Grammar Prototyping and Testing with the LinGO Grammar Matrix Customization System

Emily M. Bender, Scott Drellishak, Antske Fokkens, Michael Wayne Goodman, Daniel P. Mills, Laurie Poulson, and Safiyyah Saleem
University of Washington, Seattle, Washington, USA
{ebender,sfd,goodmami,dpmills,lpoulson,ssaleem}@uw.edu,
afokkens@coli.uni-saarland.de

Abstract

This demonstration presents the LinGO Grammar Matrix grammar customization system: a repository of distilled linguistic knowledge and a web-based service which elicits a typological description of a language from the user and yields a customized grammar fragment ready for sustained development into a broad-coverage grammar. We describe the implementation of this repository with an emphasis on how the information is made available to users, including in-browser testing capabilities.

1 Introduction

This demonstration presents the LinGO Grammar Matrix grammar customization system[1] and its functionality for rapidly prototyping grammars. The LinGO Grammar Matrix project (Bender et al., 2002) is situated within the DELPH-IN[2] collaboration and is both a repository of reusable linguistic knowledge and a method of delivering this knowledge to a user in the form of an extensible precision implemented grammar. The stored knowledge includes both a cross-linguistic core grammar and a series of "libraries" containing analyses of cross-linguistically variable phenomena. The core grammar handles basic phrase types, semantic compositionality, and general infrastructure such as the feature geometry, while the current set of libraries includes analyses of word order, person/number/gender, tense/aspect, case, coordination, pro-drop, sentential negation, yes/no questions, and direct-inverse marking, as well as facilities for defining classes (types) of lexical entries and lexical rules which apply to those types. The grammars produced are compatible with both the grammar development tools and the grammar-based applications produced by DELPH-IN. The grammar framework used is Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) and the grammars map bidirectionally between surface strings and semantic representations in the format of Minimal Recursion Semantics (Copestake et al., 2005).

The Grammar Matrix project has three goals—one engineering and two scientific. The engineering goal is to reduce the cost of creating grammars by distilling the solutions developed in existing DELPH-IN grammars and making them easily available for new projects. The first scientific goal is to support grammar engineering for linguistic hypothesis testing, allowing users to quickly customize a basic grammar and use it as a medium in which to develop and test analyses of more interesting phenomena.[3] The second scientific goal is to use computational methods to combine the results of typological research and formal syntactic analysis into a single resource that achieves both typological breadth (handling the known range of realizations of the phenomena analyzed) and analytical depth (producing analyses which work together to map surface strings to semantic representations) (Drellishak, 2009).

2 System Overview

Grammar customization with the LinGO Grammar Matrix consists of three primary activities: filling out the questionnaire, preliminary testing of the grammar fragment, and grammar creation.

2.1 Questionnaire

Most of the linguistic phenomena supported by the questionnaire vary across languages along multiple dimensions. It is not enough, for example,

[1] http://www.delph-in.net/matrix/customize/
[2] http://www.delph-in.net
[3] Research of this type based on the Grammar Matrix includes (Crysmann, 2009) (tone change in Hausa) and (Fokkens et al., 2009) (Turkish suspended affixation).

Proceedings of the ACL 2010 System Demonstrations, pages 1–6,
Uppsala, Sweden, 13 July 2010.

simply to know that the target language has coordination. It is also necessary to know, among other things, what types of phrases can be coordinated, how those phrases are marked, and what patterns of marking appear in the language. Supporting a linguistic phenomenon, therefore, requires eliciting the answers to such questions from the user. The customization system elicits these answers using a detailed, web-based, typological questionnaire, then interprets the answers without human intervention and produces a grammar in the format expected by the LKB (Copestake, 2002), namely TDL (type description language).

The questionnaire is designed for linguists who want to create computational grammars of natural languages, and therefore it freely uses technical linguistic terminology, but avoids, when possible, mentioning the internals of the grammar that will be produced, although a user who intends to extend the grammar will need to become familiar with HPSG and TDL before doing so.

The questionnaire is presented to the user as a series of connected web pages. The first page the user sees (the "main page") contains some introductory text and hyperlinks to direct the user to other sections of the questionnaire ("subpages"). Each subpage contains a set of related questions that (with some exceptions) covers the range of a single Matrix library. The actual questions in the questionnaire are represented by HTML form fields, including: text fields, check boxes, radio buttons, drop-downs, and multi-select drop-downs. The values of these form fields are stored in a "choices file", which is the object passed on to the grammar customization stage.

