Grammar Engineering for Linguistic Hypothesis Testing

Emily M. Bender

To appear: Proceedings of TLSX

1 Introduction

In this paper, I argue that the tools and techniques of grammar engineering provide a means to take the development and evaluation of syntactic hypothesis testing to a new level. Grammar engineering is the process of creating machine-readable implementations of formal grammars. Traditionally, linguistic hypotheses are encoded as statements within a grammatical theory and tested by collecting relevant examples and manually verifying that the grammars correctly predict the grammaticality and linguistic structure of those examples. Computerized implementations of their grammars allow linguists to more efficiently and effectively test hypotheses, for two reasons: First, languages are made up of many subsystems with complex interactions. Linguists generally focus on just one subsystem at a time, yet the predictions of any particular analysis cannot be calculated independently of the interacting subsystems. With implemented grammars, the computer can track the effects of all aspects of the implementation while the linguist focuses on developing just one. Second, automated application of grammars to test suites and naturally occurring data allows for much more thorough testing of linguistic analyses—against thousands as opposed to tens of examples and including examples not anticipated by the linguist.

This work is situated within the Montagovian tradition of the "method of fragments" (Montague, 1974, Partee, 1979, Gazdar et al., 1985). In this methodology, theoretical ideas are validated (and extended) through the development of explicit grammars which can relate strings from some fragment of the language under study to semantic representations. This is useful because, as Partee puts it, "otherwise it is extremely easy to think that you have a solution to a problem when in fact you don't." (1979:95) In the remainder of this section, I provide some further background on why this is the case, working from three observations.

> **Observation 1:** There is no ultimate answer to the ultimate question of syntax, phonology and morphology.[1]

Despite this, much work in theoretical and computational linguistics seems focused on finding simple, elegant answers which, even if they don't explain the whole language, nonetheless have far-reaching consequences. This search is not entirely without fruit: Important

[1] Not even 42. For discussion, see Adams, 1979.

big-picture results in computational linguistics can change the way we approach certain tasks (e.g., the application of the noisy-channel model to machine translation (Brown et al., 1993)). In theoretical linguistics, analyses (or reanalyses) of particular phenomena can motivate the development of new formal theories (Bresnan, 1978, Gazdar, 1981). And yet, it is a mistake to focus only on such big picture results. Practical working systems as well as thorough scientific analysis both require attention to the details of language. The strong performance of NLP systems typically depends on language-specific parameter-tuning and/or rich linguistic annotation of training data, though these aspects are usually de-emphasized in presentations of such work. At the same time, there exists no formal grammar that approaches comprehensive description of a language.[2] The field of documentary and descriptive linguistics provides an exception to the general trend, reflecting an interest in whole-picture descriptions of languages, including as many details of as many subsystems as possible.

> **Observations 2:** "...que chaque langage forme un système où tout se tient"
> (...that each language forms a system where everything hangs together)[3]

For the structuralists, this dictum was meant to indicate that the structure of language—the object of study of grammarians—is all in the contrasts. From the point of view of generative grammar, this observation can be taken to mean that all of the parts of the system (phonology, morphology, syntax, semantics, etc.) and all of the subparts (e.g., case, agreement, word-order) have to work together in the production and interpretation of every utterance. As as consequence, it is not possible to test a syntactic hypothesis in one subdomain without simultaneously building a model of many intersecting subdomains. Furthermore, it is not possible to test a syntactic hypothesis without considering a wide variety of sufficiently complex sentences, to illustrate the interaction of subdomains.

> **Observation 3:** Explanatory adequacy depends on descriptive adequacy.

A grammar is descriptively adequate if it accepts (and assigns the correct structure to) the right set of strings (Chomsky, 1965). However, the same set of strings (or string-meaning pairs) can be described by many different grammars.[4] Thus, according to Chomsky's 1965 definitions of the terms, to have 'explanatory adequacy' a theory of grammar must provide a way to choose among grammars describing the same string set only on the basis of the data that a human language learner would be exposed to.

Much work in theoretical linguistics has thus focused on trying to achieve explanatory adequacy, unfortunately at the expense of the logically prior descriptive adequacy. No matter how interesting or elegant the analysis, it remains a conjecture unless its descriptive adequacy can be verified. At a minimum, one would wish to verify that the analysis correctly accounts

[2]Those that come closest are without exception computer implementations, for reasons that should become clear below.

[3]The text here is taken from Meillet 1903, but the original observation is variously attributed to Meillet, de Saussure, and von der Gabelentz; translation mine.

[4]This is believed to be true in principle, and seems likely given current evidence. However, as noted above, there are no complete formal grammars for any language, so it hasn't been shown in practice.

for the data that motivated it. More broadly, analyses should be verified in the contexts of larger test suites which illustrate their interactions with intersecting phenomena. In the early days of generative grammar, such verification was prohibitively expensive.[5] With modern computing and computational linguistic technology, however, it is within reach.

The remainder of this paper is structured as follows: §2 provides some background on hypothesis testing in the field of syntax. §3 gives some general information about grammar engineering. §4 presents the LinGO Grammar Matrix and describes how it can reduce the overhead costs of developing grammars. §5 presents the Montage vision of how grammar engineering can contribute to the documentation of under-described languages.

2 Syntactic Hypothesis Testing

2.1 Preliminaries

For the purposes of this paper, I take *syntax* to be the means by which natural languages relate strings of words to their meanings, over an unbounded set of strings of words (cf. Montague, 1974; Gazdar et al., 1985).[6] Secondarily, syntax is a system which models a kind of well-formedness. From a (human or computer) processing perspective, modeling well-formedness is potentially useful because it constrains the set of possible structures assigned to sentences, which in turn reduces the problem of ambiguity resolution. In contrast to much theoretical work on syntax, I take the string-meaning mapping to be primary and the modeling of well-formedness to be secondary.

