# What's in a Word? Refining the Morphotactic Infrastructure in the LinGO Grammar Matrix Customization System

Michael Wayne Goodman and Emily M. Bender
University of Washington
Seattle, WA, USA
{goodmami,ebender}@uw.edu

## 1 Introduction

The LinGO Grammar Matrix customization system (Bender et al., 2002, 2010) is a web-based software system for creating implemented HPSG (Pollard and Sag, 1994) grammar fragments on the basis of user input of typological and lexical information. These fragments are compatible with the LKB grammar development environment (Copestake, 2002) and other DELPH-IN software. The system consists of a core grammar (types and constraints hypothesized to be useful across all languages) and a series of "libraries" of analyses of recurring, but non-universal, phenomena. Most existing and future Matrix libraries involve morphological expression in at least some languages, so the customization system must provide a means for users to define lexical rules which attach the morphemes expressing various linguistic features.

Following O'Hara (2008), we conceptualize this as problem of **morphotactics**, separating the ordering and co-occurrence restrictions of morphemes from both their syntactic and phonological consequences. This approach puts the lexical rules and their relationships to each other as our prime concern.

In this paper, we present refinements to the Grammar Matrix's original morphotactic infrastructure (O'Hara, 2008), in order to better meet two constraints: (i) The system must be able to handle all types of morpheme co-occurrence restrictions found in the world's languages; and (ii) the grammars it produces must be human- as well as machine-readable, i.e., suitable for extension and maintenance by grammar engineers.

## 2 Background

The grammars derived from the customization system include analyses of morphotactics (morpheme ordering and interdependencies) and morphosyntax (the syntactic and semantic effects of morphemes). As is standard in LKB (Copestake, 2002) grammars, morphemes are modeled by non-branching rules which are distinguished from phrase structure rules in that they are segregated at the bottom of each parse tree: The daughters of phrase structure rules may be constituents licensed by other phrase structure rules, by lexical rules, or by lexical entries directly, but lexical rules can only take constituents licensed by lexical entries or other lexical rules as their input.

In using rules to model morphemes, this would appear on the surface to be an "a-morphous" approach (Anderson, 1992), but in fact, each rule has the effect of attaching one (possibly phonologically empty) affix, and the rules apply in a particular order, giving words internal structure, making this in fact a "morphemic" approach. The resulting system nonetheless respects lexical integrity (Bresnan and Mchombo, 1995), in that the inputs (daughters) to phrase structure rules are fully-formed words.

In Stump's (2001) terms, the approach is inferential and incremental. Stump argues for an approach that is instead realizational, on the basis of multiple exponence and zero realization. Multiple exponence is not problematic on our account because the syntactic (and semantic) contributions of morphemes are modeled via unification; if multiple morphemes contribute the same information, the constraints are simply unified. As for zero realization, we contend that the formal system must in some way account

for all constraints on the distribution of inflected forms. If there is no overt morpheme associated with some of the constraints, we contend that a lexical rule with no morphological effect (i.e., a zero-marked rule) is an appropriate mechanism for enforcing such constraints.

# 3 Room for Improvement

## 3.1 Current Implementation

In the current morphotactic infrastructure developed by (O'Hara, 2008), the core Matrix grammar (common to all grammars produced by the system) only provides a single boolean feature, INFLECTED, to discern between lexemes that may be used in phrase structure rules and those that must undergo further inflection.

In this system, "slots" are an abstract concept akin to morphological paradigms. A slot defines what lexical types (or other slots) it may take as input and what constraints it puts on other slots. Morphemes are defined within a slot, with the effect that all morphemes within a given slot will appear in the same position in a word, cannot co-occur, and are all either optional or obligatory. Unlike paradigms, it is not necessary for all morphemes to affect the same (syntactic or semantic) features, and morphemes that affect the same features may be in different slots if they apply to a different set of lexical types. The kinds of constraints a slot can place on other slots are defined in Table 1. Morphemes specify the syntactic features and values they affect, if any, and their phonological contribution, if any.

| | |
|---|---|
| A **forces** B | If slot A occurs, slot B must also occur (later) |
| A **requires** B | Slot A cannot occur unless slot B has already occurred |
| A **forbids** B | If slot A occurs, slot B must not occur (before or after) |

Table 1: Co-occurrence restrictions for O'Hara's morphotactic system.

