CUSTOMIZING THE LINGO GRAMMAR MATRIX MORPHOLOGY

SARAH R. MOELLER
University of Colorado Boulder

The LinGO Grammar Matrix is rapid development grammatical analyzer that is enabled by a customization system, including an online questionnaire. The questionnaire allows any linguist to quickly build a starter grammar for any language. However, its documentation is not easily navigable. This paper describes the process of customizing the morphological section of the grammar, using Lezgi case inflection as an example. It outlines how suggestions for helping potential users to gain confidence in using the Matrix.

Keywords: computational linguistics, grammatical analysis, Lezgi, morphology, LingGO Grammar Matrix

1. INTRODUCTION

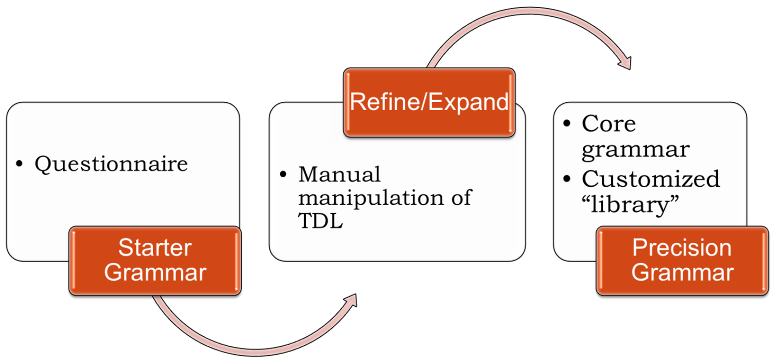
The LinGO Grammar Matrix (Bender, Flickinger & Oepen 2002; Bender et al. 2010) is "an open-source online repository of grammatical analyses which facilitates the rapid development of linguistically-motivated deep grammars compatible" (Bender 2014: 2447). This rapid development is enabled by a customization system, including an online questionnaire. The questionnaire allows any linguist to quickly build a starter grammar for any language, but its documentation is not easily navigable. Easy-to-read instructions could encourage linguists to begin implementing and testing the LinGO Grammar Matrix on more languages. This paper describes the LinGO Grammar Matrix and how to use its questionnaire (sections 2 and 3) with a case study on Lezgi case morphology (section 4). Section 5 outlines a presentation that would allow potential users to gain confidence in the customization system more quickly than is now possible.

2. THE LINGO GRAMMAR MATRIX

The LinGO Grammar Matrix (LGM) "combines a core grammar providing constraints and structures which are cross-linguistically useful with a series of libraries of analyses of cross-linguistically variable phenomena" (Bender 2014: 2447). The core grammar consists of lexical rules believed to describe universal linguistic phenomena such as case marking or basic word order. The phenomenon-specific libraries cover language-specific grammatical rules. Grammar rules and semantics are conveyed in Head-Driven Phrase Structure Grammar (HPSG) and Minimal Recursion Semantics (MRS) and represented in a machine-readable language called Type Definition Language (TDL) and displayed in the Linguistic Knowledge Builder (LKB), a software environment that essentially serves as a graphic user interface. Users can customize libraries by refining the core grammar rules. The customization process does not directly change the core grammar. Instead, new rules are created which take a core grammar rule as "supertype". The new rule inherits all features of the supertype and extends or constrains it. For example, customizing for strict SOV word order would create a new rule that inherits all constrains the "supertype" word order rule to allow only SOV word order.

The customization system builds a deep precision grammar for any language via two steps. The first step builds a "starter grammar" using a customization system questionnaire (Bender 2014; Goodman 2013). Completing the questionnaire produces a small implemented "starter grammar" that describes most phenomena encountered in simple main clauses such as basic word order, case marking, subject agreement, and TAM marking. The second step begins once the pages of the questionnaire have been completed. To extend the grammar to phenomena not handled by the questionnaire, such as word order of questions, the TDL files must be edited directly.

