From Database to Treebank: On Enhancing Hypertext Grammars with Grammar Engineering and Treebank Search

Emily M. Bender♠, Sumukh Ghodke♡,
Timothy Baldwin♡ and Rebecca Dridan♣
♠University of Washington, ♡University of Melbourne, ♣University of Oslo

Abstract This paper describes how electronic grammars can be further enhanced by adding *machine-readable grammars* and *treebanks*. We explore the potential benefits of implemented grammars and treebanks for descriptive linguistics, following the discursive methodology of Bird and Simons (2003) and the values and maxims identified by Nordhoff (2008). We describe the resources which we believe make implemented grammars and treebanks feasible additions to electronic descriptive grammars, with a particular focus on the Grammar Matrix grammar customization system (Bender et al. 2010) and the Fangorn treebank search application (Ghodke and Bird 2010). By presenting an example of an implemented grammar based on a descriptive prose grammar, we show one productive method of collaboration between grammar engineer and field linguist, and propose that a tighter integration could be beneficial to both, creating a virtuous cycle that could lead to more effective and informative resources.

1 Introduction

This paper describes how electronic grammars can be further enhanced by adding *machine-readable grammars* and *treebanks*, or sets of structured annotations produced by the machine-readable grammars.[1] Following Good (2004), we understand a descriptive grammar to be a set of annotations over texts and lexicon, including both prose descriptions and structured descriptions. In an electronic descriptive grammar, annotations are illustrated with exemplars drawn from the text but are understood to express generalizations over more examples. This is illustrated in Figure 1, from Good 2004. Machine-readable grammars can be understood as another kind of structured description. Because they are interpreted by computers, they are required to achieve a higher level of consistency, with descriptions of various phenomena integrated to form a cohesive whole (Bender 2008b). Implemented grammars can automatically produce annotations over individual examples, which can in turn be aggregated and searched (Ghodke and Bird 2010). This vision is illustrated in Figure 2.

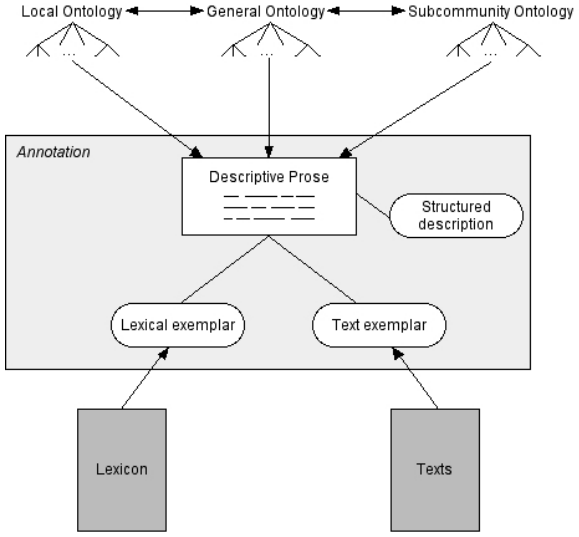

Figure 1: The structure of an annotation (Good 2004)

Our purpose in this paper is to explore the potential benefits of implemented grammars and treebanks for descriptive linguistics and to present to the descriptive linguistic community the currently existing tools which can facilitate their creation. In Section 2, we describe implemented grammars and treebanks and give an example of grammar engineering in the context of endangered languages. Section 3 describes treebank search, including use cases relevant to descriptive and documentary linguistics and how it can be integrated into an electronic descriptive grammar.

[1]We are grateful to the audience at the Conference on Electronic Grammaticography and an anonymous reviewer for helpful discussion. This material is based in part upon work supported by the National Science Foundation under Grants No. 0644097 and No. 0317826, and the Australian Research Council under Grant No. DP0988242. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

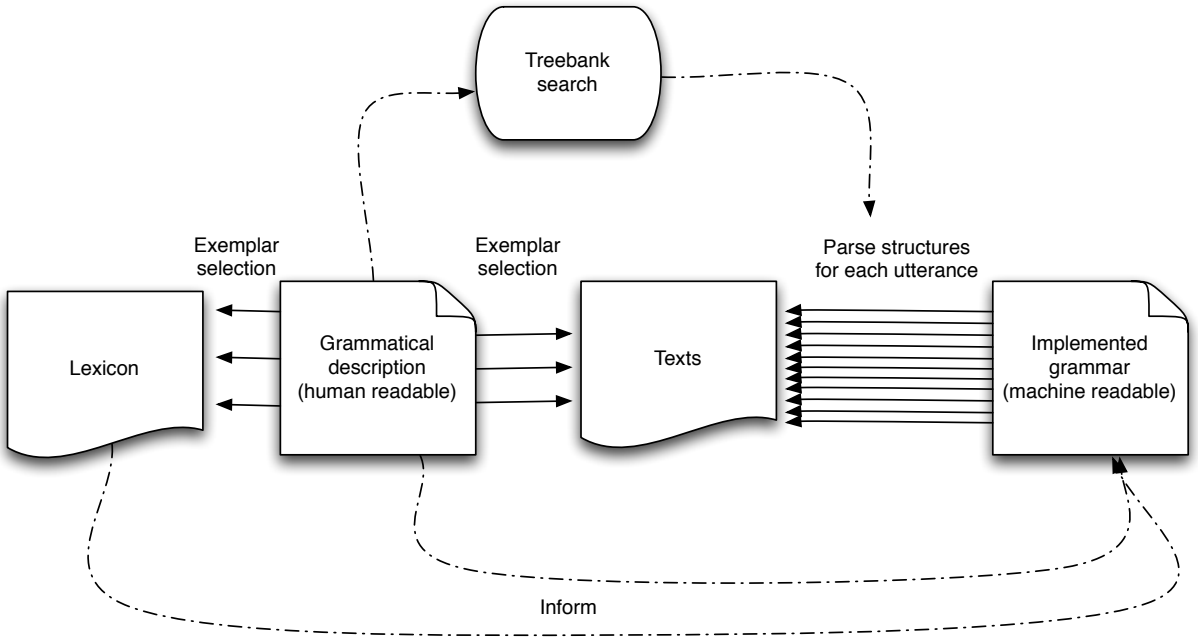

Figure 2: Schematic view of electronic grammars with treebanks

Following the discursive methodology of Bird and Simons (2003) and the values and maxims identified by Nordhoff (2008), in Section 4 we explore the impact of augmenting descriptive grammars with treebanks. Finally, in Section 5 we explore the resources which we believe make implemented grammars and treebanking feasible additions to electronic descriptive grammars.

2 Background

2.1 Implemented grammars

Implemented grammars are collections of linguistic rules written in a formalism that can be interpreted by appropriate software in order to apply those rules to linguistic inputs.[2] These inputs can be sentences of the language, in which case the goal is generally to find the syntactic and/or semantic structures assigned to those sentences by the rules (in parsing). If the rules are morphological rules, the inputs are word forms and the outputs morphological analyses of the word forms. Phonological rule sets map surface forms to underlying phoneme or feature sequences. In many cases, implemented grammars are reversible, allowing processing that takes more abstract structures (semantic representations, morphological analyses, underlying phoneme

[2]Implemented grammars of this sort are also descriptive in the sense that they capture linguistic generalizations. For present purposes, we will use the phrase *(electronic) descriptive grammars* to refer to prose statements of linguistic analyses and *implemented grammars* or *machine-readable grammars* to refer to formal sets of rules that can be interpreted by a computer.

sequences) and produces surface forms. Software for working with such rule sets is most developed for syntax (e.g., LKB (Copestake 2002), XLE (Crouch et al. 2001), TRALE (Meurers et al. 2002)), morphology (e.g., XFST (Beesley and Karttunen 2003) and the morphological engine in SIL's FieldWorks (Black and Simons 2008)), and phonology (e.g., XFST), but in the future one can expect other levels of linguistic structure to receive similar treatment. Implemented grammars can be extremely valuable for linguistic hypothesis testing, allowing linguists to check their analyses of different phenomena for consistency (Bierwisch 1963, Müller 1999, Bender 2008b, Bender et al. 2011) and to discover counterexamples to analyses in collected texts (Baldwin et al. 2005).