A *syntactic hypothesis* is a hypothesis about the structures assigned to a class of sentences or more broadly about constraints on possible grammars.

2.2 Some examples

In this section, I illustrate the notion of a syntactic hypothesis by means of a number of examples. I will concentrate here mostly on the very broad-brush kind of hypotheses which can only be validated in principle by building very large models, in some cases, models of many languages.

Principles and Parameters The Principles and Parameters conceptualization of Universal Grammar (Chomsky, 1981) holds that particular grammars are composed of universal principles and settings for particular parameters. Each parameter has a small number of possible values, and controls a range of (often apparently unrelated) surface phenomena. This approach has inspired a lot of work exploring connections between different phenomena and

[5] Though some did try: cf. Friedman et al., 1971.

[6] In fact, there's more to linguistic form than strings of words (conceived of as orthography or phonology), and the form side of this equation should include such things as intonation. For present purposes, however, I will continue to refer to the linguistic form of an utterance as a string of words.

similarities across languages. To validate the overall approach, however, would require building a system which could create grammars for (substantial fragments of) various languages on the basis of parametric descriptions of those languages.

Semantic Compositionality Another example is semantic compositionality, or the hypothesis that the meaning of an expression is computable from the meaning of its parts (constituent morphemes) and how they are put together. This hypothesis (or alternatively, principle) has appeared in many different guises (see e.g., Montague (1970), Szabó (2005), Partee (1984); for its manifestation within the Grammar Matrix, see Flickinger and Bender (2003)). Semantic compositionality has often been supported by an appeal to the argument from productivity (as far back as Frege, 1914): if it didn't hold, how could we manage to produce and understand novel utterances? Szabó (2005)

As argued by Reddy (1993), however, it is misleading to see natural language utterances as a conduit through which speakers convey their precise intended meanings to hearers. Rather, natural language utterances serve as clues to speakers' intentions, which hearers interpret together with whatever other information is at their disposal. Thus for any given utterance that we understand, novel or not, we are using much more than the words and the way they are put together. I do not wish to deny that there is something systematic about how the meaning of an expression is created from the meaning of its parts. My purpose here is to point out that semantic compositionality falls into the class of ideas which can be fruitfully explored through the method of fragments: There is much that can be learned about which aspects of meaning are handled compositionally and how by attempting to model the relationship between strings and meanings (or meaning representations) for expanding fragments of natural languages.

Natural Languages as Context-Free Languages An example of a linguistic hypothesis which has been disproven through careful investigation is the claim that natural languages (as sets of strings) belong to the class of context-free languages. Early arguments against this hypothesis were shown to be spurious by Pullum and Gazdar (1982). Gazdar et al. (1985) then went on to try to develop a grammatical formalism (illustrated through a grammar for a fragment of English) which was context-free in its generative capacity yet allowed for the expression of linguistic generalizations (in contrast to naive applications of context-free grammars to natural language). Shieber (1986) subsequently was able to prove the non-context-freeness of Swiss German (and by extension, the class of natural languages). Nonetheless, the grammatical theory worked out in Gazdar et al. (1985) remained useful as a point of departure for the development of additional precise frameworks (namely, HPSG in its various versions). A single counterexample was enough to show that natural languages (in general) are not context free. It would not have been possible in principle to show conclusively that they are (as a counterexample might always be lurking). However, the hypothesis could have been supported by building models of ever-increasing coverage, all within a context-free formalism.

Similarly, the only way to show that the analysis of natural language does not require

theoretical constructs such as empty categories or movement is to build working grammars which handle all phenomena purported to require these devices, and then continue to extend the grammars from there to larger fragments of the language. But how does one discover phenomena which have not yet been analyzed? Baldwin et al. (2005) propose a method of using an existing implemented grammar to systematically sift through a corpus for phenomena not yet handled by the grammar (and exceptions to analyses of phenomena that have been incorporated).

Core and periphery Another pair of broad-brush syntactic hypotheses concern the distinction between "core" and "peripheral" syntactic phenomena. Some approaches to syntax delineate a set of "core" phenomena, leaving the remainder (the "periphery") outside the scope of syntactic theory. In contrast, Construction Grammar (Kay and Fillmore, 1999) and related frameworks take the stance that there is no sharp line between "core" and "periphery", that natural language syntax involves both broad generalizations and specific idiosyncrasies, and that these should be stored together in one formally consistent system. Bender and Flickinger (1999) further argue that in building grammars which countenance both core and peripheral phenomena, handling the exigencies of the latter (e.g., English tag questions) provides a means of selecting among competing analyses of the former (e.g., agreement).

2.3 Testing hypotheses

As a general rule, one cannot test a syntactic hypothesis merely by looking at surface strings. These properties typically aren't apparent in that fashion, nor are they directly accessible to introspection. Instead, syntacticians build models and test the predictions about grammaticality and paraphrase relations made by the model against judgments of acceptability provided by native speakers.

Such syntactic models vary in their explicitness. At one end, we find general arguments for or against particular properties of models. When such arguments are made in the absence of a worked-out grammar for a fragment of the language, it is typically not possible to fully calculate the predictions they make about any particular string. At the other end of the scale are elaborated models which are detailed enough that one can process test examples with them and arrive at predictions of string-meaning mappings as well as grammaticality. By building elaborated models (and implementing them in machine-readable form so that computers can handle the testing), we can validate that the assumptions at least work together as intended. More importantly, since the goal is to ultimately build one model that accounts for all of the known data pertaining to linguistic competence, we can use elaborated models to track how well various analyses work together.

