. DOI: 10.3115/976815.976852. Reprinted in Dalrymple, Kaplan, Maxwell & Zaenen (1995: 311–330).

Kaplan, Ronald M. & Paula S. Newman. 1997. Lexical resource reconciliation in the Xerox Linguistic Environment. In *Proceedings of the ACL workshop on Computational Environments for Grammar Development and Engineering*. Association for Computational Linguistics.

Kaplan, Ronald M. & Jürgen Wedekind. 1993. Restriction and correspondence-based translation. In *Proceedings of the 6th conference of the European chapter of the ACL (EACL 1993)*, 193–202. Association for Computational Linguistics. DOI: 10.3115/976744.976768.

Kaplan, Ronald M. & Jürgen Wedekind. 2023. Formal and computational properties of LFG. In Mary Dalrymple (ed.), *Handbook of Lexical Functional Grammar*, 1035–1082. Berlin: Language Science Press. DOI: 10.5281/zenodo.10185982.

Khader, I. R. M. A. K. 2003. *Evaluation of an English LFG-based grammar as error checker*. Manchester: University of Manchester. (M.Sc. thesis).

Kifle, Nazareth Amlesom. 2011. *Tigrinya applicatives in Lexical-Functional Grammar*. Bergen: University of Bergen. (Doctoral dissertation).

Kim, Roger, Mary Dalrymple, Ron Kaplan & Tracy Holloway King. 2003. Multilingual grammar development via grammar porting. In *Proceedings of the 17th Pacific Asia Conference on Language, Information and Computation*, 98–105.

King, Tracy Holloway. 2011. (Xx*-)linguistics: Because we love language. *Linguistic Issues in Language Technology: Interaction of Linguistics and Computational Linguistics* 6. DOI: 10.33011/lilt.v6i.1253.

King, Tracy Holloway. 2016. Theoretical linguistics and grammar engineering as mutually constraining disciplines. In Doug Arnold, Miriam Butt, Berthold Crysmann, Tracy Holloway King & Stefan Müller (eds.), *Proceedings of the joint 2016 conference on Head-Driven Phrase Structure Grammar and Lexical Functional Grammar*, 339–359. Stanford: CSLI Publications.

King, Tracy Holloway & Mary Dalrymple. 2004. Determiner agreement and noun conjunction. *Journal of Linguistics* 4(1). 69–104. DOI: 10.1017/s0022226703002330.

King, Tracy Holloway, Stefanie Dipper, Anette Frank, Jonas Kuhn & John T. III Maxwell. 2000. Ambiguity management in grammar writing. In *Proceedings of the Linguistic Theory and Grammar Implementation workshop at European Summer School in Logic, Language, and Information (ESSLLI-2000)*.

King, Tracy Holloway, Stefanie Dipper, Anette Frank, Jonas Kuhn & John T. III Maxwell. 2004. Ambiguity management in grammar writing. *Research on Language and Computation* 2. 259–280. DOI: 10.1023/b:rolc.0000016784.26446.98.

King, Tracy Holloway, Martin Forst, Jonas Kuhn & Miriam Butt. 2005. The feature space in parallel grammar writing. *Research on Language and Computation* 3(2). 139–163. DOI: 10.1007/s11168-005-1295-z.

King, Tracy Holloway & John T. III Maxwell. 2007. Overlay mechanisms for multi-level deep processing applications. In Tracy Holloway King & Emily M. Bender (eds.), *Proceedings of the GEAF07 workshop*, 182–2002. Stanford: CSLI Publications. http://csli-publications.stanford.edu/GEAF/2007/geaf07-toc.html.

Kipper, Karin, Hoa Trang Dang & Martha Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000)*.

Knight, Kevin & Daniel Marcu. 2000. Statistics-based summarization – step one: Sentence compression. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000)*.

Korsak, Katarzyna Magdalena. 2018. *LFG-based universal dependencies for Norwegian*. Oslo: University of Oslo. (MA thesis).

Krasnowska-Kieraś, Katarzyna & Agnieszka Patejuk. 2015. Integrating Polish LFG with external morphology. In Markus Dickinson, Erhard Hinrichs, Agnieszka Patejuk & Adam Przepiórkowski (eds.), *Proceedings of the 14th international workshop on Treebanks and Linguistic Theories (TLT14)*, 134–147.

Kuhn, Jonas. 2023. LFG, Optimality Theory and learnability of languages. In Mary Dalrymple (ed.), *Handbook of Lexical Functional Grammar*, 961–1032. Berlin: Language Science Press. DOI: 10.5281/zenodo.10185980.

Laczkó, Tibor & György Rákosi. 2008–2019. *HunGram: An XLE implementation*. Tech. rep. Debrecen: University of Debrecen.