It is worth noting that while implemented grammars are necessarily *formalized* (i.e., written in some formalism which is precise enough for a machine to handle), they are not typically *formalist*. That is, where a formalist approach to linguistics attributes explanatory power to formal structures, and as a result, typically seeks to state theoretical results in the form of constraints on the allowable formal devices, grammar engineering uses formal structures in order to state analyses and typically favors flexible formalisms which allow for the exploration of multiple analyses (Bender et al. 2011). This practical approach to capturing linguistic generalizations is therefore not at odds with the goals of documentary and descriptive linguistics.

Bender's (2008a, 2010) work on developing an implemented grammar for Wambaya [wmb] on the basis of Nordlinger's (1998) descriptive grammar serves as a proof of concept of the applicability of the computational tools referenced above to endangered languages and language description. Bender (2008a) reports that in 5.5 person-weeks of work, she was able to create an implemented grammar on the basis of Nordlinger's descriptive (prose) grammar that assigned correct analyses to 91% of the exemplar sentences in the grammar and 76% of a held-out test set (one of the transcribed, glossed and translated narratives included in Nordlinger's volume).

Our purpose in citing these numbers here is to show the feasibility of grammar engineering in the context of language documentation and to emphasize its relative inexpensiveness: The grammar engineering effort built on Nordlinger's original fieldwork and analysis, and was minor in comparison, representing about 1/20th of the time. Furthermore, this case study illustrates that grammar engineering for language documentation can be done collaboratively: this methodology does not rely on individuals mastering both the skill sets required for linguistic field work and for grammar engineering, a point we return to in Section 5 below.

To make the notion of implemented grammars more concrete, we include a set of sample rule types from the Wambaya grammar in Figure 3. These types are written in TDL (Copestake 2000), the formalism interpreted by the LKB grammar development environment (Copestake 2002), and represent part of an HPSG (Pollard and Sag 1994) analysis. The supertype in this set `wmb-head-2nd-comp-phrase` describes all rules that allow a head to combine with its second complement regardless of whether it has already combined with the first (a key piece of the analysis of Wambaya's nearly free word order). The two subtypes integrate the constraints on the supertype with those on other types to define the head-initial and head-final variants of the rule (again related to free word order). The rules combine a head daughter with a non-head daughter, and match the constraints on the non-head daughter with the constraints on the second complement of the head-daughter (through the identity tag `#synsem`). These types inherit from further types

```
wmb-head-2nd-comp-phrase := non-1st-comp-phrase &
  [ SYNSEM.LOCAL.CAT.VAL.COMPS [ FIRST #firstcomp,
                                 REST [ FIRST [ OPT +,
                                                INST +,
                                                LOCAL #local,
                                                NON-LOCAL #non-local ],
           REST #othercomps ]],
    HEAD-DTR.SYNSEM.LOCAL.CAT.VAL.COMPS [ FIRST #firstcomp,
                                          REST [ FIRST #synsem &
                                                       [ INST -,
                                                         LOCAL #local,
                                                         NON-LOCAL #non-local ],
                                                         REST #othercomps ]],
    NON-HEAD-DTR.SYNSEM #synsem ].
head-comp-phrase-2 := wmb-head-2nd-comp-phrase & head-arg-phrase.
comp-head-phrase-2 := wmb-head-2nd-comp-phrase & verbal-head-final-head-nexus.
```

Figure 3: Sample rule types from Wambaya grammar

which handle aspects of the rules such as semantic compositionality and ensure that the non-head daughter is linked semantically to the appropriate role in the semantics of the head daughter.

Again in the interests of making the notion of implemented grammars concrete to readers who have not worked with them, Figure 4 shows a screen shot of the LKB processing the Wambaya sentence in (1) from Nordlinger 1998, 75.

(1) Gujarrarna nyilangunya nga yanybi.
two.II(ACC) echidna.II(ACC) 1.SG.A-PST get
'I got two echidnas.' [wmb]

The seven trees shown in the figure represent the seven analyses that the current implemented Wambaya grammar licenses for this sentence. (Note that each analysis is in fact much more detailed; the trees are merely abbreviations of larger syntactico-semantic structures which can be accessed through the software.) Of these seven, the last (bottom left) matches the gloss provided by Nordlinger (shown in (1)). In that analysis, the constituent *Gujarrarna nyilangunya* ('two echidnas') combines with the constituent *nga yanybi* ('AUX get') via the `comp-head-phrase-2` rule.[3]

Returning to the goals of integrating treebanks with electronic descriptive grammars, the examples above highlight two important points about implemented grammars and treebanks: First, annotations derived from implemented grammars make explicit ambiguity in natural language that speakers rarely notice and even linguists often skip over, focusing only on the relevant reading of the examples they are interested in.[4] In most cases, however, only one of the analyses will corre-

[3]This is because the NP 'two echidnas' is the second complement of the auxiliary+verb cluster 'AUX get'. The first complement of the auxiliary is the verb itself.

[4]One way that an implemented grammar can be incomplete is by licensing more analyses than are actually warranted for a string. Not all ambiguity, however, is spurious ambiguity (i.e., involving grammatically ill-formed struc-

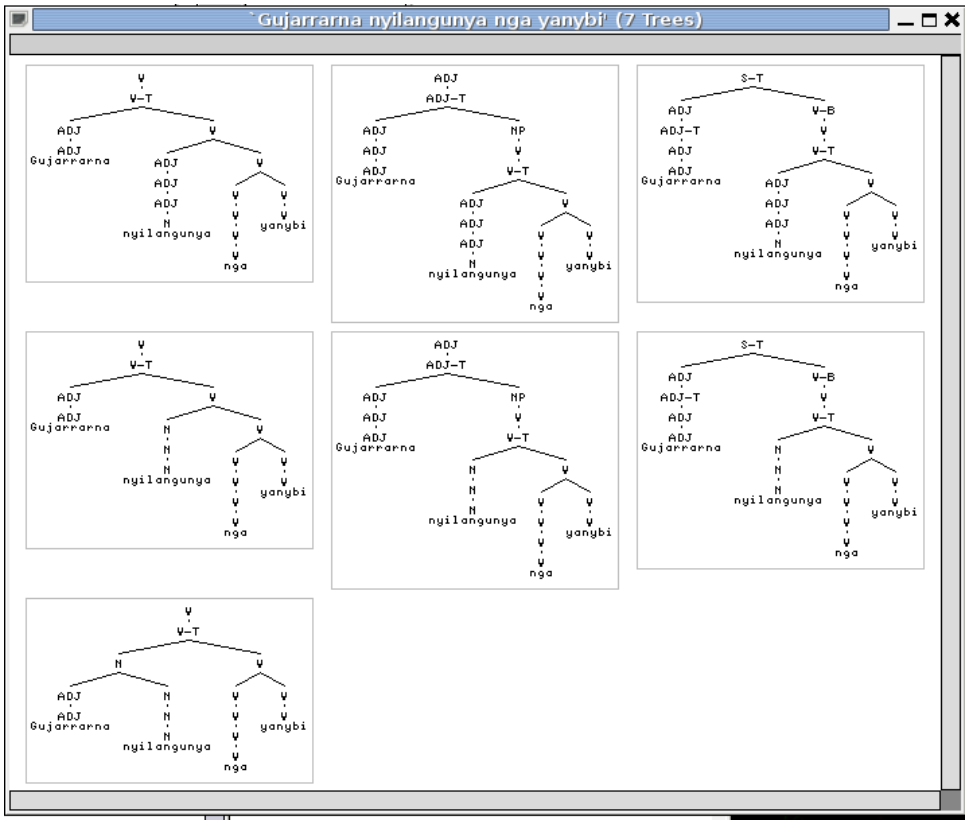

Figure 4: Seven analyses of (1)

spond to a pragmatically plausible reading. Creating a treebank involves selecting that pragmatically plausible analysis, as discussed further in Section 2.2 below. Second, implemented grammars and tools like the LKB are not enough to meet the needs of grammar readers looking to use the annotations in order to find relevant examples in a corpus. A separate set of tools is required which can take the output of the implemented grammar and the treebanking process and provide search functionality. Fangorn (Ghodke and Bird 2010) is a treebank search application that fills this need, as discussed further in Section 3.