3 Grammar Engineering

Grammar engineering is the practice of building elaborated linguistic models on computers. Grammar engineering has been used for many years now for practical purposes (e.g., Microsoft's grammar checker, Boeing's Simplified English grammar checker). Here we are concerned with its scientific uses. On the one hand, implemented grammars allow the computer to keep track of interactions between analyses. For example, if I add an analysis of case to a grammar which includes coordination, the computer can calculate the combined predictions of the two analyses regarding the grammaticality of coordinated noun phrases with different cases, or coordinated verb phrases with different requirements on the case of their (shared) subject. On the other hand, implementing a grammar allows the hypotheses encoded in that grammar to be tested against thousands of examples. Both of these tasks—calculating how analyses of separate phenomena interact and crunching through thousands of examples to calculate the grammar's predictions and how they differ from some previous state of the grammar—are well-suited to computers and ill-suited to humans.

Software support for creating implemented grammars has been around since the early days of generative grammar (Petrick, 1965, Zwicky et al., 1965, Friedman et al., 1971). As recently as 10 years ago, however, the technological state-of-the-art made grammar engineering quite tedious: parsers using the complex grammars required for deep linguistic analysis ran so slowly that running a thousand-sentence test-suite was an overnight affair. Over the past 10 years, however, advances in parsing technology (see e.g., Oepen et al., 2002) combined with Moore's Law[7] have changed the equation: it is now possible to run a substantial test suite quickly enough that such testing can be intimately integrated into the development cycle, allowing the grammar engineer to explore the consequences of each minor change to the grammar. Furthermore, since the processing of any given example is independent of the processing of all the others, it is easy to take advantage of computer clusters to process multiple examples in parallel, further reducing the time it takes to run a test suite.

The basic requirements for grammar engineering include:

1. A reasonably stable grammar formalism, typically with enough flexibility to encode a variety of theories. This in turn entails a distinction between the formalism (the language in which analyses or theories are encoded), the theory (a set of analyses) and the framework (a set of general guiding principles for syntactic inquiry).

   It is important for the formalism to be stable rather than rapidly evolving, because the formalism is interpreted by software which uses the grammars to parse and generate sentences. Creating such software requires defining (and refining) a particular version of the formalism. Subsequent changes to the formalism require changes to the software, which can be expensive. As a consequence, it is prudent to define a fairly flexible formalism, such that multiple theories can be explored without having to reimplement the associated software.

[7] The tendency of the processing power of computers to double every two years.

2. Algorithms for parsing (and ideally also generation). These algorithms, implemented as software, apply the grammars written by the linguist for processing. A parsing algorithm takes strings as input, and produces syntactic and/or semantic structures as output. A generation algorithm takes semantic structures (in an appropriate form) as input, and produces strings as output.

3. Grammar development tools. Using a grammar to parse and generate provides a wealth of valuable information. Nonetheless, in order to efficiently develop (and debug) such a grammar, it is useful to be able to manipulate it in various ways, including visualizing the components of the grammar and interactively stepping through derivations of problematic examples. A grammar development environment accordingly provides a suite of grammar visualization and debugging aides.

4. Test suite management software. Such software facilitates the batch processing of test examples and the comparison of results across different grammar versions.

Parsers and related tools are available for a variety of frameworks, including HPSG (Copestake, 2002, Meurers et al., 2002), LFG (Maxwell and Kaplan, 1996), CCG (Baldridge and Kruijff, 2003), and Minimalism (Stabler, 2001).[8] Unlike parsers and generators, test suite management tools (e.g., Oepen 2002) need not be committed to any particular formalism. As a result, a single test suite management tool can be integrated with parsers/generators from different frameworks, facilitating comparative analysis.

4 The LinGO Grammar Matrix

Despite the technological advances described above, it may seem like a daunting amount of work to implement a grammar large enough to begin to explore the particular phenomena a linguist is interested in. The LinGO Grammar Matrix (Bender et al., 2002, Flickinger and Bender, 2003, Drellishak and Bender, 2005, Bender and Flickinger, 2005) is designed to reduce this start-up cost by providing a substantial jump-start to the development of new grammars. The Grammar Matrix consists of a cross-linguistic core grammar (code shared by all grammars) and a set of phenomenon-specific 'libraries' which provide implementations for alternative realizations of phenomena such as basic word order, coordination, and negation.

The Grammar Matrix can be accessed through a web form which elicits information about the syntactic structures in the language to be modeled and the outputs a customized grammar 'start'.[9] The web form is arranged into phenomenon-specific sections. For example, the section on sentential negation asks whether sentential negation is expressed through

[8] As noted, the development of such software requires a commitment to a particular formalism. The formalisms defined by the software may be more or less closely related to the formalisms assumed in theoretical work.

[9] The Grammar Matrix customization and download page can be found here:

http://www.delph-in.net/matrix/customize/matrix.cgi

verbal inflection, an adverb or both. For verbal inflection, the user further specifies whether the affix in question attaches to auxiliaries only, main verbs only, or any finite verb; whether it is a prefix or a suffix; and how it is spelled. For adverbial negation, the user can specify an independent adverb (appearing to the left or right of V, VP or S) or a selected adverb (cf. Kim, 2000) (selected by main verbs, auxiliaries, or any finite verb). If a language uses both the inflectional and adverbial strategies for expressing negation, the user is asked to specify how they may or may not co-occur. The options are: complementary distribution, both inflection and adverb required, both optional, inflection required/adverb optional, and adverb required/inflection optional. On the basis of the user's answers to these questions, and with reference to their answers to other parts of the questionnaire, the customization system outputs the appropriate rules and constraints for associating the meaning of sentential negation with its form, as specified.

4.1 Assumptions

The jump start provided by any such system comes at the cost of accepting the assumptions of the system. The assumptions encoded in the Grammar Matrix come from three sources: the typed-feature structure formalism that we have adopted (called 'type description language' or 'tdl' and recognized by the LKB Grammar Development Environment (Copestake, 2002)), the general theoretical assumptions of Head-Driven Phrase Structure Grammar (HPSG, Pollard and Sag 1994), and particular implementation decisions we have made in the Matrix itself.