Lenat, Doug. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11). DOI: 10.1145/219717.219745.

Levin, Beth. 1993. *English verb classes and alternations*. Chicago: University of Chicago Press.

Masuichi, Hiroshi, Tomoko Ohkuma, Hiroki Yoshimura & Yasunari Harada. 2003. Japanese parser on the basis of the Lexical-Functional Grammar formalism and its evaluation. In *Proceedings of the 17th Pacific Asia Conference on Language, Information and Computation*.

Maxwell, John T. III. 2006. Efficient generation from packed input. In Miriam Butt, Mary Dalrymple & Tracy Holloway King (eds.), *Intelligent linguistic architectures: Variations on themes by Ronald M. Kaplan*, 19–34. Stanford: CSLI Publications.

Maxwell, John T. III & Ronald M. Kaplan. 1989. An overview of disjunctive constraint satisfaction. In *Proceedings of the 4th International Workshop on Parsing Technologies (IWPT 1995)*, 18–27. Also published in Tomita (1991) as 'A Method for Disjunctive Constraint Satisfaction', and reprinted in Dalrymple, Kaplan, Maxwell & Zaenen (1995: 381–402).

Maxwell, John T. III & Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics* 19. 571–590.

McDonald, Ryan, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló & Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st annual meeting of the Association for Computational Linguistics*, 92–97. Association for Computational Linguistics.

Meßmer, Moritz & Mark-Matthias Zymla. 2018. The glue semantics workbench: A modular toolkit for exploring linear logic and glue semantics. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '18 conference*, 268–282. Stanford: CSLI Publications.

Meurer, Paul. 2009. A computational grammar for Georgian. In Peter Bosch, David Gabelaia & Jérôme Lang (eds.), *Logic, language, and computation: 7th International Tbilisi Symposium on Logic, Language, and Computation, TbiLLC 2007, Revised selected papers* (Springer Lecture Notes in Artificial Intelligence 5422), 1–15. Berlin: Springer. DOI: 10.1007/978-3-642-00665-4_1.

Minos, Panagiotis. 2014. *Development of a natural language parser for the LFG formalism*. Athens: National & Kapodistrian University of Athens. (MA thesis).

Mittendorf, Ingo & Louisa Sadler. 2006. A treatment of Welsh initial mutation. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '06 conference*, 343–364. Stanford: CSLI Publications.

Müller, Stefan. 1999. *Deutsche Syntax deklarativ: Head-Driven Phrase Structure Grammar für das Deutsche*. Tübingen: Max Niemeyer Verlag. DOI: 10.1515/9783110915990.

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

Oepen, Stephan, Helge Dyvik, Jan Tore Lønning, Erik Velldal, Dorothee Beermann, John Carroll, Dan Flickinger, Lars Hellan, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård & Victoria Rosén. 2004. *Som å kapp-ete med trollet?* Towards MRS-based Norwegian-English Machine Translation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*.

Ørsnes, Bjarne. 2006. Creating raising verbs: An LFG-analysis of the complex passive in Danish. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '06 conference*, 386–405. Stanford: CSLI Publications.

Patejuk, Agnieszka & Adam Przepiórkowski. 2012. Towards an LFG parser for Polish: An exercise in parasitic grammar development. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk & Stelios Piperidis (eds.), *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, 3849–3852. European Language Resources Association (ELRA).

Patejuk, Agnieszka & Adam Przepiórkowski. 2014. Synergistic development of grammatical resources: A valence dictionary, an LFG grammar, and an LFG structure bank for Polish. In *Proceedings of the 13th International Workshop on Treebanks and Linguistic Theories (TLT13)*, 113–126. Department of Linguistics (SfS), University of Tübingen.

Pollard, Carl & Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press & CSLI Publications.

Przepiórkowski, Adam, Elżbieta Hajnicz, Agnieszka Patejuk, Marcin Woliński, Filip Skwarski & Marek Świdziński. 2014. Walenty: Towards a comprehensive valence dictionary of Polish. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)*, 2785–2792.

Przepiórkowski, Adam & Agnieszka Patejuk. 2020. From Lexical Functional Grammar to Enhanced Universal Dependencies: The UD-LFG treebank of Polish. *Language Resources and Evaluation* 54. 185–221. DOI: 10.1007/s10579-018-9433-z.

Rädle, Roman, Michael Zöllner & Sebastian Sulger. 2011. eXLEpse: An Eclipse-based, easy-to-use editor for computational LFG grammars. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '11 conference*, 422–439. Stanford: CSLI Publications.

Reuer, Veit. 2003. Error recognition and feedback with Lexical Functional Grammar. *CALICO Journal* 20. 497–512. DOI: 10.1558/cj.v20i3.497-512.