2.2 Treebanks

A treebank is a collection of natural language utterances annotated with tree structures. Traditionally, treebanks have been created by annotating the tree structures by hand, using a detailed annotation guide. To speed up the process, the text may be first processed with software tools such as a shallow parser or chunker, so that most of the manual annotation consists of correcting and elaborating an initial structure, rather than writing trees "free-hand". The most well-known

tures). Any grammar with reasonable coverage will necessarily find multiple legitimate analyses of almost any grammatical string it can analyze. One of the lessons of computational linguistics for linguistics is the sheer amount of ambiguity in natural language (Abney 1996).

treebank for English, the Penn Treebank (Marcus et al. 1993), was constructed in just such a fashion. This manual method of creating a treebank has many drawbacks. The most forbidding is the amount of time required, even with the first automatic pass, but there are other problems. Validity of the annotation is one: It is very difficult to maintain consistency when annotating complex structures by hand, particularly when different phenomena can interact in many ways. Another issue is the static nature of these treebanks. Given the amount of time and money needed to create them, once a treebank is annotated, it is rarely updated. This means that out-of-date analyses are kept, even if later investigation suggests a better hypothesis.

Dynamic treebanks, such as the Redwoods treebank (Oepen et al. 2004), are produced by a newer method of treebank construction that uses an automatic parser to process utterances according to an implemented grammar. Manual annotation is still required, but in this case the annotator selects from the possible trees produced by the grammar to nominate the most plausible (with the option of rejecting all trees in case the grammar does not provide a suitable analysis). Since the annotator never edits tree structure manually, all annotations in the final treebank are guaranteed to conform to the implemented grammar. Aside from the benefits of consistency and speed, dynamic treebanks also have the advantage of being easy to update when the grammar changes, as described below.

For this type of treebank, once the text is parsed, the annotator selects the correct tree by making a series of binary decisions based on so-called parse discriminants (Carter 1997). Figure 5 shows the interface of the Redwoods treebanking tool used for this process, with the trees displayed on the left, and the discriminants on the right. Each discriminant represents a single aspect of analysis that occurs in some trees, but not others. These differences could be in the syntactic or in the semantic representation, where a syntactic discriminant consists of a grammar rule and the portion of the sentence it is being applied to, and a semantic discriminant describes a predicate and one of its properties or arguments. An annotator can click on a discriminant and then select *Yes* to indicate that it is correct, or select *No* to exclude any trees that are compatible with this discriminant. Since there are many dependencies between discriminants, selecting one can entail decisions for many others, meaning that finding the correct tree may only require a small number of decisions. These decisions are saved, and can then be used to (semi-)automatically update the treebank after a change has been made to the implemented grammar. When the text is re-parsed with the new version of the grammar, the old decisions can be replayed where applicable, and then the annotator only needs to annotate items that are still ambiguous after the old decisions have been applied. In this way, treebanks can be easily updated to reflect improvements in the grammar without the need for complete (and costly) re-annotation.

In addition to their use in linguistic exploration, these treebanks can also be used to build a statistical *parse selection* model (Johnson et al. 1999, Toutanova et al. 2005), which can be used to rank parser output by probability. While most human-detectable ambiguity requires contextual information to resolve, the large majority of implausible analyses can be ruled out on the basis of sentence-internal patterns. These patterns are probabilistic and very difficult to model with hand-written parse ranking rules, but are well handled by the machine learning (pattern recognition) techniques prevalent in computational linguistics. For present purposes, parse selection is of interest because it makes subsequent treebanking efforts easier. By learning characteristics of the trees

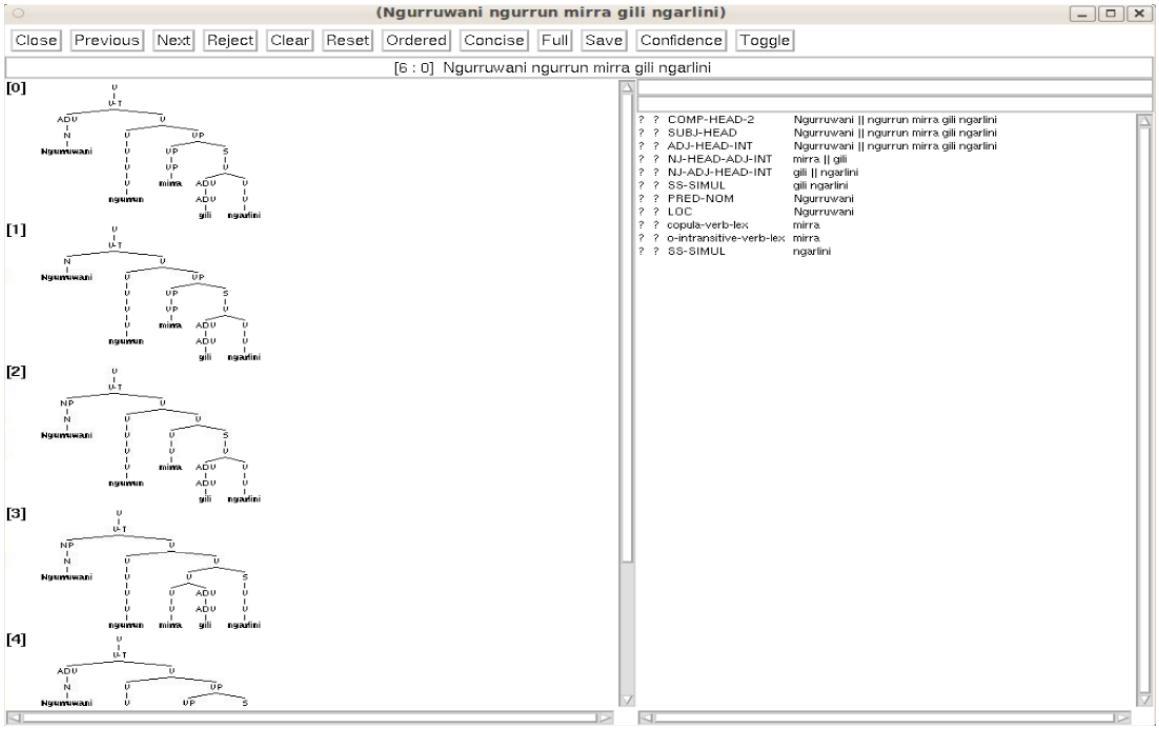

Figure 5: Treebanking tool

selected as correct in a first round of annotation, a parse selection model can be used by the parser to automatically discard the most improbable analyses before the annotator sees them, speeding up the annotation process.

2.3 Summary

In this section we have provided background on both implemented grammars and treebanks, with the goal of sketching the technology and methodology available. The following section will build on this to describe how treebanks can be used for linguistic research purposes.

3 Using Treebanked data

Treebank exploration is simplified by the use of specially designed search tools. These tools read in treebanked corpora and on request provide instances of trees, similar to the one shown in Figure 6, that match the sought tree-pattern of linguistic significance from within a corpus. Common features of such tools are: a query language to specify the pattern of interest and a user-interface to view the matching exemplars from the corpora.

Treebank search tools such as TIGERSearch (Lezius and König 2000) and TGrep2 (Rohde 2005) have been available for about a decade now. Only recently, however, with the development

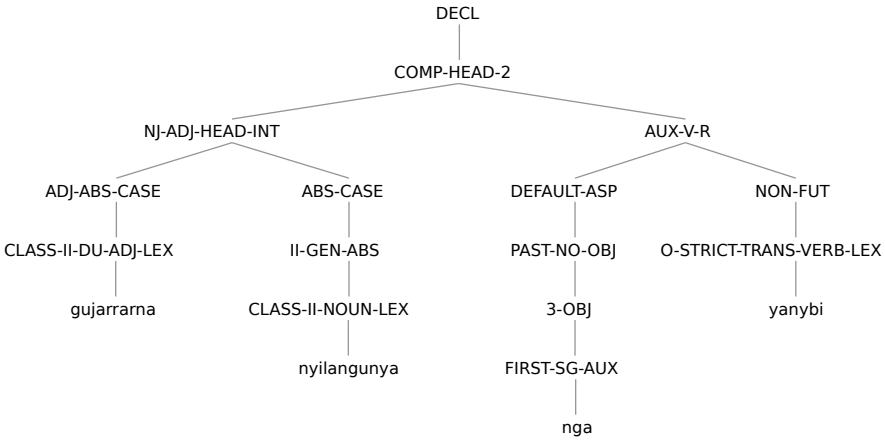

Figure 6: Treebank tree format for (1)

of Fangorn (Ghodke and Bird 2010) has a solution become available which allows for efficient search over large-scale treebanks.