Some examples of assumptions or constraints that we inherit from the tdl formalism include: (i) no relational constraints; if the value of one feature depends on the value of another, the relationship is simply one of identity, (ii) any given phrase structure rule has a fixed number of daughters, (iii) tectogrammatic/phenogrammatic equivalence: the yield of the tree gives the surface string order.

Some examples of assumptions of hypotheses we adopt from HPSG include: (i) Monostratal approach: input strings are associated with a single, elaborated structure, rather than a sequence of such structures. (ii) No empty elements. (iii) Hierarchical organization: grammars consist of rich collections of constructions, which are arranged into hierarchies in order to capture the similarities across them. (iv) Local compositionality (cf. Szabó, 2005): the semantic representation associated with any particular constituent is a function of the semantic representation of its immediate subconstituents and the identity of the rule licensing the constituent. (v) X′ theory: most phrases are headed; heads select for complements, subjects, and/or specifiers; the 'category' of the mother is determined by the category of the head daughter and the remaining valence requirements. (vi) Selection of heads by modifiers and reciprocal selection of heads by specifiers. [10]

We adopt a particular precise formalism (tdl) so that we can implement the grammars, and therefore adopt the assumptions associated with that formalism. We adopt a particular

[10]These assumptions are not uniformly held within the HPSG community, and variants of the framework may reject one or more of them.

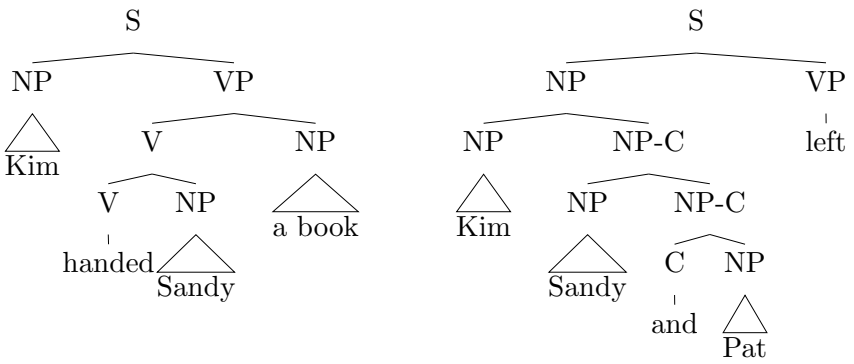

Figure 1: Recursive binary branching in head-complement and coordination constructions

framework (HPSG), for theoretical grounding as well as inspiration for particular analyses.[11] These two commitments narrow the space of possible grammars somewhat, but there are still a large number of possibilities remaining. In order to build working grammars, we must adopt many more working hypotheses. Some of the working hypotheses presently encoded in the Matrix include binary branching structure, a distinction between subjects and specifiers, the flat, surface-oriented semantic representations of Minimal Recursion Semantics (Copestake et al., 2005), and semantic monotonicity. Each of these is discussed briefly below.

Extensive use of binary branching structure For constructions with an unbounded or variable number of dependents, we posit recursive binary-branching rules rather than rules with varying number of daughters. In the case of adjuncts, this is not controversial. In the case of head-complement structures and coordination, it is somewhat more unusual. Nonetheless, we find the structures schematized in Figure 1 preferable to flatter structures because the rule systems required to produce the former (and associate them with appropriate semantic representations) are simpler.

The Grammar Matrix also allows for unary and ternary rules. The latter are a recent addition motivated by Hausa, which expresses negation through a pair of particles, one on either end of the clause, as illustrated in (1), (from Newman, 2000:363):

(1) bàà rashìn nāmàà (nē) zâi kashè mùtûm ba
NEG lack.of meat (focus marker) FUT kill person NEG
'It is not that lack of meat will kill a person.'

Treatment of valence features In the Grammar Matrix, we distinguish three kinds of arguments selected by heads: subjects, complements and specifiers. This classification encodes three primary syntactic hypotheses: (i) subjects and specifiers are distinguished from

[11] This relationship is symbiotic: Results from grammar engineering have informed theoretical work in HPSG as well.

complements as external arguments, (ii) there are no further grammatical function distinctions among complements, and (iii) subjects of verbs and other predicates require different treatment than specifiers of noun phrases and degree specifiers of modifiers. This last distinction is motivated by a semantic difference: In order to build semantic representations compositionally, each constituent 'publishes' information that constituents higher in the tree can use (cf. Copestake et al., 2001). In head-subject constructions, the syntactic head (e.g., a VP) controls which semantic information is published. In head-specifier constructions, it is the specifier (non-head) daughter which does so.

Minimal Recursion Semantics We adopt the system of Minimal Recursion Semantics (MRS, Copestake et al., 2001; Copestake et al. 2005) for our semantic representations. MRS was designed to meet the competing demands of computational tractability and linguistic adequacy. It achieves this through flat, underspecified structures which are hypothesized to be able to capture all linguistically relevant, syntactically marked semantic information while deferring disambiguation of distinctions like quantifier scope (to the extent that these are not constrained by the syntax). Minimal Recursion Semantics provides a general definition of well-formed semantic representations. Within this general definition, however, this is much room for specific realizations. In the Grammar Matrix, we adopt several design principles:

1. Semantic distinctions which are not syntactically marked in a language should be left underspecified. A possible example is sequential versus non sequential readings of VP and S coordination in English:

   (2) a. Kim entered the room and saw Sandy.

   b. Kim studied Latin and wrote books.

   Rather than produce two analyses for each of the sentences (with two different interpretations of *and*), we simply leave the distinction underspecified at the level of MRS. As long as any information internal to the sentence relevant to the disambiguation (such as the order of the conjuncts, the lexical predicates involved, and the tense/aspect information associated with each predicate) is reflected in the MRS, such disambiguation can be left to further processing steps.

