Rapid Prototyping of Scalable Grammars: Towards Modularity in Extensions to a Language-Independent Core

Emily M. Bender
Department of Linguistics
University of Washington
Box 354340
Seattle WA 98195-4340 USA
ebender@u.washington.edu

Dan Flickinger
Center for the Study of Language and Information
Stanford University
Stanford CA 94305-2150 USA
danf@csli.stanford.edu

Abstract

We present a new way to simplify the construction of precise broad-coverage grammars, employing typologically-motivated, customizable extensions to a language-independent core grammar. Each 'module' represents a salient dimension of cross-linguistic variation, and presents the grammar developer with simple choices that result in automatically generated language-specific software. We illustrate the approach for several phenomena and explore the interdependence of the modules.

1 Introduction

Manual development of precise broad-coverage grammar implementations, useful in a range of natural language processing/understanding tasks, is a labor-intensive undertaking, requiring many years of work by highly trained linguists. Many recent efforts toward reducing the time and level of expertise needed to produce a new grammar have focused on adapting an existing grammar of another language (Butt et al., 2002; Kim et al., 2003; Bateman et al., ip). Our work on the 'Grammar Matrix' has pursued an alternative approach, identifying a set of language-independent grammar constraints to which language-specific constraints can be added (Bender et al., 2002). This approach has the hitherto unexploited potential to benefit from the substantial theoretical work on language typology. In this paper, we present a prototype Grammar Matrix customization system. This system draws on phenomenon-specific modules encoding dimensions of linguistic variation, presents the grammar developer with simple choices for each phenomenon, and then automatically generates a working starter-grammar, incorporating both the cross-linguistic Matrix core and language-specific constraints. The prototype addresses basic word order, sentential negation, yes-no questions, and a small range of lexical entries.

2 The Grammar Matrix

Wide-coverage grammars representing deep linguistic analysis exist in several frameworks, including Head-Driven Phrase Structure Grammar (HPSG), Lexical-Functional Grammar, and Lexicalized Tree Adjoining Grammar. In HPSG (P. and Sag, 1994), the most extensive grammars are those of English (Flickinger, 2000), German (Hinrichs et al., 1997; Müller and Kasper, 2000; Crysmann, ip), and Japanese (Siegel, 2000; Siegel and Bender, 2002). The Grammar Matrix is an attempt to distill the wisdom of existing grammars and document it in a form that can be used as the basis for new grammars. The main goals of the project are: (i) to develop in detail semantic representations and the syntax-semantics interface, consistent with other work in HPSG; (ii) to represent generalizations across linguistic objects and across languages; and (iii) to allow for very quick start-up as the Matrix is applied to new languages.

The original Grammar Matrix consisted of types defining the basic feature geometry, types associated with Minimal Recursion Semantics (e.g., (Copestake et al., 2001)), types for lex-

ical and syntactic rules, and configuration files for the LKB grammar development environment (Copestake, 2002) and the PET system (Callmeier, 2000). Subsequent releases have refined the original types and developed a lexical hierarchy. The constraints in this 'core' Matrix are intended to be language-independent and monotonically extensible in any given grammar. With the typology-based modules presented here, we extend the constraint definitions which can be supplied to grammar developers to those that capture generalizations holding only for subsets of languages.

3 Typology-based modules

In general, we find two kinds of typological variation across languages. On the one hand, there are systems (formal or functional) which must be represented in every language. For example, every language has some set of permissible word orders (formal) and a means of expressing sentential negation (functional). On the other hand, there are linguistic phenomena which appear in only some languages, and are not typically conceptualized as alternative realizations of some universal function, phenomena such as noun incorporation, numeral classifiers, and auxiliary verbs. Each of these phenomena are found in recurring varieties that can be subjected to typological analysis (see, e.g., (Mithun, 1984)). Our approach is designed to handle both kinds of typological variation.

As with earlier versions of the Matrix, we aim to support rapid prototyping of precision grammars that can scale up to broad-coverage (as have the NorSource (Hellan and Haugereid, 2003) and Modern Greek (Kordoni and Neu, 2003) grammars, based on early versions of the Matrix). This sets a high bar for the modules themselves, requiring them to be good early approximations which may need to be refined but not thrown out. It also requires that the automatically generated grammar files maintain a high degree of readability so that they may be effectively modified. In future work, we intend to extend the system to allow the linguist to revise decisions in the face of new information or improved linguistic analyses.