3.1 Fangorn

Fangorn uses a path-based query language that is a subset of the LPath query language (Bird et al. 2006). For a detailed comparison of different treebank query languages see Lai and Bird (2004).

Path query languages are used to specify tree nodes of relevance. A path is a sequence of required node labels together with operators that state the relationship between consecutive labels. In other words, the path functions as a linear specification of nodes in trees of interest. For example, one interesting property of Wambaya is that modifiers can generally stand alone in argument positions (Nordlinger 1998). A linguist interested in sentences with this property might begin with a query like (2), which finds all parses where a declarative clause (label: DECL) has a complement of a verb realized by a modifier (label: HEAD-COMP-MOD-2) below it.

(2) //DECL//HEAD-COMP-MOD-2

The operator // in (2) is a *descendant* operator. Table 1 lists the different query operators in the Fangorn query language. The operators have been categorized into two types: horizontal and vertical, depending on whether they specify dominance or sequential positional constraints. The semantics of the vertical navigation operators are the same as their definition in the context of a tree. Similarly, the names for the sibling operators are mnemonic for their functions. The *following* operator specifies that the node to the right of the operator is temporally after the node to the left of the operator. The *immediately following* operator specifies that the leftmost descendant of the node to the right is temporally immediately after the rightmost descendant of the node to the left of the operator. The *preceding* and *immediately preceding* operators are the inverse of *following* and *immediately following*, respectively.

Vertical navigation		**Horizontal navigation**	
descendant	//	following	-->
ancestor	\\	preceding	<--
child	/	immediately following	->
parent	\	immediately preceding	<-
		following sibling	==>
		preceding sibling	<==
		immediately following sibling	=>
		immediately preceding sibling	<=

Table 1: Query operators and their symbols

The first operator in the query specifies whether the query pattern should appear at the root of the tree or anywhere in the tree. For example, if query (2) started with a *child* operator rather than a *descendant* operator as it currently is, then, the pattern /DECL//HEAD-COMP-MOD-2, would match only if the topmost node of the tree was a declarative clause and that node had a HEAD-COMP-MOD-2 under it.

Query (2) specifies a path consisting of two nodes, however, paths could contain any number of operator-node pairs. Both vertical and horizontal navigation operators can appear at any position in the path. The only restriction is on the first operator in the main path in the query. It has to be either a *child* or a *descendant* operator.

An operator in a path specifies the relationship between the node preceding and following it. In some cases, however, we would want to specify more than one relationship at a single node. For example, we may want to modify query (2) so that the head and complement positions are irrelevant, which means the node under the declarative clause could be either HEAD-COMP-MOD-2 or COMP-HEAD-MOD-2. Each of the two labels has a *descendant* relationship with the declarative clause DECL. Example (3) describes such a query.

(3) //DECL[//HEAD-COMP-MOD-2 OR //COMP-HEAD-MOD-2]

Square brackets after any node in a path are called *filter expressions*, and can be used to indicate alternatives (a split into two or more possible paths) or conjoined constraints on a node as shown in (3). The paths within a filter expression are connected using logical operators AND, OR and NOT to add flexibility to the constraints. Further, since the paths within filter expressions are themselves paths again, queries can have nested filter expressions.

Fangorn uses a web-based user interface, where the page contains a search box and a result display area to show the matching trees. A screen shot of the user interface for query (2) is shown in Figure 7. The left-hand top corner of the display area shows the total number of trees that match the query pattern in the corpus. The first page, showing 10 trees, is displayed by default. The user can choose to navigate to other pages. Each tree may match the query pattern more than once. Hence, the total number of matches within the tree and the match currently being displayed is shown at the top of each tree. Other matches within the same tree can be viewed using the '<' and '>' buttons at the top of each tree. The trees are minimally expanded to show the results in a

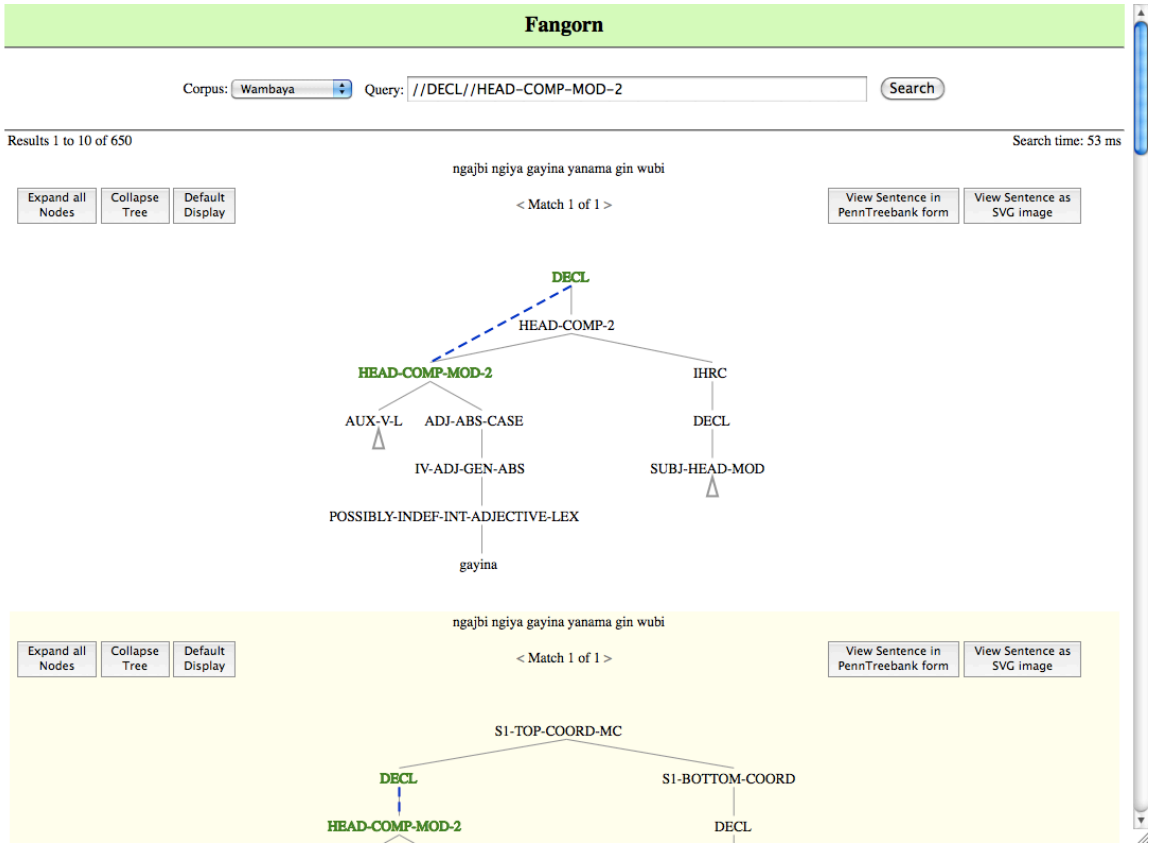

Figure 7: Fangorn's web interface

concise manner, but additional nodes can be expanded or collapsed in the display. The nodes that match the query are highlighted and joined by lines that denote the operators between the nodes. Each matching tree can be exported in either a bracketed format or as an SVG image identical to how the tree is displayed on screen.

Fangorn can be used in a number of modalities, including linguistic exploration, grammar engineering, and cross-linguistic comparison.

Linguistic exploration can be viewed as a kind of "exploratory data analysis" (Tukey 1977), whereby users query for particular lexico-syntactic patterns in a given language, and, e.g., explore the productivity of a construction, investigate the interactions between different constructions, investigate the distribution/behavior of a given construction across different domains (as instantiated in different treebanks), or simply observe the distribution of given lexical items in different syntactic contexts. Resnik et al. (2005) is a good example of this sort of exploration using linguistic structure, although that work was not tied to a descriptive grammar.

Grammar engineers can use Fangorn to validate a new analysis, by analyzing all instances of the given lexical rule or construction in trees licensed by a grammar. Traditionally, Fangorn has been applied to "gold standard" disambiguated trees. In indexing a larger set of analyses licensed by the grammar (e.g., the top-500 analyses, as selected by a parse selection model), however, it is

possible to retrieve all analyses in which a given pattern occurs, allowing the grammar engineer to gauge whether an analysis is adding spurious ambiguity. Fangorn can also aid in the education of grammar engineers, in exploring how the analysis of a given construction is manifested in syntactic trees, and comparing this back to the "source code" for the analysis in the grammar files.