2.1.1 Unbounded Content

Early versions of the customization system (Bender and Flickinger, 2005; Drellishak and Bender, 2005) only allowed a finite (and small) number of entries for things like lexical types. For instance, users were required to provide exactly one transitive verb type and one intransitive verb type. The current system has an iterator mechanism in the questionnaire that allows for repeated sections, and thus unlimited entries. These repeated sections can also be nested, which allows for much more richly structured information.

The utility of the iterator mechanism is most apparent when filling out the Lexicon subpage. Users can create an arbitrary number of lexical rule "slots", each with an arbitrary number of morphemes which each in turn bear any number of feature constraints. For example, the user could create a tense-agreement morphological slot, which contains multiple portmanteau morphemes each expressing some combination of tense, subject person and subject number values (e.g., French *-ez* expresses 2nd person plural subject agreement together with present tense).

The ability provided by the iterators to create unbounded content facilitates the creation of substantial grammars through the customization system. Furthermore, the system allows users to expand on some iterators while leaving others unspecified, thus modeling complex rule interactions even when it cannot cover features provided by these rules. A user can correctly model the morphotactic framework of the language using "skeletal" lexical rules—those that specify morphemes' forms and their co-occurrence restrictions, but perhaps not their morphosyntactic features. The user can then, post-customization, augment these rules with the missing information.

2.1.2 Dynamic Content

In earlier versions of the customization system, the questionnaire was static. Not only was the number of form fields static, but the questions were the same, regardless of user input. The current questionnaire is more dynamic. When the user loads the customization system's main page or subpages, appropriate HTML is created on the fly on the basis of the information already collected from the user as well as language-independent information provided by the system.

The questionnaire has two kinds of dynamic content: expandable lists for unbounded entry fields, and the population of drop-down selectors. The lists in an iterated section can be expanded or shortened with "Add" and "Delete" buttons near the items in question. Drop-down selectors can be automatically populated in several different ways.[4] These dynamic drop-downs greatly lessen the amount of information the user must remember while filling out the questionnaire and can prevent the user from trying to enter an invalid value. Both of these operations occur without refreshing the page, saving time for the user.

[4]These include: the names of currently-defined features, the currently-defined values of a feature, or the values of variables that match a particular regular expression.

2.2 Validation

It makes no sense to attempt to create a consistent grammar from an empty questionnaire, an incomplete questionnaire, or a questionnaire containing contradictory answers, so the customization system first sends a user's answers through "form validation". This component places a set of arbitrarily complex constraints on the answers provided. The system insists, for example, that the user not state the language contains no determiners but then provide one in the Lexicon subpage. When a question fails form validation, it is marked with a red asterisk in the questionnaire, and if the user hovers the mouse cursor over the asterisk, a pop-up message appears describing how form validation failed. The validation component can also produce warnings (marked with red question marks) in cases where the system can generate a grammar from the user's answers, but we have reason to believe the grammar won't behave as expected. This occurs, for example, when there are no verbal lexical entries provided, yielding a grammar that cannot parse any sentences.

2.3 Creating a Grammar

After the questionnaire has passed validation, the system enables two more buttons on the main page: "Test by Generation" and "Create Grammar". "Test by Generation" allows the user to test the performance of the current state of the grammar without leaving the browser, and is described in §3. "Create Grammar" causes the customization system to output an LKB-compatible grammar that includes all the types in the core Matrix, along with the types from each library, tailored appropriately, according to the specific answers provided for the language described in the questionnaire.

2.4 Summary

This section has briefly presented the structure of the customization system. While we anticipate some future improvements (e.g., visualization tools to assist with designing type hierarchies and morphotactic dependencies), we believe that this system is sufficiently general to support the addition of analyses of many different linguistic phenomena. The system has been used to create starter grammars for more than 40 languages in the context of a graduate grammar engineering course.