The core Matrix and modular extensions to it may appear analogous to the Principles and Parameters proposed by Chomsky (1981) and others. However, whereas Parameters are meant to be abstract 'switches' which simultaneously control multiple different, apparently unrelated phenomena, the modules in the Matrix each encode the constraints necessary to handle one particular phenomenon. Nonetheless, this does not make the modules trivial: they need to be carefully designed in order to be mutually consistent, ideally across all possible combinations. Our strategy is thus consistent with a bottom-up, data-driven investigation of linguistic universals and constraints on cross-linguistic variation. As the number and breadth of implemented grammars grows, we expect linguistic predictions to emerge and become part of improved modules, particularly with respect to interactions among the distinct phenomena covered. Our approach should in time be instrumental in assisting large-scale typological investigations (covering hundreds of languages), making use of the linguistically precise constraints encoded in these modules to uncover deeper and more subtle facts about languages.

4 Implementations of prototype system

We have implemented a prototype system with a small set of modules targeting basic word order, main-clause yes-no questions, and sentential negation.[1] The corresponding choices and a questionnaire for creating a small lexicon are presented to the user through an html form interface. A perl/cgi back-end produces a starter grammar from the user input and an internal knowledge base. The resulting grammars can be used immediately to parse and generate a fragment of the target language. The system can be accessed at http://www.delph-in.net/matrix/modules.html. This section describes its linguistic coverage.

4.1 Word order

The word order module addresses the so-called basic word order in a language: the relative order of subjects, verbs, and verbal complements. Languages vary in their rigidity in this respect, and the question of how to determine the basic word-order of a language is notoriously complex. Nonetheless, we believe that most linguists working on linguistic description analyze some orders as primary and others as derived. Thus the word

[1] Drellishak and Bender (ta) present a module for coordination which is integrated with those described here.

order module is meant to capture the relative ordering of major constituents in clauses without word-order changing phenomena such as topicalization, extraposition, subject-verb inversion, etc. Modules for such phenomena will need to interact appropriately with the basic word-order module.

The Matrix core grammar provides definitions of basic head-complement and head-subject schemata which are consistent with our implementation of compositional semantics (Flickinger and Bender, 2003), as well as definitions of head-initial and head-final phrase types. The word order module creates subtypes joining the head-complement and head-subject schemata with the types specifying head/dependent order, creates instances of those types as required by the LKB parser, and constrains the rules to eliminate spurious ambiguity in the case of free word order. It currently handles SOV, SVO, VSO, VOS, OVS, OSV, V-final, V-initial, and free word order. We leave to future work variations such as V2 order, differing word order in main v. subordinate clauses, and flexible ordering among complements in otherwise strict word order languages.

4.2 Yes-no questions

For yes-no questions, we implement four alternatives: inversion of the subject and a main or auxiliary verb relative to declarative word order and sentence-initial or final question particles.

Inversion of the subject and the main verb is implemented with a lexical rule which relocates the subject (the value of SUBJ in the valence specifications) to be the first on the COMPS list, and further assigns a positive value for an additional feature INV (inverted) on verbs. This feature may well have independent syntactic motivation in the language, but is in any case used here so the declarative/interrogative distinction can be made in the semantics once the clause is constructed. Subject-aux inversion is a minor extension of the basic inversion type, constraining the lexical rule to only apply to auxiliary verbs. This module handles 'support' verbs like *do* in English in not licensing inversion with main verbs, while licensing similar strings with a semantically empty support verb (if it is in the lexicon). The third type of mechanism employs a distinct question particle, here treated as a pre- or post-modifying sentence adverb. The grammar developer is prompted for this positional distinction, and for the spelling of the particle; the code for the relevant lexical entry is then autogenerated, instantiating a question particle type which supplies the remaining syntactic and semantic constraints needed.

Future work on this module includes support for 'intonation questions', where the same string can be associated with either proposition or question semantics, as well as the integration of declarative/interrogative punctuation contrasts.

4.3 Sentential negation

The sentential negation module handles two general negation strategies, several variants on each, and allows for both to coexist in a single grammar.