Finally, assuming a comparable label set between grammars of different languages, it is possible to perform cross-linguistic queries to compare, e.g., differences in right-node raising between English and German in a data-driven manner. We come back to this briefly in Section 5.3.

3.2 Incorporating Treebank Search in Descriptive Grammars

Good (2004) presents a vision of electronic descriptive grammars as linked to searchable corpora, where exemplars chosen by the author to illustrate particular phenomena can be linked back to their original context, and additional examples can be retrieved from the database. We see the main benefit of treebanks to descriptive grammars as enriching the range of ways in which examples can be retrieved.

Treebank search can be incorporated into a descriptive grammar in two different (and complementary) ways: (i) The author of the grammar can include specific "canned" queries at various points in order to allow the reader to retrieve examples with properties relevant to the discussion (this would of course be in addition to providing exemplars, as is usual practice); (ii) The treebank search interface can be made available to the reader to input arbitrary queries matching their own interests. At present, Fangorn allows searching over tree structures with the nodes labeled by the rule (phrasal or lexical) or lexical type which licensed them, as well as over the words at the leaves of the tree. For the purposes of inclusion in descriptive grammars, it would be useful to extend that search capability to include the other annotations over the data (i.e., glosses and translations). Longer term, it would also be interesting to include searches over the feature structures abbreviated by the tree structures, including especially the embedded semantic representations.

The addition of "canned" queries will be relatively straightforward (once the implemented grammar and treebank are built), and will most likely be done in collaboration between the field linguist writing the descriptive grammar and the grammar engineer developing the treebank (cf. Section 5). The exact mechanism for making the canned queries and associated results accessible to users of the grammars is open to debate, but can potentially build off the work of Hashimoto et al. (2007) on lexical type documentation via illustrative positive and negative examples.

Making the search interface itself useful to readers will require documentation. General documentation about the query language will be applicable to all such treebank-enhanced grammars, but information about the implemented grammar licensing each treebank will also be required. The "canned" queries themselves will provide a useful part of this documentation, serving as models for other similar queries. In addition, the documentation should include a glossary of all of the labels in the treebanks (e.g., names of phrase structure rules, names of lexical rules, names of lexical types, as well as category labels used). Ideally, this glossary would include links to the relevant sections of the descriptive grammar and thus be accessible from those sections as well (following the links in the opposite direction).

3.3 Summary

In this section, we have given a brief overview of Fangorn, how it can be used to formulate queries, and how those queries could be used to assist readers of electronic descriptive grammars in getting information from an associated treebank. In the following section we reflect further on how implemented grammars and treebanks can help fulfill the goals of descriptive and documentary linguistics.

4 Values and Maxims

Bird and Simons (2003) structure a discussion of best practices for creating portable (and thus useful and enduring) language documentation around a series of value statements and maxims that follow from those values. Nordhoff (2008) picks up that discussion with a particular focus on how values identified by Bird and Simons influence the form of electronic descriptive grammars and inform the design of software supporting the development of such resources. Nordhoff is focusing in particular on those values that are relevant to electronic grammars with non-linear (i.e., graph-like) structure.

In this section, we explore how the addition of treebanks to electronic descriptive grammars can respond to some of those values, with a particular focus on those that treebanks speak to, either because they can enhance the ability of an electronic grammar to fulfill a maxim or because they would in fact be problematic in some way. Following Nordhoff, we structure the discussion according to the general areas of data quality, grammar creation (by authors), and grammar exploration (by readers). All of the maxims are given in the consequent of a conditional with the associated value in the antecedent, as in Bagish (1983), the inspiration for Bird and Simons's (2003) approach to discussing these issues.[5] This discussion includes both near-term goals using existing technology as well as longer-term possibilities.

4.1 Data quality

(4) ACCOUNTABILITY: If we value the application of the scientific method, more sources for a phenomenon are better than fewer sources (Rice 2006, 395, Noonan 2006, 355).

This maxim is the most obvious win for treebank and treebank search enhanced electronic grammars: If an electronic grammar is paired with a database of interlinear glossed text (IGT), it is already possible to search for some phenomena in that database. The trees in a treebank make much more of the structure of the sentences explicit than even the most meticulous IGT, and thus treebanks make it possible to more easily find examples of a broader range of phenomena (cf. Section 3).

[5]Except where noted, the antecedents and consequents are direct quotes from Nordhoff (2008:308–318). Additional citations are provided when Nordhoff indicated other sources for the maxims. Where there are multiple maxims for the same value statement, we have kept them as separate statements or merged them into one according to the most convenient structure for the present discussion. In Nordhoff's terminology, 'GD' stands for 'grammatical description', i.e., what we are referring to here as an electronic descriptive grammar.

(5) ACCOUNTABILITY: If we value the application of the scientific method, every step of the linguistic analysis should be traceable to a preceding step, until the original utterance of the speaker is reached.

As noted above, an implemented grammar requires that the various analyses it implements be integrated into a cohesive whole. The flip side of this is that every tree in a treebank represents several levels of linguistic structure. In grammars such as the Wambaya grammar discussed in Section 2.1, these include semantic, syntactic and morphological analyses. Thus, to the extent that the reader is supported in exploring the trees, the trees themselves will help ground semantic and syntactic analyses in previous steps. The connection between the morphological string and the original utterance will have to be handled outside of the treebank, however.

(6) ACCOUNTABILITY: If we value the application of the scientific method, the context of the utterance should be retrievable (Weber 2006:450).

Nordhoff discusses this one in terms of the communicative context (who's speaking, to whom, with what goals, etc). We assume that if this information is documented in the database underlying the treebank, then it should be accessible from the treebank as well. However, another issue relating to context arises for treebanks, namely the importance of preserving the linguistic context. All implemented grammars are in fact grammar fragments, and thus will not necessarily have complete coverage over arbitrary samples of naturally occurring text. With reasonably mature grammars there are robustness strategies (Kiefer et al. 1999, Riezler et al. 2002) that potentially allow for partial analyses in a treebank, but these are unlikely to be applicable or desirable in this application. Thus, a treebank associated with an electronic descriptive grammar will necessarily have gaps, i.e., sentences which are not assigned any trees. In order to preserve the linguistic context of the examples which are assigned trees, and which thus can be retrieved with Fangorn, it will be important to maintain links between the treebank and the underlying dataset.

(7) ACTUALITY: If we value scientific progress, a GD should incorporate provisions to incorporate scientific progress.

Nordhoff notes descriptive grammars are never finished. In the same way, implemented grammars also always have room to grow. It is a major benefit of the Redwoods approach to treebank construction (Oepen et al. 2004) that treebanks can be cheaply and rapidly updated when the grammar that produced them has been changed. Thus modern treebanking methodology makes it possible for electronic grammars with treebanks to rise to this maxim.

(8) HISTORY: If we value the recognition of the historic evolution of ideas, the GD should present both historical and contemporary analyses (Noonan 2006:360).

The same software that supports the creation of treebanks ([incr tsdb()], Oepen and Flickinger 1998) allows for detailed comparisons between treebanks based on different grammar versions. The primary purpose of these comparisons has been to allow grammar engineers to explore the impacts of various changes they have made to the grammar, in terms of which items (sentences) are assigned different (or more or fewer) analyses by one version of a grammar than another. It

is possible that this same software could be adapted to facilitate the exploration of the evolution of analyses either of particular examples in an implemented grammar, or of classes of examples. Doing so in a way that would make it informative for linguists who are not grammar engineers would, however, require significant additional user interface effort.

4.2 Grammar creation

The particular maxims that Nordhoff proposes under this heading are specific to the design of a grammar-authoring platform that supports the creation of electronic descriptive grammars, and therefore don't speak to the creation of implemented grammars. Accordingly, instead of reviewing Nordhoff's maxims, we have proposed some of our own that concern the creation of implemented grammars (and thus treebanks), but relate to the same set of values.

(9) ASSISTANCE: If we value speed of creation and comparability (across grammars), we should seek to provide means to assist linguists in rapidly creating comparable implemented grammars.