Riezler, Stefan, Tracy Holloway King, Richard Crouch & Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for Lexical-Functional Grammar. In *Proceedings of the*

2003 Human Language Technology Conference of the North American chapter of the Association for Computational Linguistics, 197–204. DOI: 10.3115/1073445.1073471.

Riezler, Stefan, Tracy Holloway King, Ronald M. Kaplan, Richard Crouch, John T. III Maxwell & Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. Philadelphia: Association for Computational Linguistics.

Riezler, Stefan & John T. III Maxwell. 2006. Grammatical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, main conference*, 248–255. New York: Association for Computational Linguistics. DOI: 10.3115/1220835.1220867.

Rohrer, Christian & Martin Forst. 2006. Improving coverage and parsing quality of a large-scale LFG for German. In Miriam Butt, Mary Dalrymple & Tracy Holloway King (eds.), *Intelligent linguistic architectures: Variations on themes by Ronald M. Kaplan*. Stanford: CSLI Publications.

Rosén, Victoria. 2023. LFG treebanks. In Mary Dalrymple (ed.), *Handbook of Lexical Functional Grammar*, 1169–1206. Berlin: Language Science Press. DOI: 10.5281/zenodo.10185992.

Rosén, Victoria, Koenraad De Smedt, Paul Meurer & Helge Dyvik. 2012. An open infrastructure for advanced treebanking. In Jan Hajič, Koenraad De Smedt, Marko Tadić & António Branco (eds.), *Proceedings of the META-RESEARCH workshop on Advanced Treebanking at LREC'12*, 22–29. Istanbul: European Language Resources Association (ELRA).

Rosén, Victoria, Paul Meurer & Koenraad De Smedt. 2007. Designing and implementing discriminants for LFG grammars. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG '07 conference*, 397–417. Stanford: CSLI Publications.

Rosén, Victoria, Paul Meurer & Koenraad De Smedt. 2009. LFG Parsebanker: A toolkit for building and searching a treebank as a parsed corpus. In Frank Van Eynde, Anette Frank, Gertjan van Noord & Koenraad De Smedt (eds.), *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories (TLT7)*, 127–133. Utrecht: LOT.

Rypa, Marikka & Ken Feuerman. 1995. CALLE: An exploratory environment for foreign language learning. In V. Holland, J. Kaplan & M. Sams (eds.), *Intelligent language tutors: Theory shaping technology*, 55–76. Lawrence Erlbaum Associates.

Sagot, Benoît & Pierre Boullier. 2006. Deep non-probabilistic parsing of large corpora. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'06)*. Genoa: European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2006/pdf/806_pdf.pdf.

Sagot, Benoît, Lionel Clément, Éric De La Clergerie & Pierre Boullier. 2006. The Lefff 2 syntactic lexicon for French: Architecture, acquisition, use. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'06)*.

Sarveswaran, Kengatharaiyer & Miriam Butt. 2019. Computational challenges with Tamil complex predicates. In Miriam Butt, Tracy Holloway King & Ida Toivonen (eds.), *Proceedings of the LFG '19 conference*, 272–292. Stanford: CSLI Publications.

Schneider, David & Kathleen F. McCoy. 1998. Recognizing syntactic errors in the writing of second language learners. In *ACL '98/COLING '98: Proceedings of the 36th annual meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, 1198–1204. Montréal: Association for Computational Linguistics. DOI: 10.3115/980691.980765.

Sells, Peter (ed.). 2001. *Formal and empirical issues in Optimality Theoretic syntax*. Stanford: CSLI Publications.

Sheil, Beau & Bjarne Ørsnes. 2006. Using a large external dictionary in an LFG grammar: The STO experiments. In Miriam Butt, Mary Dalrymple & Tracy Holloway King (eds.), *Intelligent linguistic architectures: Variations on themes by Ronald M. Kaplan*, 167–198. Stanford: CSLI Publications.

Shemtov, Hadar. 1997. *Ambiguity management in natural language generation*. Stanford: Stanford University. (Doctoral dissertation).

Szűcs, Péter. 2019. Left dislocation in Hungarian. In Miriam Butt, Tracy Holloway King & Ida Toivonen (eds.), *Proceedings of the LFG '19 conference*, 293–313. Stanford: CSLI Publications.

Tomita, Masaru (ed.). 1991. *Current issues in parsing technology*. Dordrecht: Kluwer Academic Publishers. DOI: 10.1007/978-1-4615-3986-5.

Zaenen, Annie. 2023. Information structure. In Mary Dalrymple (ed.), *Handbook of Lexical Functional Grammar*, 823–853. Berlin: Language Science Press. DOI: 10.5281/zenodo.10185972.

Zweigenbaum, Pierre. 1991. Un analyseur pour grammaires lexicales-fonctionnelles. *TA Informations* 32. 19–34.