The first strategy is negation via verbal inflection. For this strategy, the grammar developer specifies whether the inflection attaches to main verbs, auxiliaries, or either; whether it is a prefix or a suffix; and the form of the affix. We currently only allow for strictly concatenative morphology. In a more fully developed system, the syntax-semantics modules here would be interfaced with a separate means of specifying morphophonology (cf. (Bender and Good, ip)).

The second strategy is negation via a negative adverb, with two sub-cases: The negative adverb may be an independent modifier (of V, VP, or S and pre- or post-head) or it may be a selected complement of the verb (main verbs only, auxiliaries only, or both) (Kim, 2000). The grammar developer specifies the form of the adverb.

Neither, either or both of these strategies may be selected. If neither, the grammar produced will not contain an analysis of negation. If both, the grammar developer must specify how the strategies interact, from among five choices: (i) the two strategies are in complementary distribution, (ii) the two strategies can appear independently or together, (iii) both inflection and an adverb are required to express sentential negation, (iv) the adverb is obligatory, but it may appear with or without the inflection, and (v) the inflection is obligatory, but it may appear with or without the adverb.

In the generated grammars, independent adverbs are implemented by adding appropriate lexical types and lexical entries. Selected adverbs and inflection are handled via lexical rules similar

to those presented in (Sag et al., 2003). For example, in a language where sentential negation can be expressed by inflection alone or inflection in combination with a (selected) adverb, we generate two lexical rules. One changes the form of the verb and adds the negative semantics. The other changes the form of the verb and adds the negative adverb to its complements list.

4.4 Lexicon

As HPSG is a strongly lexicalist theory, words tend to carry quite a bit of information. This information is encoded in lexical types; lexical entries merely specify the type they instantiate, their orthographic form, and their semantic predicate. Many of the constraints required (e.g., for the linking of syntactic to semantic arguments) are already provided by the core Matrix. However, there is also cross-linguistic variation.

We ask the grammar developer to specify two nouns and two verbs (one transitive and one intransitive), as well as an auxiliary, two determiners, two case-marking adpositions, a negative adverb and a question particle, if appropriate. Nouns are specified as to whether they require, allow, or disallow determiners. Verbs are specified as to whether each argument is expressed as an NP or a PP, and optionally for an additional (non-finite) form. Auxiliaries are specified as to whether they introduce independent predicates or only carry tense/aspect; take S, VP or V complements; appear to the left, right or either side of their complements; and take NP or PP subjects. Case-marking adpositions must be specified as either prepositions or postpositions. Finally, the questionnaire requires orthographic forms and predicate names. Note that the forms are assumed to be fully inflected (modulo negation), support morphological processes awaiting future work.

We use this information and the knowledge base to produce a set of lexical types inheriting from the types defined in the core Matrix and specifying appropriate language-specific constraints, and a set of lexical entries.

5 Limits of modularity

Recent computational work in HPSG has asked whether different parts of a single grammar can be abstracted into separate, independent modules, either for processing (Kasper and Krieger, 1996; Theofilidis et al., 1997) or grammar development (Kešelj, 2001). Our work is most similar to Kešelj's though we are pursuing different goals: Kešelj is looking to support a division of labor among multiple individuals working on the same grammar and to support variants of a single grammar for different domains. His modules each have private and public features and types, and he illustrates the approach with a small-scale question answering system. In contrast, we are approaching this issue from the perspective of reuse of grammar code in the context of multilingual grammar engineering (a possibility suggested, but not developed, by Theofilidis et al).

Our notion of modularity is influenced by the following constraints: (i) The questions in the customization interface must be sensible to the working linguist; (ii) The resulting starter grammars must be highly readable so that they can be extended by the grammar developer (typically only one per grammar); and (iii) HPSG practice values capturing linguistic generalizations by having single types encode many different constraints and, ideally, single constraints contribute to the analysis of many different phenomena.

Even with the modest linguistic coverage of the existing system, we have found many cases of non-trivial interaction between the modules: Our phrase structure rules, following HPSG practice, capture cross-categorial generalizations: if both verbs and adpositions follow their complements, then a single complement-head rule serves for both. However, few languages (if any) are completely consistent in their ordering of heads and dependents. Thus, before defining the types and instances for these rules, we must determine whether the fragment requires auxiliaries (for negation or yes-no questions) or case-marking adpositions, and whether their order with respect to their complements is consistent with that of main verbs. A second example is the lexical type for main verbs, whose definition depends on whether the language has auxiliaries (requiring a feature AUX distinguishing the two kinds of verbs and a feature FORM governing the distribution of finite and non-finite verbs). As a third example, the negation and question modules each have options requiring auxiliaries, but we must posit the asso-

ciated types and constraints at most once.