2. Semantic representations should be closely tied to the surface string and surface syntactic relations. This is required by the combination of local compositionality and the monostratal theory we adopt. It is also useful in facilitating comparison (for practical applications) with the even more underspecified semantic representations gleaned from shallow processing techniques (e.g., Callmeier et al., 2004).

3. Semantic representations should be harmonized across languages to the extent possible. This facilitates the creation of language-independent software applications which can be specialized for a particular language by plugging in the right grammar. In addition, it is interesting in the context of machine translation. At the same time, we are not

treating MRS as an interlingua, as this design principle is often in tension with the second one. Languages differ, and so it will not be possible to completely harmonize "surfacy" semantic representations across languages.

Semantic monotonicity Lexical rules and phrase structure rules can add semantic information, but they cannot remove or alter information provided by their constituents.[12] For example, consider the causative-inchoative alternation illustrated in (3):

(3) a. The door opened.

b. Kim opened the door.

One way to represent the difference in meaning between these two uses of *opened* is to have *opened* in (3b) contribute a 'causative' relation in addition to the 'open' relation. If we wish to model this alternation via a lexical rule, the constraint of semantic monotonicity requires that the input be the inchoative *open* (3a) and the output the causative variant in (3b). In this way, the rule is adding the 'causative' relation, rather than taking it away.

A second example comes from languages which overtly mark definiteness but give unmarked NPs an indefinite interpretation. The situation is particularly interesting in Hebrew, where demonstrative adjectives can combine with either definite or bare nouns. An NP is interpreted as indefinite (i.e., a discourse status of 'type identifiable' in Gundel et al.'s (1993) system) if it contains neither the definiteness inflection nor any demonstrative adjectives.[13]

(4) a. klb ien
kelev yashan
dog slept
'A dog slept.'

b. klb zh ien
kelev ze yashan
dog this slept
'This dog slept.'

c. h-klb ien
ha-kelev yashan
DEF-dog slept
'The dog slept.'

d. h-klb h-zh ien
ha-kelev ha-ze yashan
DEF-dog DEF-this slept
'This dog slept.'

The interpretation of (4a) is not underspecified, but rather stands in contrast to the other three. One way to handle this would be to create a (phonologically inert) lexical rule for indefinites parallel to the lexical rule which adds the definite prefix. However, given examples like (4b), this approach runs afoul of semantic monotonicity: We cannot remove the information that a phrase gets an indefinite interpretation once it has been added. Thus instead

[12]This constraint is motivated by the goal of making our grammars bidirectional (useful in generation as well as parsing), as it significantly simplifies the search problem faced by the generator.

[13]Thanks for Margalit Zabludowski for drawing my attention to this phenomenon and providing the examples. The first line in each example gives a transliteration from standard orthography. The second is a transcription line representing the pronunciation. There is a subtle difference in meaning between (4b) and (4d) not indicated in the English glosses.

we delay the introduction of this information until the level of NP, where we introduce it just in case no demonstrative adjectives have been attached.

I have been alternately referring to these propositions as assumptions and hypotheses. This is because, given the current state of the art and the modes of inquiry available, we can only test linguistic hypotheses relative to the rest of the model. Thus at any given point, a constraint (or grammar rule, or other aspect of the model) may be the hypothesis being directly considered or a background assumption in the testing of another hypothesis. When one conceives of generative grammar in model-theoretic terms, there are two possible candidates for the modeling domain: the first is the set of acceptable sentences of the language in question, and the second is the actual knowledge of language encoded in the brains of speakers. The analysis (i.e., mapping of surface form to semantics and verification of well-formedness) of any particular string depends on multiple parts of a grammar. Thus under the first alternative, it is not generally possible to conclusively support or refute any particular part of a grammar independently of the grammar that embeds it. Even if we take the object of study to be the actual linguistic knowledge encoded in the brains of speakers, we are still typically checking the validity of our model by comparing its predictions with judgments provided by speakers.[14]

It should be noted that this relativity of theoretical results is common across syntactic frameworks, and is furthermore not limited to implemented models. Proposals in theoretical syntax always contain ancillary assumptions (explicit or implicit), and arguments for or against particular properties of grammatical models are typically only valid relative to those assumptions (Bender, 2002). This might seem to be more of a problem for grammar engineering than pen-and-paper syntax: Once an implemented grammar is built around certain assumptions, those assumptions take on a kind of inertia, as great effort can be required to revise away from them. When the model only exists in its descriptions in theoretical work, it would appear easier to revise any piece of it. I would argue, however, that this is actually an advantage, rather than a disadvantage, of grammar engineering. Without building a single model which integrates all of the analyses so far, and without testing that model against test suites representing all of the phenomena analyzed, it can be hard to tell when current assumptions are subtly incompatible with those required by previous analyses. If the goal is ultimately a model of linguistic competence, rather than separate models for separate aspects of linguistic competence, then the extra work required to maintain earlier analyses over time is beneficial. It is not a matter of slowing down theoretical progress so much as making theoretical results more durable.

[14]Psycholinguistics and neurolinguistics can provide more direct information about what people know about language and how they use that knowledge, and such information should inform the design of models of grammar (Sag et al., 2003, Ch. 9). Such investigations are expensive (in time and money), however, and cannot be used to test every single constraint of a model of syntax. It is far more efficient to narrow down the possibilities first on the basis of acceptability judgments. Furthermore, psycholinguistic and neurolinguistic studies are completely infeasible in the case of moribund or extinct languages.

4.2 Cross-linguistic hypothesis testing

The Grammar Matrix core contains constraints which are expected to be useful in all languages. These include the implementation of semantic compositionality, a large menu of valence patterns defined in semantic terms (i.e., abstracting away from the part-of-speech of the arguments, but specifying their semantic type as individuals, propositions/questions, or expletives), and a superset of part of speech types, arranged into a powerset of disjunctive types.