This is in fact the goal of the Grammar Matrix project (Bender et al. 2002, Bender et al. 2010). The Grammar Matrix provides a common core grammar, which defines things such as the format of semantic representations (using Minimal Recursion Semantics (Copestake et al. 2005)), an implementation of semantic compositionality, and general types of rules and lexical entries. In addition, the Grammar Matrix provides a set of libraries of analyses of cross-linguistically variable phenomena. These libraries are developed on the basis of a review of the typological literature, though of course are not assumed to be comprehensive: the project always anticipates the addition of new options within a library, as well as changes to the core grammar. The Grammar Matrix is described further in Section 5 below.

(10) CREATIVITY: If we value the individual mind's expressive abilities, support for creating implemented grammars should not preclude the linguist exploring alternative analyses in the implemented grammar.

The Grammar Matrix is in a sense analogous to the prose templates that Nordhoff proposes as part of a grammar authoring platform. Both make it easier to create a linguistic resource (descriptive grammar or implemented grammar), in terms of coverage of phenomena and in terms of compatibility with the relevant set of tools. At the same time, these aids can also have the adverse effect of limiting creativity or biasing analyses towards those anticipated by the creator of templates/libraries of analyses. Compared to prose templates, Grammar Matrix libraries are more difficult to create (represent a larger investment of time and effort) and are likely also more limiting. Both of these effects follow from the fact that implemented grammars require all of their component analyses to interact. On the one hand, the grammar engineers constructing the libraries must design them carefully to be interoperable with all of the options of all of the other libraries (Drellishak 2009:Ch. 2). On the other hand, a linguist attempting to develop an alternative analysis for one phenomenon will find herself hemmed in to a certain extent by the decisions made in the analyses of other phenomena.

Nonetheless, we believe that grammar engineering provides a net benefit for analysis exploration because computers can be harnessed to test the analyses against large data sets (Bender 2008b, Bender et al. 2011). Thus even though it can require non-trivial work to implement alternative analyses, their relative advantages and disadvantages can be empirically explored (Bender 2010). Recently, Fokkens (2011) has been investigating the potential of 'metagrammar engineering', or the development of systems that provide not only implemented analyses of varying phenomena, but in fact multiple analyses per variant. Fokkens argues that this will alleviate the risks of implemented grammars being shaped by the order in which phenomena are analyzed.

(11) COLLABORATION: If we value the potential for faster progress when multiple investigators collaborate, we should develop methodologies and tools which support collaboration.[6]

As will be discussed further in Section 5, grammar engineering for language documentation is an excellent example of the kind of project that thrives on collaboration, in this case between one or more field linguists and one or more grammar engineers. The grammar engineering work is dependent on the field work and cannot proceed without data collection, transcription and analysis done by the field linguist. At the same time, grammar implementation allows the hypotheses generated by both the field linguist and (eventually) the grammar engineer to be systematically tested against the collected data. Tools which would facilitate this kind of collaboration include those which help field linguists to produce consistent and well-formatted IGT, on the one hand, and those which make the resulting implemented grammar available for interactive inspection (such as web-interfaces to parsing and generation algorithms) on the other.

4.3 Grammar exploration

(12) EASE OF FINDING: If we value ease and speed of retrieving the information needed, a GD which has a table of contents, an index and full text search is preferable.

Fangorn is an extremely valuable tool for finding information within a treebank, provided that readers know how to formulate the queries they are interested in. We envision embedding preformulated search queries within the electronic prose grammar (as links that retrieve additional examples from the database, for example). An annotated index of these queries could be a useful source of information for a linguist seeking to formulate additional queries.

A second question is how to link back to the relevant parts of the grammar from the trees in the treebank. Ideally, each phrase structure rule, lexical rule and lexical type in the implemented grammar would be annotated with the phenomenon or phenomena that it implements. With such annotations, a reader could move from the tree assigned to a particular sentence to the relevant discussions in the prose grammar. The exact means of encoding these annotations is an issue for future work. However, we note here that linking to a particular prose descriptive grammar is probably a more tractable problem than producing a general index of a stand-alone implemented grammar.

[6] Nordhoff provides a different value statement under the heading collaboration: "We value collaboration and the recognition of the respective contributions of the collaborators" (p. 302). We do not disagree with that value statement, but find the one provided in (11) more relevant to the present discussion.

(13) INDIVIDUAL READING HABITS: If we value the individual linguist's decisions as to what research questions could be interesting (Rice 2006:402), a GD should permit the reader to follow his or her own path to explore it and a short path between two related phenomena is better.

(14) MANIPULATION: If we value portability and reusability of the data, the data presented in a GD should be easy to extract and manipulate.

The key issue regarding individual reading habits, according to Nordhoff (2008), is that some readers will be asking how a particular function is realized within a language, while others will be interested in the function(s) associated with a particular form. In this context, the fact that Redwoods-style treebanks (as described here) include semantic representations is a key asset. While at present, Fangorn only applies over syntactic structures, it can conceivably be extended to searches over semantic structures as well as combined syntactic/semantic queries.

Furthermore, if the implemented grammar is made available along with the treebank, it, too, becomes a tool for both form- and function-based exploration, greatly enhancing the ways that the data can be manipulated. For form-based exploration, the reader can send strings to the grammar for parsing.[7] For function-based exploration, the reader would want to use a generation algorithm (e.g., that included with the LKB (Carroll et al. 1999)). Generation algorithms that work with Grammar Matrix-derived grammars take as input Minimal Recursion Semantics representations (Copestake et al. 2005), which cannot easily be written by hand. However, the Grammar Matrix is compatible with the LOGON machine translation (MT) infrastructure (Lønning et al. 2004). While it would take additional work to produce an MT system (notably the writing of a transfer grammar), this could in principle be done. In that case, within the coverage of the MT system, readers could submit sentences in the other language of the MT pair and retrieve strings in the language being documented. The trees associated with those strings could then point into the descriptive grammar (as above).[8]

(15) FAMILIARITY: If we value ease of access, a GD that is similar to other GDs known to the reader is better.

Here we must acknowledge that treebanks and implemented grammars will not be immediately familiar to linguists on first encounter, and there is work to be done to make them more accessible. However, once a linguist has become familiar with one such resource, that familiarity should be readily transferable to another one constructed in a similar fashion. This once again underscores the importance of tools in promoting standardization (cf. (9)).

(16) GUIDING: If we value an informed presentation of the data, the GD should present the data in a didactically preferred way (Rice 2006:401).

[7]The grammar will return multiple analyses in many cases, but if a parse selection algorithm is trained on the treebank, those analyses can be ranked by their predicted likelihood.

[8]Note that for QUALITY ASSESSMENT (19) and ACCOUNTABILITY (5), strings returned by the generator would need to be flagged according to whether they match strings in the collected data or in fact represent generalizations based on the hypotheses encoded in the grammar.

(17) EASE OF EXHAUSTIVE PERCEPTION: If we value the quest for comprehensive knowledge of a language (Cristofaro 2006:162), the readers should be able to know that they have read every page of the grammar.

Implemented grammars are intricate objects (full of interconnections) and the field of grammar engineering is still struggling with developing best practices for documenting them. It is unlikely that a treebank or implemented grammar would assist in the ordering of information or the creation of paths through a prose descriptive grammar or with ease of exhaustive perception. However, by embedding links to Fangorn in that grammar, the prose descriptive grammar could become a very effective guide to the implemented grammar (to the extent that the analyses in the implemented grammar all directly map to analyses in the prose grammar).

(18) RELATIVE IMPORTANCE: If we value the allocation of scarce resources of time to primary areas of interest, the relative importance of a phenomenon for (a) the language and (b) language typology should be retrievable (Zaefferer 1998, 2, Noonan 2006, 355).

There are of course many different ways of defining importance of phenomena. If one takes a quantitative approach, a treebank can be a useful tool in exploring such things. Assuming we have an index linking grammar rules and lexical entries to phenomena described in the prose grammar, it should be possible to quantify the text frequency of each phenomenon. Likewise, the number of rules/entries in the grammar which are indexed to the phenomenon would give a sense of the degree to which that phenomenon interacts with others.