Thus we find that, for our purposes, the relevant notion of modularity is modularity from the point of view of the linguist who uses the system to create a starter grammar. To support this, we strive to make the questions we ask of the linguist be as independent of each other as possible, and to make it clear when one particular choice (e.g., negation as inflection) requires further information (suffix v. prefix). The fact that the questions we present to the linguist don't correspond to neatly isolated parts of the underlying knowledge base is not a failure of the approach, but rather a reflection of the complexity of language. The very interconnectedness of grammatical phenomena is at the heart of research in theoretical syntax. We intend our system to provide a data-driven cross-linguistic exploration of that interconnection.

6 Validation of prototype system

To verify the mutual consistency of the modules developed so far and to illustrate their applicability to a interesting range of languages, we developed abstract test suites for seven languages. This convenience sample of languages is not representative, either typologically or genetically. The grammatical and ungrammatical examples in each test suite use a small, artificial lexicon, and reflect the typological properties of each language along the dimensions of basic word order, sentential negation, and yes-no questions (Table 1). Table 2 presents the performance of each grammar (as generated by our prototype system with appropriate input) on its associated test suite.

Language[2]	Order	Negation	Yes-no Q[3]
English	SVO	aux-selected adv	aux inv
Hindi	SOV	pre-V adv	S-init part.
Japanese	V-fi nal	verbal suffi x	S-fi nal part
Mandarin	SVO	pre-V adv	S-fi nal part, A-not-A
Polish	free	pre-V adv	S-init part
Slave	SOV	post-V adv	S-init part
Spanish	SVO	pre-V adv	main V inv

Table 1: Languages used in testing

While these test suites are quite modest, we believe they show that the prototype system is able to produce good first-pass grammar fragments for an interesting variety of languages. More study is needed to develop a means of testing the cross-compatibility of all choices on all modules, to evaluate the coverage against a typologically justified sample, and to gauge the success of this strategy in producing grammars which are comprehensible to beginning grammar developers.

Language	Pos.	Coverage	Neg.	Overgen.
English	5	100%	10	10%
Hindi	5	100%	10	0%
Japanese	6	100%	10	0%
Mandarin	4	75%	9	0%
Polish	14	100%	8	0%
Slave	3	100%	6	0%
Spanish	5	100%	7	0%

Table 2: Parsing evaluation results

7 Conclusion and outlook

We have described a method for extending a language-independent core grammar like the Grammar Matrix with modules handling cross-linguistically variable but still recurring patterns. This method allows for extremely rapid prototyping of deep precision grammars in such a way that the prototypes themselves can serve as the basis for sustained development. We envision at least four potential uses for this kind of grammar prototyping: (i) in pedagogical contexts, where it would allow grammar engineering students to more quickly work on cutting-edge problems, (ii) in language documentation, where a documentary linguist in the field might be collaborating remotely with a grammar engineer to propose and test hypotheses, (iii) in leveraging the results from economically powerful languages to reduce the cost of creating resources for minority languages, and (iv) in supporting typological or comparative studies of linguistic phenomena or interactions between phenomena across languages.

Acknowledgments

We thank Scott Drellishak, Stephan Oepen, Laurie Poulson, and the 2004 and 2005 multilingual grammar engineering classes at the University of Washington for valuable input and NTT Communication Science Laboratories for their support through a grant to CSLI (Stanford). All remaining errors are our own.

[2]Sources: Hindi: Snell and Weightman, 2000, Mandarin: Li and Thompson, 1981, Polish: Adam Przepiórkowski, p.c., Slave (Athabaskan): Rice, 1989

[3]In addition to intonation questions, if any.

References

J.A. Bateman, I. Kruijff-Korbayová, and G.-J. Kruijff. ip. Multilingual resource sharing across both related and unrelated languages: an implemented, open-source framework for practical natural language generation. *Res. on Lang. and Computation*.

E.M. Bender and J. Good. ip. Implementation for discovery: A bipartite lexicon to support morphological and syntactic analysis. In *CLS 41*.