FIGURE 1. LINGO GRAMMAR MATRIX STEPS TOWARDS A PRECISION GRAMMAR



3. THE CUSTOMIZATION QUESTIONNAIRE

LGM's customization questionnaire is an online interface. Its pages dedicated to topics such as word order, number, case, and "other features" guide users to provide a high-level description of a language's main clause structure. For example, the first part of the Case page presents a choice of nine case marking systems, including "no case", nominative-accusative, and ergative-absolutive, and the second part builds an expandable list of any other cases. The two parts together are added to the list of semantic/syntactic features that can be assigned to lexical items on the Lexicon page or to morphemes on the Morphology page.

The Morphology page is the most complex part of the questionnaire. The LGM treats morphology inferentially (Goodman 2013: 5–6), defining relationships between roots and their inflected forms via morphological rules. The morphological rules are constructed in a three-level hierarchy: Position Classes, Lexical Rule Types, and Lexical Rule Instances. Position Classes group morphemes by the slot they occupy on the inflected word. Within Position Classes, morphemes with common syntactic/semantic features or morphotactic restrictions are grouped in Lexical Rule Types. A Rule Type may serve as supertype to other Rules that will inherit the supertype's features and morphotactics but may have additional features or constraints. These features and constraints do not target individual orthographic representations (essentially spelling changes such as the double consonants added to some English verbs after *–ing*: *hop* → *hopping*). Orthographic idiosyncracies are specified by Lexical Rule Instances which inherit all features from its Lexical Rule Type.

4. CUSTOMIZING THE LINGO GRAMMAR MATRIX FOR LEZGI

Lezgi (or Lezgian) [lez] is a Lezgic language belonging to the Nakh-Daghestanian family (Simons & Fennig 2018). It is spoken by over 600,000 people in the Daghestan Republic of Russia and Azerbaijan. Like many languages in the Caucasus Mountains, Lezgi exhibits ergative morphology as shown in sentence (1) and (2).

(1)	itim-di	sev	gata-na
	man-ERG	bear.ABS	beat-AORIST

'The man beat the bear.'

(2)	sev	itim-di	gata-na
	bear.ABS	man-ERG	beat-AORIST

'The man beat the bear.'

4.1 LEZGI CASE

Lezgi has fourteen cases marked by suffixes. Absolutive case is unmarked and the ergative suffix attaches directly to the noun stem. Other cases incrementally add suffixes to the ergative (what Haspelmath (1993: 74) calls the oblique stem). The genitive, dative, postessive, subessive, superessive, and inessive add one suffix (the inessive can be analyzed as a null suffix). The elative and directive meaning are added to the last four suffixes as a third suffix. Table 1 illustrates this pattern.

TABLE 1. LEZGI CASE INFLECTION

	'bear'	'man'	
absolutive	*sev*	*itim*	'the N'
ergative	*sev-re*	*itim-di*	'the N'
genitive	*sev-re-n*	*itim-di-n*	'of the N'
dative	*sev-re-z*	*itim-di-z*	'to the N'
addessive (adess)	*sev-re-v*	*itim-di-v*	'at the N'
adelative (adel)	*sev-re-v-ay*	*itim-di-v-ay*	'from the N'
addirective (addir)	*sev-re-v-di*	*itim-di-v-di*	'toward the N'
postessive (poess)	*sev-re-q*	*itim-di-q*	'behind the N'
postelative (poel)	*sev-re-q-ay*	*itim-di-q-ay*	'from behind the N'
postdirective (podir)	*sev-re-q-di*	*itim-di-q-di*	'to behind the N'
subessive (sbess)	*sev-re-k*	*itim-di-k*	'under the N'
subelative (sbel)	*sev-re-k-ay*	*itim-di-k-ay*	'from under the N'
subdirective (sbdir)	*sev-re-k-di*	*itim-di-k-di*	'to under the N'
superessive (spess)	*sev-re-l*	*itim-di-l*	'on the N'
superelative (spel)	*sev-re-l-ay*	*itim-di-l-ay*	'off the N'
superdirective (spdir)	*sev-re-l-di*	*itim-di-l-di*	'onto the N'
inessive (iness)	*sev-re*	*itim-di*	'in the N'
inelative (inel)	*sev-rə-y*	*itim-di-y*	'out of the N'

Table 2 shows a position class chart for the nouns in Table 1, including number suffixes that attach directly to the noun stem.