Limiting the Matrix to cross-linguistically valid constraints, however, would significantly limit its usefulness. There are recurring structures across languages which are not universal. To the extent that there are known analyses for these structures, the Matrix should be able to provide them. We have begun to do so by developing the phenomena libraries mentioned above. Work is presently underway on the expansion of existing libraries (for basic word order, coordination, sentential negation and yes-no questions) as well as on new libraries for case, agreement, tense, and aspect.

The core grammar and the libraries together allow us to take the hypothesis-testing benefits of grammar engineering to the level of crosslinguistic hypotheses, forming a kind of computational linguistic typology. In developing the libraries, we attempt to handle all known variants of each phenomenon (and document those which we are not yet able to account for) while harmonizing semantic representations across different language types. We furthermore aim for cross-compatibility of the libraries, such that each option in the word order library could in principle be paired with each option in the coordination library. Where cross-compatibility fails, we explore why: is there a logical incompatibility between the choices or a specific incompatibility in our analyses? Do the analyses of each particular option (e.g., VSO word order, or negation through verbal affixation) apply equally to all languages evincing that phenomenon?

Another approach to using grammar engineering for crosslinguistic hypothesis testing is represented by the ParGram project (see e.g., King et al., 2005). The ParGram project involves many grammar engineers at many sites creating broad coverage grammars for a variety of languages, while working to keep the analyses and representations used by the grammars parallel. In this way, data from many languages can inform the analyses they develop. The MetaGramamr (Kinyon et al., 2006) and KPML (Bateman et al., 2005) projects take yet another tack creating grammar resources in which each piece is tagged for which languages it applies to. Kinyon et al. have used this approach to explore the similarities and differences in V2 phenomena across a handful of V2 languages.

4.3 Evaluation

Monolingual grammars (and the hypotheses they represent) are evaluated by using them to process test suites. Ideally such test suite consist of both hand-constructed (positive and negative) data as well as naturally occurring corpus data. Evaluating the grammar against the test suite entails not only checking its predictions of grammaticality but also verifying that all of the analyses returned for each string are legitimate and lead to well-formed and

appropriate semantic representations.[15] Software assistance for this task is available, in the form of the treebanking software developed by the Redwoods project (Oepen et al., 2004). This system assists annotators to choose the preferred analyses for each sentence to be included in the treebank from among those returned by the grammar. Rather than examining all of the analyses directly, annotators select among binary 'discriminants' for each forest of trees, indicating, e.g., the attachment site of a particular prepositional phrase or the appropriate lexical entry in context for some word. The original purpose for this system was to create dynamic treebanks which could be updated as the grammar evolved (by re-running the annotator's choices among discriminants). It is also very useful for grammar engineers trying to understand how various analyses of a given string are licensed, and to verify that the analyses are legitimate. Furthermore, once a treebank has been annotated in this fashion, it can serve as a gold standard for regression testing as the grammar continues to evolve.

Evaluating a cross-linguistic resource such as the Matrix requires doing such testing repeatedly, against a variety of languages. In order to get interesting coverage over the test suites, however, the grammar starts provided by the Matrix customization system need to be extended. Thus any such evaluation is simultaneously evaluating the Matrix and the language-specific extensions which have been added to it. Nonetheless, the process of creating grammars on the basis of the Matrix has been extremely informative in terms of highlighting errors in the matrix (overly strong assumptions in the cross-linguistic core) as well as lacunae in the phenomenon libraries. For example, early work applying the Matrix to Norwegian (Ellingsen, 2004) pointed up the fact that constraining complements to attach before subjects disallowed VSO word order. More recently, work on Sanskrit and Inupiaq has highlighted additional coordination marking strategies not covered by Drellishak and Bender (2005). This Matrix has been tested most intensively in the multilingual grammar engineering course at the University of Washington, covering 42 languages in four years. The Grammar Matrix is also being used as the basis for several larger grammars undergoing sustained development, including grammars of Norwegian (Hellan and Haugereid, 2003), Modern Greek (Kordoni and Neu, 2005) and Spanish[16]. In addition, though the JACY grammar of Japanese (Siegel and Bender, 2002) predates the Grammar Matrix, JACY has since been adapted to comply with the Matrix.

A logically prior evaluation step, however, is the verification that the libraries are in fact functioning as intended. Because the libraries are not independent of each other (*tout se tient*), they need to be checked for appropriate interactions. The space of possible grammars is enormous (currently in the hundreds of thousands, and we're just getting started). Therefore, we have taken a strategy (first laid out in Poulson 2006) of creating a test suite of abstract strings over a shared vocabulary (e.g., *det n1 det n2 tv*). These strings are associated with gold standard semantic representations. The strings are then permuted to create new string-semantics pairs. We create a set of regular expression filters (relative to the semantic

[15]Verifying that all of the legitimate analyses are found is more subtle. Typical practice is to rely on the assumption that a grammar that fails to return a valid parse for one item is likely to fail to return any parse for another item.

[16]http://www.upf.edu/pdi/iula/montserrat.marimon/spanish_resource_grammar.html

identity of the string in question as well as individual properties of the language type being tested or small combinations thereof) which allow us to generate appropriate test suites for randomly-generated grammar starts.

5 The Montage Vision

Since 2003, work on the Grammar Matrix has been situated within the larger context of exploiting computational tools for language documentation (dubbed the 'Montage Project'). On the one hand, we hope to bring the hypothesis-testing power of grammar engineering to linguists engaged in primary linguistic documentation. On the other hand, applying the Grammar Matrix to as yet undescribed languages will help expand its typological coverage and provide an interesting test of its claimed linguistic universals.