To give sense of the size of the grammars produced by the customization system, Table 1 compares the English Resource Grammar (ERG) (Flickinger, 2000), a broad-coverage precision grammar in the same framework under development since 1994, to 11 grammars produced with the customization system by graduate students in a grammar engineering class at the University of Washington. The students developed these grammars over three weeks using reference materials and the customization system. We compare the grammars in terms of the number types they define, as well as the number of lexical rule and phrase structure rule instances.[5] We separate types defined in the Matrix core grammar from language-specific types defined by the customization system. Not all of the Matrix-provided types are used in the definition of the language-specific rules, but they are nonetheless an important part of the grammar, serving as the foundation for further hand-development. The Matrix core grammar includes a larger number of types whose function is to provide disjunctions of parts of speech. These are given in Table 1, as "head types". The final column in the table gives the number of "choices" or specifications that the users gave to the customization system in order to derive these grammars.

3 Test-by-generation

The purpose of the test-by-generation feature is to provide a quick method for testing the grammar compiled from a choices file. It accomplishes this by generating sentences the grammar deems grammatical. This is useful to the user in two main ways: it quickly shows whether any ungrammatical sentences are being licensed by the grammar and, by providing an exhaustive list of licensed sentences for an input template, allows users to see if an expected sentence is not being produced.

It is worth emphasizing that this feature of the customization system relies on the bidirectionality of the grammars; that is, the fact that the same grammar can be used for both parsing and generation. Our experience has shown that grammar developers quickly find generation provides a more stringent test than parsing, especially for the ability of a grammar to model ungrammaticality.

3.1 Underspecified MRS

Testing by generation takes advantage of the generation algorithm include in the LKB (Carroll et al.,

[5] Serious lexicon development is taken as a separate task and thus lexicon size is not included in the table.

Language	Family	Lg-specific types	Matrix types	Head types	Lex rules	Phrasal rules	Choices
ERG	Germanic	3654	N/A	N/A	71	226	N/A
Breton	Celtic	220	413	510	57	49	1692
Cherokee	Iroquoian	182	413	510	95	27	985
French	Romance	137	413	510	29	22	740
Jamamadí	Arauan	188	413	510	87	11	1151
Lushootseed	Salish	95	413	510	20	8	391
Nishnaabemwin	Algonquian	289	413	510	124	50	1754
Pashto	Iranian	234	413	510	86	19	1839
Pali	Indo-Aryan	237	413	510	92	55	1310
Russian	Slavic	190	413	510	56	35	993
Shona	Bantu	136	413	510	51	9	591
Vietnamese	Austro-Asiatic	105	413	510	2	26	362
Average		182.9	413	510	63.5	28.3	1073.5

Table 1: Grammar sizes in comparison to ERG

1999). This algorithm takes input in the form of Minimal Recursion Semantics (MRS) (Copestake et al., 2005): a bag of elementary predications, each bearing features encoding a predicate string, a label, and one or more argument positions that can be filled with variables or with labels of other elementary predications.[6] Each variable can further bear features encoding "variable properties" such as tense, aspect, mood, sentential force, person, number or gender.

In order to test our starter grammars by generation, therefore, we must provide input MRSs. The shared core grammar ensures that all of the grammars produce and interpret valid MRSs, but there are still language-specific properties in these semantic representations. Most notably, the predicate strings are user-defined (and language-specific), as are the variable properties. In addition, some coarser-grained typological properties (such as the presence or absence of determiners) lead to differences in the semantic representations. Therefore, we cannot simply store a set of MRSs from one grammar to use as input to the generator.

Instead, we take a set of stored template MRSs and generalize them by removing all variable properties (allowing the generator to explore all possible values), leaving only the predicate strings and links between the elementary predications. We then replace the stored predicate strings with ones selected from among those provided by the user. Figure 1a shows an MRS produced by a grammar fragment for English. Figure 1b shows the MRS with the variable properties removed and the predicate strings replaced with generic place-holders. One such template is needed for every sentence type (e.g., intransitive, transitive,

a. ⟨ h1,e2, {h7:_cat_n_rel(x4:SG:THIRD),
h3:exist_q_rel(x4, h5, h6),
h1:_sleep_v_rel(e2:PRES, x4)},
{h5 qeq h7} ⟩
b. ⟨ h1,e2, {h7:#NOUN1#(x4),
h3:#DET1#(x4, h5, h6),
h1:#VERB#(e2, x4)},
{h5 qeq h7} ⟩

Figure 1: Original and underspecified MRS

negated-intransitive, etc.). In order to ensure that the generated strings are maximally informative to the user testing a grammar, we take advantage of the lexical type system. Because words in lexical types as defined by the customization system differ only in orthography and predicate string, and not in syntactic behavior, we need only consider one word of each type. This allows us to focus the range of variation produced by the generator on (a) the differences between lexical types and (b) the variable properties.