TABLE 2. POSITION CLASS CHART FOR *-re* AND *-di* NOUNS

0 Stem	+1 number	+2 case1	(+3) case2	(+4) case3
	-ø 'sg' -ar 'pl'	-ø 'abs' -re/di 'erg'	-n 'gen' -z 'dat' -v 'adess' -q 'poess' -k 'sbess' -l 'sress' -ø 'iness'	-ay 'ad/po/sb/sr/in -el' -di 'ad/po/sb/sr -dir'

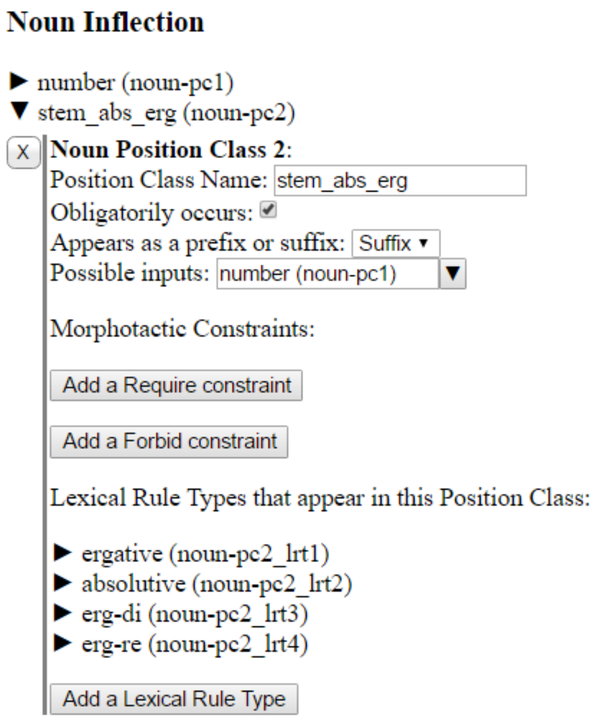
Table 2 simplifies Lezgi noun morphology in two ways. First, morphophonological changes that produce two plural suffixes and the inelative are ignored because the LGM does not handle morphophonology (Goodman 2013: 11). Users are expected to provide underlying phonological representations. Second, the ergative case has eight more suffixes. Although Lezgi is described as having no noun classes and no agreement (Haspelmath 1993; Manning 1994), these suffixes are selected by semantic and phonological factors that indicate possible remnants of noun class agreement. For example, the *-a* ergative suffix occurs on personal names ending in a consonant, but the *–i* ergative suffix occurs on the same word if it is used as a personal name (e.g. cükwer-a 'of the flowers' but Cükwer-i 'Flower's').

4.2 CUSTOMIZING MORPHOLOGICAL RULES

In order to describe the Lezgi case morphology in Table 2 three pages of the questionnaire were modified. First, an ergative-absolutive system was chosen and the other twelve cases were declared on the Case page. Second, noun roots were added to the Lexicon as two noun types: one that takes *-re* 'ergative' suffix and one that takes *-di* 'ergative'. It is worth noting that since the Lexicon allows semantic/syntactic features to be assigned to noun types a new user might mistakenly assign the absolutive case as a feature of bare noun roots. Subsequently adding a ergative suffix would give the noun two clashing case features, which would prevent parsing because of LGM incremental approach to morphology (Goodman 2013: 5–6). Instead, the absolutive case should be added by morphological rule as a zero affix.

Finally, on the Morphology page, the case suffixes were defined and morphological rules created to describe how they attach to noun roots. A Position Class (pc) was created for number and for ergative and absolutive case. As Figure 2 shows, the case (pc2) takes number (pc1) as its input and is obligatorily attached as a suffix. Since the number suffix attaches obligatorily to all nouns, together the Position Classes dictate that a noun must be inflected for both number and case, in that order.