The Matrix as it currently exists is not yet ready to be put to use directly by field linguists. To get there, it will need libraries covering a wider range of phenomena, as well as interfaces to tools for lexicon and morphological analyzer development.[17] In the meantime, however, the benefits of grammar engineering can be brought to field linguists by supporting collaborations between field linguists and grammar engineers.

The potential benefits include hypothesis testing and data exploration, allowing field linguists to ask such question as "Does the grammar account for the data in the texts?" and "What's here that we haven't accounted for?" (cf. Baldwin et al., 2005). In the longer term, we envision semi-automated annotation, where a small grammar together with underspecified lexical entries can create parses for as yet unanalyzed sentences. Furthermore, the Grammar Matrix can be a vehicle for bringing natural language technology to low-density languages. By creating grammars in a consistent format with consistent, though not identical, semantic representations, the Matrix facilitates the adaptation of software for grammar checkers, computer-assisted language learning applications and machine translation to new languages.

6 Conclusion

The tasks involved in formal linguistic analysis include:

1. inventing possible analyses,
2. calculating/verifying the predictions of the analysis for known data, and
3. determining what further predictions those analyses make, leading to further relevant data to test.

While being able to estimate (2) is an important skill, and actually typically integrated in (1), doing it with sufficient thoroughness is tedious and time consuming for people, while

[17] For example, SIL's Fieldworks (http://fieldworks.sil.org/) or EMELD's FIELD (http://emeld.org/tools/fieldinput.cfm).

well-suited to computers. (3) likewise is essential, and must be done at least in part by a human linguist. At the same time, the computer can nicely complement the work a person can do: relevant data may be lurking already in the existing test suite, and furthermore, processing naturally occurring corpora can turn up example types that the linguist may never arrive at otherwise.

I have argued that grammar engineering can be useful, though it is expensive and time-consuming. The Grammar Matrix reduces the cost of creating new grammars, and reduces it more with each library that is added. At the same time, it increases the benefits of creating implemented grammars by increasing both interoperability for cross-linguistic hypothesis testing and interoperability for practical applications.

Acknowledgments

This paper reports on work done in collaboration with Dan Flickinger, Stephan Oepen, Scott Drellishak, Laurie Poulson, Margalit Zabludowski and Jeff Good. The project has benefited greatly from the time and effort of the students in four years of multilingual grammar engineering courses at the University of Washington. I am also grateful for the input of audiences at at the University of Washington and at Texas Linguistic Society X. The development of the Grammar Matrix is supported by NSF Grant BCS-0644097 and a gift to the Turing Center at the University of Washington from the Utilika Foundation.

References

Adams, Douglas. 1979. *The Hitchhiker's Guide to the Galaxy*. Pan Books.

Baldridge, Jason, and Geert-Jan Kruijff. 2003. Multi-modal combinatory categorial grammar. In *Proceedings of EACL 2003*.

Baldwin, Timothy, John Beavers, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2005. Beauty and the beast: What running a broad-coverage precision grammar over the BNC taught us about the grammar — and the corpus. In S. Kepser and M. Reis (Eds.), *Linguistic Evidence: Empirical, Theoretical, and Computational Perspectives*, 49–70. Berlin: Mouton de Gruyter.

Bateman, John A., Ivana Kruijff-Korbayová, and Geert-Jan Kruijff. 2005. Multilingual resource sharing across both related and unrelated languages: An implemented, open-source framework for practical natural language generation. *Research on Language and Computation, Special Issue on Shared Representations in Multilingual Grammar Engineering* 3(2):191–219.

Bender, Emily, and Dan Flickinger. 1999. Peripheral constructions and core phenomena: Agreement in tag questions. In G. Webelhuth, J.-P. Koenig, and A. Kathol (Eds.), *Lexical and Constructional Aspects of Linguistic Explanation*, 199–214. Stanford, CA: CSLI.

Bender, Emily M. 2002. Review of Martin et al (eds) Step by step: Essays on Minimalist syntax in honor of Howard Lasnik. *Journal of Linguistics* 38:432–439.

Bender, Emily M., and Dan Flickinger. 2005. Rapid prototyping of scalable grammars: Towards modularity in extensions to a language-independent core. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing IJCNLP-05 (Posters/Demos)*, Jeju Island, Korea.

Bender, Emily M., 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 J. Carroll, N. Oostdijk, and R. Sutcliffe (Eds.), *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, 8–14, Taipei, Taiwan.

Bresnan, Joan. 1978. A realistic transformational grammar. In M. Halle, J. Bresnan, and G. Miller (Eds.), *Linguistic Theory and Psychological Reality*, 1–59. Cambridge, MA: MIT Press.

Brown, Peter F., Stephan A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics* 19:146–151.

Callmeier, Ulrich, Andreas Eisele, Ulrich Schäfer, and Melanie Siegel. 2004. The DeepThought core architecture framework. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, 1205–1208.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.

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

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

Copestake, Ann, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language & Computation* 3(2–3):281–332.

Copestake, Ann, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics*, Toulouse, France.

Drellishak, Scott, and Emily M. Bender. 2005. A coordination module for a crosslinguistic grammar resource. In S. Müller (Ed.), *The Proceedings of the 12th International Conference on Head-Driven Phrase Structure Grammar, Department of Informatics, University of Lisbon*, 108–128, Stanford. CSLI Publications.

Ellingsen, Liv. 2004. Norwegian word order in HPSG. In *Proceedings of the 11th International Conference on HSPG*, Leuven, Belgium.

Flickinger, Dan, and Emily M. Bender. 2003. Compositional semantics in a multilingual grammar resource. In E. M. Bender, D. Flickinger, F. Fouvry, and M. Siegel (Eds.), *Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Devel, ESSLLI 2003opment*, 33–42, Vienna, Austria.