Based on the information the user provides, the customization system will decide which basic rule types (as defined in the Matrix core) should be used in the definitions of the custom lexical rules. O'Hara's system also introduces the notion of a TRACK features, or **flags**, which keep track of the application of specific rules and are used for long-distance restrictions on slot co-occurrence. Further, the lexical rules created by this system will set INFLECTED to + only when the last necessary rule has applied (necessity being obligatory slots or slots forced by a previous rule).

## 3.2 Stem versus Word Distinction

While the simplicity of having one feature, INFLECTED, define whether a lex-item can be used in a phrase is appealing, it is the reason for one major drawback of O'Hara's system.

Figures 1–3 show three morphotactic systems, each (ideally) with 3 slots varying in the obligatoriness of slots and the co-occurrence restrictions they place on each other. Only the first system works as intended, and the other two result in inelegant redundancies (i.e. additional slots differing only in the value they place on INFLECTED), complicating the task of hand-editing lexical rules later.

## 3.3 Constraints on Lexical Types

Another context in which the current system leads to redundancy is when certain slots are optional for some lexical types but obligatory for others. In order to model this, one would need to provide duplicate slots that define the same morphemes, features, etc., but with one being optional and the other obligatory, and taking different inputs.

# 4 Proposed System

In light of the deficiencies of the current system, we propose changes to both the Matrix core as well as the customization logic.

## 4.1 Extending INFLECTED

The core of our new proposal is to change the value of INFLECTED from a single boolean flag to a complex AVM containing multiple ternary flags.[1] Each of these flags is defined in the language-specific grammar files. In essence, we are merging TRACK into INFLECTED.

Moving from a boolean value to a complex value for the feature INFLECTED also allows us to simplify the inventory of lexical rules types in the Matrix core. In particular, there is no longer any need to cross-classify these types

[1] Three leaf-types on a type hierarchy, so more than three values are possible. See Figure 5.

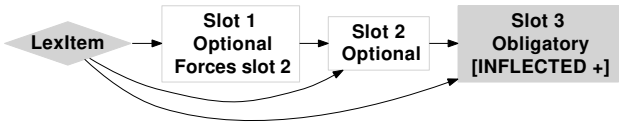


Figure 1: Forcing a slot before an obligatory slot.

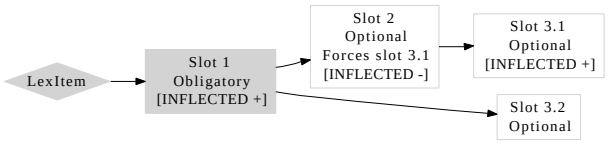


Figure 2: Forcing a slot after an obligatory slot

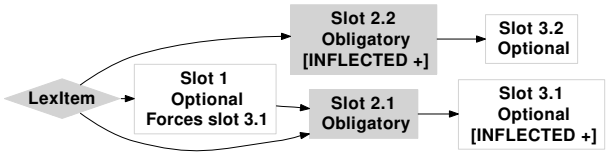


Figure 3: Forcing a slot around an obligatory slot

along the dimension of **lexeme-to-word-rule** vs. **lexeme-to-lexeme-rule**.[2]

The new system defines the value of INFLECTED in the Matrix core as in Figure 4. The flags that appear on these structures are then defined in the language-specific grammar files, and the set will be identical for each of `inflected`, `infl-initial`, and `infl-satisfied`. All lex-items start with the `infl-initial` configuration, and phrase structure rules, rather than checking for [INFLECTED +], now check for [INFLECTED *infl-satisfied*].

```
[ INFLECTED inflected ].
  inflected := avm.
  infl-initial := inflected.
  infl-satisfied := inflected.
```

Figure 4: The new definition of INFLECTED

The values of the flags come from the `luk` hierarchy in the Matrix core, as shown in Figure 5.[3] We use the following interpretation for the values: + means the lexical rule has been applied, – means the lexical rule has not applied and must apply before the lex-item can be used in a phrase, and `na` means the lexical rule has not applied but does not need to apply. Using these values, we can simplify, and extend, the logic for morphotactic slots.

[2]These might be more properly named stem-to-word-rule and stem-to-stem-rule.