3.2 Test by generation process

The first step of the test-by-generation process is to compile the choices file into a grammar. Next, a copy of the LKB is initialized on the web server that is hosting the Matrix system, and the newly-created grammar is loaded into this LKB session.

We then construct the underspecified MRSs in order to generate from them. To do this, the process needs to find the proper predicates to use for verbs, nouns, determiners, and any other parts of speech that a given MRS template may require. For nouns and determiners, the choices file is searched for the predicate for one noun of each lexical noun type, all of the determiner predicates, and whether or not each noun type needs a determiner or not. For verbs, the process is more complicated, requiring valence information as well as predicate strings in order to select the correct MRS template. In order to get this information, the process traverses the type hierarchy above the verbal lexical

[6] This latter type of argument encodes scopal dependencies. We abstract away here from the MRS approach to scope underspecification which is nonetheless critical for its computational tractability.

types until it finds a type that gives valence information about the verb. Once the process has all of this information, it matches verbs to MRS templates and fills in appropriate predicates.

The test-by-generation process then sends these constructed MRSs to the LKB process and displays the generation results, along with a brief explanation of the input semantics that gave rise to them, in HTML for the user.[7]

4 Related Work

As stated above, the engineering goal of the Grammar Matrix is to facilitate the rapid development of large-scale precision grammars. The starter grammars output by the customization system are compatible in format and semantic representations with existing DELPH-IN tools, including software for grammar development and for applications including machine translation (Oepen et al., 2007) and robust textual entailment (Bergmair, 2008).

More broadly, the Grammar Matrix is situated in the field of multilingual grammar engineering, or the practice of developing linguistically-motivated grammars for multiple languages within a consistent framework. Other projects in this field include ParGram (Butt et al., 2002; King et al., 2005) (LFG), the CoreGram project[8] (e.g., (Müller, 2009)) (HPSG), and the MetaGrammar project (de la Clergerie, 2005) (TAG).

To our knowledge, however, there is only one other system that elicits typological information about a language and outputs an appropriately customized implemented grammar. The system, described in (Black, 2004) and (Black and Black, 2009), is called PAWS (Parser And Writer for Syntax) and is available for download online.[9] PAWS is being developed by SIL in the context of both descriptive (prose) grammar writing and "computer-assisted related language adaptation", the practice of writing a text in a target language by starting with a translation of that text in a related source language and mapping the words from target to source. Accordingly, the output of PAWS consists of both a prose descriptive grammar and an implemented grammar. The latter is in the format required by PC-PATR (McConnel, 1995), and is used primarily to disambiguate morphological analyses of lexical items in the input string.

Other systems that attempt to elicit linguistic information from a user include the Expedition (McShane and Nirenburg, 2003) and Avenue projects (Monson et al., 2008), which are specifically targeted at developing machine translation for low-density languages. These projects differ from the Grammar Matrix customization system in eliciting information from native speakers (such as paradigms or translations of specifically tailored corpora), rather than linguists. Further, unlike the Grammar Matrix customization system, they do not produce resources meant to sustain further development by a linguist.

5 Demonstration Plan

Our demonstration illustrates how the customization system can be used to create starter grammars and test them by invoking test-by-generation. We first walk through the questionnaire to illustrate the functionality of libraries and the way that the user interacts with the system to enter information. Then, using a sample grammar for English, we demonstrate how test-by-generation can expose both overgeneration (ungrammatical generated strings) and undergeneration (gaps in generated paradigms). Finally, we return to the questionnaire to address the bugs in the sample grammar and retest to show the result.

6 Conclusion

This paper has presented an overview of the LinGO Grammar Matrix Customization System, highlighting the ways in which it provides access to its repository of linguistic knowledge. The current customization system covers a sufficiently wide range of phenomena that the grammars it produces are non-trivial. In addition, it is not always apparent to a user what the implications will be of selecting various options in the questionnaire, nor how analyses of different phenomena will interact. The test-by-generation methodology allows users to interactively explore the consequences of different linguistic analyses within the platform. We anticipate that it will, as a result, encourage users to develop more complex grammars within the customization system (before moving on to hand-editing) and thereby gain more benefit.