Frege, Gottlob. 1914. Letter to Jourdain. In G. Gabriel (Ed.), *Philosophical and Mathematical Correspondence*, 78–80. Chicago: Chicago University Press. 1980.

Friedman, Joyce, Thomas H. Bredt, Robert W. Doran, Bary W. Pollack, and Theodore S. Martner. 1971. *A Computer Model of Transformational Grammar*. New York: Elsevier.

Gazdar, Gerald. 1981. Unbounded dependencies and coordinate structure. *Linguistic Inquiry* 12(2):155–184.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, and Ivan Sag. 1985. *Generalized Phrase Structure Grammar*. Cambridge, MA: Harvard Univrsity Press.

Gundel, J.K., N. Hedberg, and R. Zacharski. 1993. Cognitive status and the from of referring expressions in discourse. *Language* 69:274–307.

Hellan, Lars, and Petter Haugereid. 2003. Norsource: An exercise in matrix grammar-building design. In E. M. Bender, D. Flickinger, F. Fouvry, and M. Siegel (Eds.), *Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development, ESSLLI 2003*, 41–48, Vienna, Austria.

Kay, Paul, and Charles J. Fillmore. 1999. The *what's x doing y* construction. *Language* 75(1):1–33.

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

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

Kinyon, Alexandra, Owen Rambow, Tatjana Scheffler, SinWon Yoon, and Aravind K. Joshi. 2006. The metagrammar goes multilingual: A cross-linguistic look at the V2-phenomenon. In *In Proceedings of the Eighth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+8)*, Sydney, Australia.

Kordoni, Valia, and Julia Neu. 2005. Deep analysis of Modern Greek. In K.-Y. Su, J. Tsujii, and J.-H. Lee (Eds.), *Lecture Notes in Computer Science*, Vol. 3248, 674–683. Berlin: Springer-Verlag.

Maxwell, John, and Ron Kaplan. 1996. An efficient parse for LFG. In *Proceedings of the First LFG Conference*, Grenoble, France.

Meillet, Antoine. 1903. *Introduction à l'étude comparative des languges indo-européennes*. Paris: Hachette.

Meurers, Detmar W., Gerald Penn, and Frank Richter. 2002. A web-based instructional platform for constraint-based grammar foramlisms and parsing. In D. Radev and C. Brew (Eds.), *Effective Tools and Methodologies for Teaching NLP and CL*, 18–25. New Brunswick, NJ: Association for Computational Linguistics.

Montague, Richard. 1970. English as a formal language. In B. Visentini (Ed.), *Linguaggi nella Società e nella Tecnica*, 189–224. Milan: Edizioni di Communità. Reprinted in Montague (1974).

Montague, Richard. 1974. *Formal Philosophy: Selected Papers of Richard Montague, edited and with an introduction by Richmond Thomason*. New Haven CT: Yale University Press.

Newman, Paul. 2000. *The Hausa Language: an Encyclopedic Reference Grammar*. New Haven: Yale University Press.

Oepen, Stephan. 2002. *Competence and Performance Profiling for Constraint-based Grammars: A New Methodology, Toolkit, and Applications*. PhD thesis, Universität des Saarlandes.

Oepen, Stephan, Daniel Flickinger, Kristina Toutanova, and Christopher D. Manning. 2004. LinGO Redwoods. A rich and dynamic treebank for HPSG. *Journal of Research on Language and Computation* 2(4):575 – 596.

Oepen, Stephan, Daniel Flickinger, J. Tsujii, and Hans Uszkoreit (Eds.). 2002. *Collaborative Language Engineering. A Case Study in Efficient Grammar-based Processing*. Stanford, CA: CSLI Publications. forthcoming.

Partee, Barbara. 1979. Constraining Montague grammar: A framework and a fragment. In S. Davis and M. Mithun (Eds.), *Linguistics, Philosophy, and Montague Grammar*, 51–101. Austin TX: University of Texas Press.

Partee, Barbara H. 1984. Compositionality. In F. Landman and S. Veltman (Eds.), *Varieties of Formal Semantics: Proceedings of the 4th Amsterdam Colloquim, Sept. 1982*, 281–311. Dordrecht: Foris Publications.

Petrick, Stanley R. 1965. *A Recognition Procedure for Transformational Grammars*. PhD thesis, MIT.

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

Poulson, Laurie. 2006. Evaluating a cross-linguistic grammar model: Methodology and test-suite resource development. Unpublished ms., MA thesis, University of Washington.

Pullum, Geoffrey K., and Gerald Gazdar. 1982. Natural languages and context free languages. *Linguistics and Philosophy* 4:471–504.

Reddy, Michael. 1993. The conduit metaphor: A case of frame conflict in our language about language. In A. Ortony (Ed.), *Metaphor and Thought*, 164–201. New York: Cambridge University Press, 2nd edition.

Sag, Ivan A., Thomas Wasow, and Emily M. Bender. 2003. *Syntactic Theory: A Formal Introduction*. Stanford, CA: CSLI. Second edition.

Shieber, Stuart M. 1986. Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 8:333–343.

Siegel, Melanie, and Emily M. Bender. 2002. Efficient deep processing of Japanese. In *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization at the 19th International Conference on Computational Linguistics*, Taipei, Taiwan.

Stabler, Edward. 2001. Minimalist grammars and recognition. In C. Rohrer, A. Rossdeutscher, and H. Kamp (Eds.), *Linguistic form and its computation*, 327–352. Stanford, CA: CSLI.

Szabó, Zoltán Gendler. 2005. Compositionality. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy*. Spring 2005 edition.

Zwicky, A.M., Joyce Friedman, Barbara C. Hall, and D.E. Walker. 1965. The MITRE syntactic analysis procedure for transformational grammars. In *Proc. Fall Joint Computer Conference*, Vol. 67, Pt 1, 317–326.