FIGURE 2. LEZGI CASE POSITION CLASS



Noun Inflection

▶ number (noun-pc1)
▼ stem_abs_erg (noun-pc2)

Noun Position Class 2:
Position Class Name: stem_abs_erg
Obligatorily occurs: ✓
Appears as a prefix or suffix: Suffix
Possible inputs: number (noun-pc1)

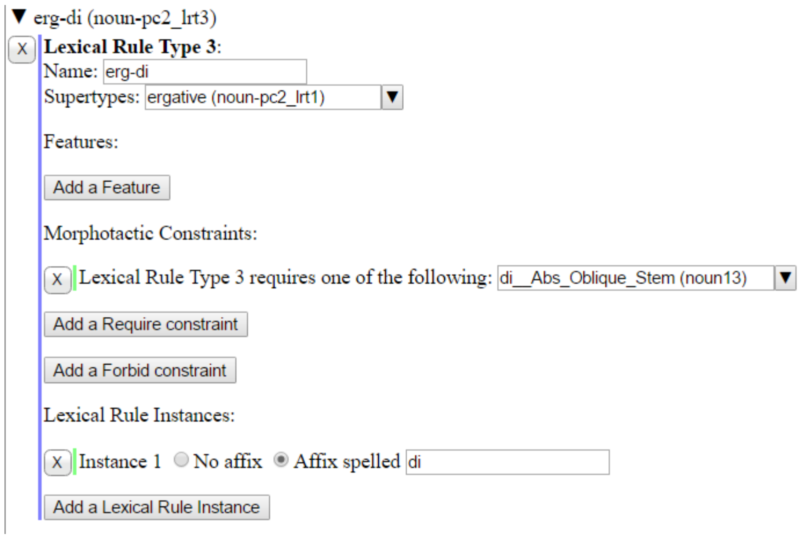
Morphotactic Constraints:

Lexical Rule Types that appear in this Position Class:

▶ ergative (noun-pc2_lrt1)
▶ absolutive (noun-pc2_lrt2)
▶ erg-di (noun-pc2_lrt3)
▶ erg-re (noun-pc2_lrt4)

The singular and plural morphemes are defined in the pc1 as two Lexical Rule Types each with one Lexical Rule Instance. Similarly, in pc2, the absolutive case is defined in a Lexical Rule Type with one Lexical Rule Instance (“No Affix”). The ergative case is defined as a hierarchy of three Lexical Rule Types. The first rule (noun-pc2_lrt1) serves as the supertype and its semantic feature (ergative case) is inherited by the other two: erg-di (noun-pc2_lrt3) and erg-re (noun-pc2_lrt4). These two rules each provide a Lexical Rule Instance: *di* and *re*. The two subrules differ only in Morphotactic Constraints. The constraints on erg-di (noun-pc2_lrt3) enforces cooccurrence with the -*di* noun type in the Lexicon, as Figure 3 shows, and erg-re (noun-pc2_lrt4) enforces co-occurrence with -*re* nouns.

FIGURE 3. ERGATIVE -*di* LEXICAL RULE TYPE


▼ erg-di (noun-pc2_lrt3)

Lexical Rule Type 3:

Name: erg-di

Supertypes: ergative (noun-pc2_lrt1)

Features:

Morphotactic Constraints:

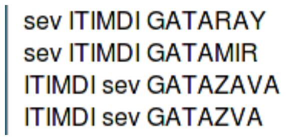
Lexical Rule Type 3 requires one of the following: di__Abs_Oblique_Stem (noun13)

Lexical Rule Instances:

Instance 1 No affix [✓ Affix spelled] di

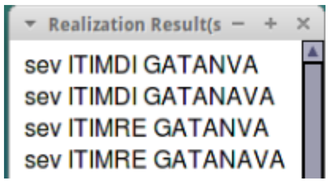
Morphotactic Constraints were intended to prevent underinflection and overinflection (Goodman 2013: 7, 32) and this can be tested in LKB by generation. Putting the morpheme variants in a hierarchy of Lexical Rule Types eliminates overinflection, as shown in Figure 4 (cf. Figure 5).