E.M. Bender, D. Flickinger, and S. Oepen. 2002. The grammar matrix. *COLING 2002 Workshop on Grammar Engineering and Evaluation*.

M. Butt, H. Dyvik, T.H. King, H. Masuichi, and C. Rohrer. 2002. The parallel grammar project. In *COLING 2002 Workshop on Grammar Engineering and Evaluation*.

U. Callmeier. 2000. PET — A platform for experimentation with effi cient HPSG processing techniques. *Natural Lang. Engineering*, 6 (1):99 – 108.

N. Chomsky. 1981. *Lectures on Government and Binding*. Foris, Dordrecht.

A. Copestake, A. Lascarides, and D. Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *ACL 2001*.

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

B. Crysmann. ip. Relative clause extraposition in german: An effi cient and portable implementation. *Research on Lang. and Computation*.

S. Drellishak and E.M. Bender. ta. Coordination modules for a crosslinguistic grammar resource. In *Proc. of HPSG 2005*.

D. Flickinger and E.M. Bender. 2003. Compositional semantics in a multilingual grammar resource. In *Proc. of the Workshop on Ideas and Strategies for Multilingual Grammar Development, ESSLLI 2003*, pages 33–42.

D. Flickinger. 2000. On building a more effi cient grammar by exploiting types. *Natural Lang. Engineering*, 6 (1):15 – 28.

L. Hellan and P. Haugereid. 2003. Norsource: An exercise in matrix grammar-building design. In *Proc. of the Workshop on Ideas and Strategies for Multilingual Grammar Development, ESSLLI 2003*, pages 41–48.

W.D. Hinrichs, E.and Meurers, F. Richter, M. Sailer, and H. Winhart. 1997. Ein HPSG-Fragement des Deutschen. Arbeitspapiere des Sonderforschungsbereichs 340, Bericht Nr. 95.

W. Kasper and H.-U. Krieger. 1996. Modularizing codescriptive grammars for effi cient parsing. In *COLING 1996*, pages 628–633.

V. Kešelj. 2001. Modular HPSG. Technical Report CS-2001-05, Department of Computer Science, University of Waterloo, Waterloo, Ont., Canada.

R. Kim, M. Dalrymple, R.M. Kaplan, T.H. King, H. Masuichi, and T. Ohkuma. 2003. Multlingual grammar development via grammar porting. In *Proc. of the Workshop on Ideas and Strategies for Multilingual Grammar Development, ESSLLI 2003*, pages 49–56.

J. Kim. 2000. *The Grammar of Negation: A Constraint-Based Approach*. CSLI, Stanford, CA.

V. Kordoni and J. Neu. 2003. Deep gramamr development for Modern Greek. In *Proc. of the Workshop on Ideas and Strategies for Multilingual Grammar Development, ESSLLI 2003*, pages 65–72.

C.N. Li and S.A. Thompson. 1981. *Mandarin Chinese: A Functional Reference Grammar*. University of California Press, Berkeley, CA.

M. Mithun. 1984. The evolution of noun incorporation. *Language*, 60(4):847–894.

S. Müller and W. Kasper. 2000. HPSG analysis of German. In W. Wahlster, editor, *Verbmobil. Foundations of Speech-to-Speech Translation*, pages 238 – 253. Springer, Berlin, Germany.

Carl P. and I.A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The Univeristy of Chicago Press, Chicago, IL.

K. Rice. 1989. *A Grammar of Slave*. Mouton de Gruyter, Berlin.

I.A. Sag, T. Wasow, and E.M. Bender. 2003. *Synactic Theory: A Formal Introduction*. CSLI, Stanford, CA, 2nd edition.

M. Siegel and E.M. Bender. 2002. Effi cient deep processing of Japanese. In *Proc. of the 3rd Workshop on Asian Language Resources and International Standardization at COLING 2002*.

M. Siegel. 2000. HPSG analysis of Japanese. In W. Wahlster, editor, *Verbmobil. Foundations of Speech-to-Speech Translation*, pages 265 – 280. Springer, Berlin, Germany.

R. Snell and S. Weightman. 2000. *Hindi*. Teach Yourself Books.

A. Theofi lidis, P. Schmidt, and T. Declerck. 1997. Grammar modularization for effi cient processing: Language engineering devices and their instantiations. In *Proc. of the DGFS/CL*.