[7] This set-up scales well to multiple users, as the user's interaction with the LKB is done once per customized grammar, providing output for the user to peruse as his or her leisure. The LKB process does not persist, but can be started again by reinvoking test-by-generation, such as when the user has updated the grammar definition.

[8] http://hpsg.fu-berlin.de/Projects/core.html

[9] http://www.sil.org/computing/catalog/show_software.asp?id=85

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0644097. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Emily M. Bender and Dan Flickinger. 2005. Rapid prototyping of scalable grammars: Towards modularity in extensions to a language-independent core. In *Proc. of IJCNLP-05 (Posters/Demos)*.

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2002. The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In *Proc. of the Workshop on Grammar Engineering and Evaluation at COLING 2002*, pages 8–14.

Richard Bergmair. 2008. Monte Carlo semantics: McPIET at RTE4. In *Text Analysis Conference (TAC 2008) Workshop-RTE-4 Track. National Institute of Standards and Technology*, pages 17–19.

Cheryl A. Black and H. Andrew Black. 2009. PAWS: Parser and writer for syntax: Drafting syntactic grammars in the third wave. In *SIL Forum for Language Fieldwork*, volume 2.

Cheryl A. Black. 2004. Parser and writer for syntax. Paper presented at the International Conference on Translation with Computer-Assisted Technology: Changes in Research, Teaching, Evaluation, and Practice, University of Rome "La Sapienza", April 2004.

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. In *Proc. of the Workshop on Grammar Engineering and Evaluation at COLING 2002*, pages 1–7.

John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznański. 1999. An efficient chart generator for (semi-) lexicalist grammars. In *Proc. of the 7th European workshop on natural language generation (EWNLG99)*, pages 86–95.

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

Ann Copestake. 2002. *Implementing Typed Feature Structure Grammars*. CSLI, Stanford.

Berthold Crysmann. 2009. Autosegmental representations in an HPSG for Hausa. In *Proc. of the Workshop on Grammar Engineering Across Frameworks 2009*.

Éric Villemonte de la Clergerie. 2005. From metagrammars to factorized TAG/TIG parsers. In *Proc. of IWPT'05*, pages 190–191.

Scott Drellishak and Emily M. Bender. 2005. A coordination module for a crosslinguistic grammar resource. In Stefan Müller, editor, *Proc. of HPSG 2005*, pages 108–128, Stanford. CSLI.

Scott Drellishak. 2009. *Widespread But Not Universal: Improving the Typological Coverage of the Grammar Matrix*. Ph.D. thesis, University of Washington.

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

Antske Fokkens, Laurie Poulson, and Emily M. Bender. 2009. Inflectional morphology in Turkish VP-coordination. In Stefan Müller, editor, *Proc. of HPSG 2009*, pages 110–130, Stanford. CSLI.

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

Stephen McConnel. 1995. *PC-PATR Reference Manual*. Summer Institute for Linguistics. http://www.sil.org/pcpatr/manual/pcpatr.html.

Marjorie McShane and Sergei Nirenburg. 2003. Parameterizing and eliciting text elements across languages for use in natural language processing systems. *Machine Translation*, 18:129–165.

Christian Monson, Ariadna Font Llitjs, Vamshi Ambati, Lori Levin, Alon Lavie, Alison Alvarez, Roberto Aranovich, Jaime Carbonell, Robert Frederking, Erik Peterson, and Katharina Probst. 2008. Linguistic structure and bilingual informants help induce machine translation of lesser-resourced languages. In *LREC'08*.

Stefan Müller. 2009. Towards an HPSG analysis of Maltese. In Bernard Comrie, Ray Fabri, Beth Hume, Manwel Mifsud, Thomas Stolz, and Martine Vanhove, editors, *Introducing Maltese linguistics. Papers from the 1st International Conference on Maltese Linguistics*, pages 83–112. Benjamins, Amsterdam.

Stephan Oepen, Erik Velldal, Jan Tore Lnning, Paul Meurer, Victoria Rosn, and Dan Flickinger. 2007. Towards hybrid quality-oriented machine translation. On linguistics and probabilities in MT. In *11th International Conference on Theoretical and Methodological Issues in Machine Translation*.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago, IL.