[3]In using this generalization of the notion of boolean features we follow the English Resource Grammar (Flickinger, 2000).

We note that the optional versus obligatory distinction is equivalent to saying that any obligatory slot is just forced by the lexical types it takes as input. Therefore, by allowing users to specify co-occurrence restrictions on lexical types and by removing the dimension of slot optionality, we simplify the system and create additional functionality by allowing lexical types to individually specify the slots they require.

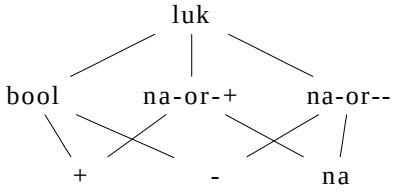


Figure 5: The `luk` hierarchy.

### 4.2 Co-occurrence Restrictions

We can now redefine how the three co-occurrence restrictions are implemented. First, we set `infl-initial` so all flags are `na`, and all flags in `infl-satisfied` are `na-or-+`. Rules that don't explicitly change the value of a flag are instead set to copy it up from daughter to mother. The logic of the co-occurrence restrictions is shown in Table 4.2.

### 4.3 Disjunctive Slots

The new flag system allows us to easily make disjunctive requirements; a lex-type can specify that either slot A or slot B is necessary before the lex-type can be a word.

| Restriction | Rule Definition |
|---|---|
| A **forces** B | A-rule: `[INFLECTED.B -]` |
| | B-rule: `[INFLECTED.B +]` |
| B **requires** A | A-rule: `[INFLECTED.A +]` |
| | B-rule: `[DTR.INFLECTED.A +]` |
| A **forbids** B | A-rule: `[INFLECTED.A +]` |
| | B-rule: `[DTR.INFLECTED.A na]` |

We do this by creating one flag for both slots, and when one occurs the flag is satisfied, and nothing changes if the other rule also occurs. If neither of the rules occur, the flag is left unsatisfied, and the lex-item's INFLECTED value will not unify with `infl-satisfied`.

We can create disjunctive requirements in two ways: explicitly or implicitly. An explicit disjunction is specified by the user and allows for alternations in sequential slots, while an implicit disjunction is calculated by the system based on the slots' structure. Since explicit disjunctions are specified on single slots, a pattern such as "(A and B) or (C and D)" cannot be modeled with an explicit disjunction, but it can be modeled with an implicit disjunction. Figure 6 illustrates this case.

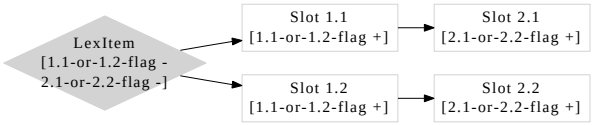


Figure 6: An implicit disjunction

We have designed the system to create the minimum number of flags to model these kinds of systems. While we could have a series of three flags, such as {[1.1-or-1.2], [1.1-or-2.2], [1.2-or-2.1]}, we only create the minimal set {[1.1-or-1.2], [2.1-or-2.1]}.[4]

## 5 Test Cases

While possibly all of the morphotactic systems we want to create could have been done under the old system, the result would sometimes be a complex set of rules with many duplicated features. The new system promises to solve these problems in a much more straightforward way, leading to greater elegance in analyses and less of a burden on the grammar developer.

[4]Here, the flag names show the slots they are satisfied by. Thus, [1.1-or-1.2] would be satisfied if either slot 1.1 or slot 1.2 occur.

### 5.1 French Pronominal Affixes

In French [fra], strict transitive verbs such as *prendre* ('take') require either object clitic (analyzed as a prefix, following Miller and Sag (1997)) or a full NP object, while optionally transitive verbs such as *manger* ('eat') can take the clitic, a full NP object, or no object at all:[5]

(1) Je mange (le biscuit)/Je le-mange
I eat.1SG (the cookie)/I 3SG.M-eat1SG
'I eat the cookie/it/$\phi$.' [fra]

(2) Je prends *(le biscuit)/Je le-prends
I take.1SG *(the cookie)/I 3SG.M-eat1SG
'I take the cookie/it/*$\phi$.' [fra]

The argument optionality library in the Grammar Matrix (Saleem, forthcoming) handles such alternations via lexical rules. For a verb like *prendre*, there is an obligatory slot which houses the object prefix rules as well as a zero-marked rule which constrains the object to be incompatible with the object-drop phrase structure rule. A verb like *manger*, however, should not go through this rule. Rather, it should simply optionally take the object prefixes. On the old system, this required duplicating slots. On the new system, however, it can be elegantly handled by placing a **forbids** restriction on the lexical type for *manger* so it cannot take the zero-marked rule and a **requires** restriction on *prendre* so that it must either take an object prefix or the zero-marked rule.