FIGURE 4. CORRECT CASE MORPHOLOGY GENERATION


```
sev ITIMDI GATARAY
sev ITIMDI GATAMIR
ITIMDI sev GATAZAVA
ITIMDI sev GATAZVA
```

If a new user did not understand the "supertype" hierarchy and instead represented ergative case as two Lexical Rule Instances in a single Lexical Rule Type, the generation would yield the overinflection in Figure 5. The suffix -*di* should only appear on *itim* 'man' (and -*re* only on *sev* 'bear').

FIGURE 5. INCORRECT CASE MORPHOLOGY GENERATION



```
sev ITIMDI GATANVA
sev ITIMDI GATANAVA
sev ITIMRE GATANVA
sev ITIMRE GATANAVA
```

Except for bipartite verb stems, Morphotactic Constraints cannot target Lexical Rule Instances via the questionnaire. This limitation raises two questions. First, why allow multiple Lexical Rule Instances in one Lexical Rule Type in the questionnaire? Except in rare cases of free variation, it seems unlikely that orthographic variants can be properly handled by one Lexical Rule Type. At the same time, using multiple Lexical Rule Instances in a single Lexical Rule Type would seem to new users a deceptively simple solution for handling morphophonology or agreement paradigms.

Second, what advantage does the supertype hierarchy give over unrelated Lexical Rule Types? Creating two Lexical Rule Types for the ergative case and assigning them the same syntactic/semantic feature produces the same results. Morphotactic constraints would still forbid ergative suffixes to co-occur on the same noun and overinflection with the absolutive case is disallowed because all the case Rule Types belong to the same Position Class. Neither the questionnaire nor Goodman (2013) provide insights into why one configuration might be preferred.

4.3 LEZGI CASE MORPHOLOGY: FURTHER STEPS

Since the LGM takes an incremental-inferential approach to morphology, attempting to define all columns in Table 2 as Position Classes would cause each additional suffix to clash with the case feature assigned to the previous suffix. Handling the remaining 12 cases in Lezgi requires a decision whether to reflect the morphology in Table 2 or simplify the morphological rules. Instead of defining all 14 cases on the Case page, the oblique cases could be defined on the Other Features page. This would create new semantic "case" features that would not clash with the erg-abs Position Class. This approach would reflect Lezgi's sequence of case suffixes. A second approach would treat each sequence of suffixes as a single morpheme. Each case "morpheme" would be added as a Lexical Rule Type to the second Position Class. This approach does not accurately describe Lezgi morphology, but it is simpler. As Goodman (2013: 3) points out, the goal of computational grammars is to "parse and generate valid sentences" rather than "capture the behavior of interesting linguistic phenomena."

5. TOWARDS GREATER ACCESSIBILITY

LGM's developers wish to collect language specific libraries (Bender 2014) so moving the state of the art forward means more data from more languages. Customizing a precision grammar requires a great deal of time and some specialized knowledge. By eliminating the need to know HPSG and TDL, LGM's customization questionnaire has reduced the learning curve from perhaps 80+ workhours to about 30. However, much of the learning time must still be spent finding and identifying resources for learning the customization process. On the LGM homepage, publications are sorted chronologically, not by relevance to learners. The wiki provides some instructions but not in logical sequence. Certain instructions lack screenshots that would allow new users to compare their progress against desired results. Other instructions appear to refer to previous versions of the LKB interface. Online courses that teach the LGM assume a background in grammar engineering or HPSG.

This section assembles resources from literature about the LGM, online course materials, and the experience of the author as a new user. It summarizes basic information that is needed when beginning to customize the LGM. It is imagined as an "orientation" page for the questionnaire, allowing potential users to assess what resources they need gather and what steps they will take as they proceed.

"BEFORE YOU START"

These instructions assume you are a trained linguist. It does not assume you are familiar with computational linguistics, HPSG, or MRS. It does assume you are somewhat familiar with Unix.

OVERVIEW TO CUSTOMIZING A STARTER GRAMMAR

There are two presteps to building a starter grammar with the LGM's customization questionnaire: 1) download the software, 2), choose a language and fill out the General Information page. The rest of the process is an iterative cycle that will continue until all pages in the questionnaire have been completed and tested.