Regarding cross-linguistic relevance, in the long term, the Grammar Matrix project has the potential to provide this information: If there are many grammars constructed on the basis of the Grammar Matrix and its libraries (the "customization system"), we will be able to quantify the relative prevalence of each choice in each library, as well as co-occurrence tendencies between the choices. In addition, it is in principle possible to detect whether specific analyses in an implemented grammar remain consistent with the starting point provided by the Grammar Matrix or required changes.[9]

(19) QUALITY ASSESSMENT: If we value indication of the reliability of analyses, the quality of a linguistic description should be indicated.

The development of an implemented grammar is a fairly stringent test of the quality of linguistic analyses. It is not of course the case that implemented analyses are necessarily correct. However, with implemented analyses it is possible to tell whether analyses of multiple phenomena are consistent with each other and also the extent to which the analyses collectively account for the available data (cf. Good's concept of *internal coverage* (this volume)). That is, the quality of a treebank and its underlying implemented grammar can be partially assessed in terms of the number of examples in the underlying database which are assigned a tree.

In order for this assessment to reflect on the prose descriptive grammar which is the basis of the implemented grammar, at least two points need to be made explicit: (1) the extent to which the

[9]Of course, there is also always the influence of the grammar engineer (cf. CREATIVITY (10) above): a grammar could differ from the analyses provided by the Grammar Matrix because those analyses did not work for the language in question, or because the grammar engineer chose to explore alternatives.

analyses in the implemented grammar are faithful to the descriptive grammar, and (2) the extent to which the implemented grammar incorporates all of the analyses in the descriptive grammar. That is, the descriptive grammar could be very comprehensive, but if the implemented grammar does not include analyses for every phenomenon treated in the descriptive grammar, treebank coverage will be poor.

Using treebanks to assess the quality of individual analyses is somewhat more problematic. The same indexing of implemented grammars that was suggested for measuring the relevance or importance of particular phenomena could also be used to estimate the success of their analysis in implemented grammar. However, these two factors are confounded: A highly central or important phenomenon with a poor analysis would appear to be relatively unimportant, since the poor analysis could lead to poor coverage for the sentences with the phenomenon. Thus what is called for is an independent way to measure the frequency of phenomena (perhaps based on interlinear glossed text alone) and then compare that to the measurements taken over the treebank.

So far this brief discussion has considered grammar/treebank quality only in terms of coverage, or the ability to find a correct analysis for any given example. Another important measure of grammar quality, however, is ambiguity: Grammars or analyses which are underconstrained will produce many spurious analyses. These will not be apparent in the final treebank, as they are discarded in the manual annotation step. However, the maxim in (19) suggests that the degree of ambiguity found by the grammar underlying the treebank should be reported. In addition, it is straightforward to quantify the extent to which particular rules and lexical entries contribute to ambiguity.[10]

Finally, we note that in the development of treebanks for descriptive grammars, the correctness of a tree is determined by comparing the semantic representation associated with that tree to the translation and gloss provided for the example. Thus to the extent that quality issues in the descriptive grammar affect the quality of the glossing, these issues will be masked in measures involving the treebank.

(20) PERSISTENCE: If we value citability (Bird and Simons 2003:14), in order to facilitate longterm reference, a grammatical description should not change over time.

As Bird & Simons and Nordhoff note, the solution to the conflict between this maxim and the one in (7) is to take snapshots which can be the anchors for citations. The addition of treebanks to electronic descriptive grammars makes this somewhat more difficult as the versions between the treebank (and implemented grammar) on the one hand and the prose descriptive grammar on the other need to be synchronized. This is perfectly possible, however, with proper planning.

(21) TANGIBILITY: If we value the appreciation of a grammatical description as a comprehensive aesthetic achievement, a GD that can be held in the hand is better.

Implemented grammars and treebanks are only valuable as computational artifacts, and thus will only be the sort of thing that can be held in the hand when hand-held devices are powerful

[10]Note, however, if a particular rule is prone to adding ambiguity, that may not be the fault of the analyses of the phenomena it currently implements (and thus is indexed for) but rather the analysis of some other phenomenon which should involve constraints on that rule but does not.

enough to run them. That said, the addition of an implemented grammar should not get in the way of the production of the associated descriptive grammar as book. In practical terms, any links to treebank searches that are embedded in prose chapters should be either stripped or made non-disruptive. In addition, any links in a printed volume must be stable links that will continue to function as long as possible.

(22) MULTILINGUALIZATION: If we value the interest of every human in a given language, especially interest from the speakers of the language in question, a GD should be available in several languages, among others the language of wider communication in the region where the language is spoken (Weber 2006:433).

There are two primary ways in which implemented grammars and treebanks can respond to this maxim. The first is to be designed to be able to incorporate and display glossing of the primary data into multiple different languages. The second (and longer term) method is through the machine translation possibilities discussed in reference to the values INDIVIDUAL READING HABITS (13) and MANIPULATION (14) above. It is possible in principle to set up multiple MT systems between a language being described and different languages of wider communication. Furthermore, while there is additional work required for every language pair, each additional language pair should require less set up work than the first.

4.4 Summary

Our purpose in this discussion has been twofold: On the one hand, by reflecting on values and maxims, we have proposed a series of design desiderata for the incorporation of treebanks in electronic descriptive grammars. On the other hand, we hope to have provided arguments in favor of the value of treebanks and implemented grammars to the enterprise of language documentation and description, and clarified the role that they can play.

The incorporation of treebanks into descriptive grammars is possible on the basis of existing technology, including technology supporting grammar development, parsing, generation, treebank creation and maintenance and treebank search. This is sketched in the following section. The preceding discussion makes it clear that achieving the full potential of the integration will rely on further advances in a few areas. These include: methodologies and software support for indexing components of implemented grammars, support for rapid deployment of machine translation, and user-interface improvements to make implemented grammars and the analyses they assign to strings more accessible to non-grammar engineer linguists.

5 Getting there

In the previous sections, we have presented the idea of augmenting electronic descriptive grammars with treebanks and reflected on how doing so will help descriptive grammars fulfill the values that have been articulated for them. This section addresses the feasibility of creating implemented grammars and treebanks and discusses the resources that are available to assist in the creation of such resources as well as future directions.

5.1 The Grammar Matrix

Building implemented grammars can be expensive and time-intensive. The English Resource Grammar (ERG) has been under development since 1994, and now achieves 62-94% verified coverage over naturally occurring corpora from a variety of genres (Flickinger 2011).[11] The example of the ERG shows that this kind of grammar and treebank construction is indeed possible, but it also suggests that it might be too expensive to be applied in context of language documentation, as envisioned here.

We contend that it is not, for several reasons. First, a grammar does not have to be comprehensive, or even approach the ERG's level of coverage, in order to be useful. Even a partial treebank would begin to yield benefits for linguists searching for examples (though it will be important to make clear the extent to which the treebank covers the total corpus, cf. QUALITY ASSESSMENT (19) above). Secondly, it is unfortunately the case that language documentation projects rarely approach a range of genre diversity in the data collected that compares to the range of genres the ERG has been tested against. A smaller genre range means a more tractable problem for grammar engineering. Finally, much of the effort that has gone into the ERG represents solutions to problems that are not in fact English-specific, but more general contributions to efficient, implemented HPSG parsing.

This last point is the motivation for the LinGO Grammar Matrix (Bender et al. 2002, Bender et al. 2010). Specifically, the Grammar Matrix aims to facilitate the development of implemented grammars in languages without such resources by curating and making available advances from the ERG and other broad-coverage grammars developed in the DELPH-IN[12] consortium, most notably the Jacy Japanese grammar (Siegel and Bender 2002) and the German grammar (Müller and Kasper 2000). The Grammar Matrix consists of a core grammar, shared by all language-specific grammars derived from it, a series of libraries of analyses of cross-linguistically variable phenomena, and a "customization system" which allows users to select from among those analyses by filling in a web-based questionnaire.

The core grammar includes definitions (constraints) that are hypothesized to be cross-linguistically useful. The customization system pairs these with more specialized constraints on the basis of information collected through the questionnaire. The questionnaire elicits from a linguist-user typological information of varying granularity: for example, major constituent word order (including various flexible word order options), the means of expression of negation, the range of cases (if any) marked on core arguments, the possibility of dropping each core argument and information about its interpretation when dropped. It also allows users to define classes of lexical items and morphological rules. Morphological rule definitions include morpheme forms,[13] morphosyntactic and morphosemantic features (case, tense, etc.) associated with the morphemes, and ordering and co-occurrence constraints with other morphemes (including stems).