### 5.2 Lushootseed Tense and Aspect Markers

Lushootseed [lut] is an agglutinative Coast Salish language. It has morphemes for tense and for aspect, with the requirement that at least one of them (tense or aspect) must occur, or both may occur (Hess, 1967). (3) is an example of a sentence with both tense and aspect specified. With the old system, this disjunctive relationship would require complex arrangements, and much duplication of slots, but with the new system we can model it more elegantly.

(3) Lulexwil ti cacas
Lu-le-xwil t-i cacas
FUT-PRG-get.lost det.DEF-DIR child
'The child will become lost.' [lut]

[5]We assume for the sake of the example that there is only one lexical entry for *manger*.

We would first create the slots and set their inputs as needed to capture the affix ordering correctly, then we would add a disjunctive **forces** restriction on all lexical types that require a tense or aspect marker, with the forcees being the tense and aspect slots. The system will create one flag for these two slots, and when one slot occurs the flag is satisfied. If both occur, the value of the flag is still satisfied, but if neither occur it will be left unsatisfied, disallowing it from being used in phrasal rules. Note that the Lushootseed case differs from that depicted in Figure 6, in that the disjunctively **forced** slots are on the same inflection path, allowing them to co-occur.

## 6 Conclusion and Future Work

We have proposed a new morphotactic system that allows for simple analyses of a wider range of patterns than the previous system. The primary innovation in our proposal is in generalizing the distinction between stems and words from a simple binary one to a more nuanced notion of words as inflectionally satisfied. The HPSG notion of types allows us to make this generalization while maintaining a simple constraint on the daughters of phrase structure rules, namely, that they be [INFLECTED *infl-satisfied*].

The benefit of the proposed system is immediately obvious for a context like the LinGO Grammar Matrix customization system, where robustness across many typologically diverse languages is necessary, but it may also prove useful for individual hand-built grammars—particularly agglutinating or polysynthetic languages where there may be complex dependencies among the morphemes.

In future work, we would like to consider techniques for simplifying the rule sets output by the system with the goal of further enhancing their maintainability when grammar engineers extend these grammars. We plan to explore graph-based methods for simplifying the morphotactic system. For instance, we would like to find and remove unused edges (i.e. inputs to rules), or use a cleverly restricted set of edges rather than an overabundance of flags.

## References


Stephen R. Anderson. 1992. *A-morphous Morphology*. Cambridge University Press.

Emily M. Bender, Scott Drellishak, Antske Fokkens, Michael Wayne Goodman, Daniel P. Mills, Laurie Poulson, and Safiyyah Saleem. 2010. Grammar prototyping and testing with the LinGO Grammar Matrix customization system. In *Proceedings of ACL 2010 Software Demonstrations*.

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 *Proceedings of the Workshop on Grammar Engineering and Evaluation at COLING 2002*, pages 8–14.

Joan Bresnan and Sam A. Mchombo. 1995. The lexical integrity principle: Evidence from Bantu. *Natural Language & Linguistic Theory*, 13(2):181–254.

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

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

T.M. Hess. 1967. *Snohomish grammatical structure.* Ph.D. thesis, University of Washington.

Philip H. Miller and Ivan A. Sag. 1997. French clitic movment without clitics or movement. *Natural Language and Linguistic Theory*, 15:573–639.

Kelly O'Hara. 2008. *A Morphotactic Infrastructure for a Grammar Customization System*. Master's thesis, University of Washington.

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

Safiyyah Saleem. forthcoming. *Argument Optionality: A New Library for the Grammar Matrix Customization System*. Master's thesis, University of Washington.

Gregory T. Stump. 2001. *Inflectional morphology: A theory of paradigm structure*. Cambridge Univ Press.