OVERVIEW OF CUSTOMIZATION CYCLE

1. Describe one to three phenomena by filling out the relevant pages of questionnaire.
2. Add grammatical and ungrammatical sentences that illustrate the phenomena to your test suite.
3. On the Lexicon page, enter any new vocabulary in those sentences.
4. On the Morphology page, add/edit Position Classes, Lexical Rule Types (morphemes sharing semantic/syntactic features and morphotactic constraints), and Lexical Rule

Instances (orthographic representation of morphemes) to cover all new morphology in the test suite.

5. Download and save latest version of the choices file.
6. Generate a new version of the starter grammar.
7. Unzip the grammar and load it into LKB software.
8. Try parsing some individual sentences from your test suite.
9. Generate sentences to examine morphology or try batch parsing the test suite.
10. Based on generation and batch parsing results, edit the questionnaire.
11. Repeat steps #6-11 until satisfied, then proceed to step #1 in order to add new phenomena.

NECESSARY RESOURCES

- Reference grammar for your chosen language
- Software
- Some knowledge about HPSG and MRS
- Understanding of the LGM's approach to morphology

DOWNLOADING SOFTWARE

- List of Software:
  - LKB – this is where you test the latest version of your grammar; it parses and generates sentences, displays phrase structure trees, attribute value matrices (AVM), and semantic representations. It only runs in Ubuntu.
  - Virtual Box – allows you to run the Ubuntu operating system inside your Windows or OS X computer.
  - Ubuntu – you will download a package called Ubuntu+LKB that has LKB already installed in it.
  - emacs – this program already installed in Ubuntu. It launches LKB and sometimes displays useful information while LKB is running (e.g. Debug report). It has a built-in tutorial that you may find useful. However, you really only need one command: M-x lkb (type ALT+x, then lkb) to launch LKB.
- Downloading Software
  - It is not uncommon to run into complications while downloading software. This can be a discouraging way to start, so have someone nearby who can help.
  - Go to: http://depts.washington.edu/uwcl/twiki/bin/view.cgi/Main/KnoppixLKBVboxApp. Follow the instructions under "Install VirtualBox" and "Set up the Ubuntu+LKB appliance".
  - Create or choose a folder to keep related files and follow the instructions for "Setting up a shared folder" at the bottom of the link.

- Becoming familiar with LKB
    - Take time to play around with LKB and become comfortable with it. Go to: http://courses.washington.edu/ling567/lab1.html. Follow the instructions under "Grammar Customization: Get a small grammar for English" and "LKB: Getting Started" (ignore the first step). If you are not familiar with Ubuntu, we highly recommend that you find someone to guide you through these steps the first time. You will repeat them often.

LEARNING HPSG AND MRS

It is possible to customize a starter grammar without knowing HPSG or MRS. It is difficult, however, to identify syntactic or semantic problems and impossible to go beyond the questionnaire if you cannot decipher the AVM used to display HPSG and MRS in LKB.

- HPSG introduction: Levine, Robert D. 2003. Head-Driven Phrase Structure Grammar. *Encyclopedia of Cognitive Science*.
- AVM Cheat Sheet:
    - SYNSEM – contains LOC and NONLOC properties
    - LOC(al) – locally relevant properties of the word, phrase, or clause; identifies lexical or phrasal properties via CAT
    - NONLOC(al) – information that extends over a larger syntactic domain, (e.g. long distance dependency)
    - CAT – contains VAL and HEAD values
    - VAL(ence) – valence information; specifies subject as SUBJ, non-subject arguments as COMPs
    - HEAD – properties of lexical items shared by the phrase it heads, (e.g. (POS), CASE, AUX(iliary), etc.)
    - ARG-ST(ructure) – verb's argument structure; identical to the list of COMPs + subject
    - SPR – modifier/specifier
    - CONX – contextual information
    - C-CONT or CONT(ent) – logical aspects of semantic interpretation; relations among agent, patient, and so on
    - INDEX – part of the CONT for nominals; encodes reference

MRS reference guide (somewhat out of date): Flickinger, Dan, Emily M. Bender and Stephan Oepen. 2003. *MRS in the LinGO Grammar Matrix: A Practical User's Guide.* Ms.

- General introduction: Copestake, Ann, Dan Flickinger, Carl Pollard & Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation* 3.281–332.

REPRESENTING MORPHOLOGY

The Morphology page will describe your chosen language's morphological rules. This is the most complex part of the questionnaire, but once you learn the basic principles, it is fairly simple to use.

- LGM Morphology overview with some examples (read all of sections 2 & 3): Goodman, Michael Wayne. 2013. Generation of machine-readable morphological rules from human readable input. (Ed.) Sanghoun Song & Joshua Crowgey. *UW Working Papers in Linguistics* 30. http://depts.washington.edu/uwwpl/vol30/goodman_2013.pdf (23 September, 2016)

GETTING STARTED

Start by taking notes on the language's grammar and constructing a small test suite of example sentences. Follow these instructions: http://hpsg.stanford.edu/05inst/prep.html.

NOTES ON TEST SUITE

As you proceed, you will need to test your starter grammar against real language data, so construct sentences that demonstrate *and* violate the phenomena described in the questionnaire. It is wise to start with simple sentences and reuse words as much as possible. Avoid nonverbal predicates and copulae until later (e.g., *The cat is old.*). Here are some sample sentences: http://moin.delph-in.net/MatrixMrsTestSuite.

Here, with examples, are all grammatical phenomena that the test suite should eventually cover: http://compling.hss.ntu.edu.sg/courses/hg7021/testsuites.html#phenomena. Skim this page and start thinking about how to illustrate these phenomena using minimal vocabulary and morphology.

- Test Suite Specifications
  - If the language is not normally written with spaces between words, add spaces.
  - All examples should be complete sentences.
  - Each sentence should be on a new line.
  - Ungrammatical examples should generally have only one thing wrong with them.
  - Since the system is not designed to handle morphophonology, choose inflections that do not undergo morphophonological changes or else only write underlying phonemic representations.
  - Save your test suite as a plain text file (.txt).
  - Encode the file as utf-8.

NOTES ON TEST SUITE FORMAT

If you wish your starter grammar to be available online for future users, the test suite needs to be formatted as described here:
http://compling.hss.ntu.edu.sg/courses/hg7021/testsuites.html#formatting.
Otherwise, simply type the example sentences directly into the text file and put a semicolon (;) or number sign (#) before any comments (for yourself or others).

FOR MORE INFORMATION

For more help, explore the links on the LGM wiki page: http://moin.delph-in.net/MatrixTop.

6. CONCLUSION

The LGM encourages the implementation of computational grammars. Its customization questionnaire provides a relatively simple way to build a precision grammar for any language, including lesser resourced languages, as this case study with Lezgi demonstrates. However, getting started proves difficult because the questionnaire's supporting resources are not organized for independent learners. This might be solved by more prominent and better organized instructions for new users such as those outlined in this paper.

REFERENCES

Bender, Emily M. 2014. Language CoLLAGE: Grammatical Description with the LinGO Grammar Matrix. *Proceedings of the Ninth International Conference of Language Resources and Evaluation (LREC-2014)*, 2447–2451. http://www.lrec-conf.org/proceedings/lrec2014/pdf/639_Paper.pdf.

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

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. *Proceedings of the 2002 workshop on Grammar engineering and evaluation-Volume 15*, 1–7. Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=1118785 (19 October, 2016).

Goodman, Michael Wayne. 2013. Generation of Machine-Readable Morphological Rules from Human-Readable Input. (Ed.) Sanghoun Song & Joshua Crowgey. *UW Working Papers in*

Linguistics 30. http://depts.washington.edu/uwwpl/vol30/goodman_2013.pdf (23 September, 2016).

Haspelmath, Martin. 1993. *A grammar of Lezgian*. Berlin; New York: Mouton de Gruyter.

Manning, Christopher. 1994. *Ergativity: Argument Structure and Grammatical Relations*. Stanford PhD. http://www.press.uchicago.edu/ucp/books/book/distributed/E/bo3643756.html (30 June, 2014).

Simons, Gary F. & Charles D. Fennig (eds.). 2018. *Ethnologue: Languages of the World*. Twenty-first. Dallas, Texas: SIL International. http://www.ethnologue.com.