[11] The low end of that spectrum relates to fairly technical corpora, including technical manuals and chemistry papers. Verified coverage over 80% is more typical.

[12] `http://www.delph-in.net/`

[13] These are expected to be regularized underlying forms. We follow Bender and Good (2005) in advocating for separate components for morphophonological and morphosyntactic analysis. Various tools exist for creating morphophonological analyzers, including XFST (Beesley and Karttunen 2003).

This strategy of code reuse necessarily involves taking analyses developed on the basis of well-studied languages and applying them to lesser-studied languages. However, the Grammar Matrix project is explicitly data-oriented in its approach to cross-linguistic universals and cross-linguistic variation. With regards to the constraints provided in the core grammar, these are treated as working hypotheses only, ready to be revised (or more precisely made variable and moved into the libraries) when we encounter languages which present counter-examples. The methodology of grammar engineering allows us to empirically test the applicability of analyses and determine when an analysis really won't work. Furthermore, our library development methodology begins with a review of the typological literature so that we are working with the most comprehensive possible view of the range of variation in the world's languages as we develop the libraries of analyses.

The work on Wambaya cited above (Bender 2008a) provides a case-study in the feasibility of grammar engineering for language documentation. The grammar produced is approximately an order of magnitude less complex than the English Resource Grammar. Nonetheless, it provided interesting coverage over both the exemplars cited in Nordlinger (1998) and a naturally occurring text used to test the grammar's ability to generalize beyond the data used in grammar development. It was possible to achieve this level of coverage so rapidly thanks in part to the restricted range of data being considered (relatively short sentences, relatively little genre variation) but more importantly thanks to the analytical work done by Nordlinger. It is in some ways easier to implement analyses presented in a descriptive grammar than to work from intuitions about one's own language combined with analyses gleaned from the theoretical linguistic literature. In other words: the original descriptive work (done in this case by Nordlinger) is the hard part. When this is done thoroughly and done well, the grammar engineering is relatively straightforward.

5.2 Treebanking Support

Once an initial version of an implemented grammar has been written, building an *undisambiguated* treebank is an automatic process that can be easily initiated using the suite of tools available. Treebanking, the process of selecting the most plausible tree using discriminants (see Section 2.2), requires further effort, but much less than manually annotating the data. While treebanking will always require a human annotator if we wish to maintain quality, there is some work on automatic methods to help make treebanking easier. One method that has proved successful in previous experiments is to rank the discriminants so as to present those most likely for an annotator to select at the top of the list. Zhang and Kordoni (2010) showed that treebanking speed could be improved by using the annotation logs of their treebankers to build a statistical model that ranked the discriminants in the order which an individual treebanker would be likely to select them. Another method, called blazing (Tanaka et al. 2005, MacKinlay et al. 2011), uses supplementary information available about the text to partially pre-annotate. In this work, the authors used pre-existing annotations of parts of speech (in the case of Tanaka et al. 2005) or phrase structure trees (in the case of MacKinlay et al. 2011) to automatically mark some discriminants, leaving the annotator less decisions to make, and also found increases in treebanking speed. Rather than requiring external information, a third strategy would be to use all the analyses produced for all items to learn trends in the analyses. While this information is noisy, the trends from the less ambiguous items

inform the decisions to be made for the more ambiguous items and this partial information can be used to automatically learn a probabilistic ranking function to rank all analyses. In this way, we can prune improbable analyses and in the process accelerate treebanking (Dridan and Baldwin 2010).

5.3 Future Work on Fangorn

At present, Fangorn has a query language that is sufficient to express simple queries using paths and filter expressions. However, for more complicated queries with multiple logical operators in a filter expression, allowing braces to group terms would make queries more expressive. For example, let us consider query (3) and add further conditions that require the search to match only those parses where the complement of the verb is realized only by a modifier and not by a nominal head. We would now have to exclude trees which have a HEAD-COMP-2 or COMP-HEAD-2 label underneath the declarative clause. Query (3) would have to be reframed as shown in (23). If the grouping of elements, using braces, were allowed we could rewrite (23) as (24), which is not only more concise, but is also easier to read.

(23) //DECL[//HEAD-COMP-MOD-2 AND NOT //HEAD-COMP-2 AND NOT //COMP-HEAD-2 OR //COMP-HEAD-MOD-2 AND NOT //HEAD-COMP-2 AND NOT //COMP-HEAD-2]

(24) //DECL[(//HEAD-COMP-MOD-2 OR //COMP-HEAD-MOD-2) AND NOT (//HEAD-COMP-2 OR //COMP-HEAD-2)]

Another potential means of grouping terms within a filter expression would be to take advantage of the types declared in the grammar behind the treebank, which group together sets of rules. For instance, there are grammar types in Wambaya grammar, shown in (25) (cf. Figure 3), that match the two elements of the filter expression in (24). Replacing elements in (24) with their grammar type would allow us to further refine the query to (26). The search tool could itself perform the substitution of grammar types with actual labels rather than expect the user to input expanded queries. For this to work, Fangorn has to be aware of grammar types and expand them prior to execution.

(25) `head-2nd-comp-mod-phrase: { head-comp-mod-2, comp-head-mod-2 }`
`wmb-head-2nd-comp-phrase: { head-comp-2, comp-head-2 }`

(26) //DECL[(//HEAD-2ND-COMP-MOD-PHRASE AND NOT //WMB-HEAD-COMP-2ND-COMP-PHRASE]

The current version of Fangorn operates over the abbreviated tree structures that are used for presentation purposes. The abbreviated trees are sufficient to distinguish between competing analyses, but they don't expose all the information that a user might wish to search for. As mentioned in Section 4.3, the semantic information embedded in the tree would provide a useful mode of querying. This provides some interesting challenges in determining the best way to represent semantic information so that it can be queried, since a tree structure is not a natural representation for semantics. We are currently pursuing two different approaches to this problem: either trying

to find a natural way to represent the semantic information we have in a treelike form, or alternatively, looking for an intuitive extension of the query language that would allow querying over a more appropriate representation.

Being able to query the syntax and semantics separately provides different views and avenues of access to the same data. Likewise, other levels of annotation that exist, such as glosses and translations, could be useful in finding the examples that a user requires. A simple extension to Fangorn is planned to allow different annotation levels to be aligned, so that it is possible to search using one representation and see how the same data is analyzed at a different level. A more complex extension would allow a query to filter using multiple levels of annotation, for example using semantic restrictions to filter a syntactic query. This extension could require extensive changes to the query language for a fully general solution, but it might be possible to achieve most of the desired capabilities by designing a means of specifying metadata about annotations both within the treebank and in the query language.

Another area of future work for Fangorn is the mapping of labels onto a cross-linguistic label set, e.g. based on GOLD (Farrar and Lewis 2007). This would involve aligning individual grammar rules, lexical rules and lexical types onto the GOLD ontology to mark features such as verb transitivity, noun case and clause illocutionary force, while preserving the language-specific rule types and/or more familiar node labels (e.g., NP and VP) as are currently used. This would significantly enhance cross-linguistic treebank search, as the label set would be harmonized to a much greater extent than occurs using the "native" label set for individual grammars.

5.4 Virtuous Cycles and the Montage Vision

The Wambaya case study described above was an exercise in post-hoc grammar engineering: The implemented grammar wasn't developed until a decade after the original field work was complete, and sadly, the language lost its last fully fluent speakers in that time. The process of grammar engineering always raises further questions about the data (as no grammatical description is ever complete), and the Wambaya case study suggests that collaborations between grammar engineers and field linguists could be very fruitful:[14] While a considerable amount of data collection and analysis has to take place before grammar engineering can get off the ground, if the field linguist is still working with speakers when the grammar implementation work begins, there is the potential for a feedback loop that speeds up and strengthens the descriptive work.

The Montage project (Bender et al. 2004) envisioned a software environment which integrated tools for the production of IGT, electronic descriptive grammars and implemented grammars. The IGT and the descriptive grammar would inform the implemented grammar, and even possibly be input to a system that could automatically create a partial implemented grammar. The implemented grammar would in turn feed IGT and descriptive grammar development by locating interesting exemplars (through Fangorn[15]), highlighting possible inconsistencies in glossing, and testing out

[14]We would like to emphasize that nothing in the preceding discussion requires that the grammar engineering and field work be done by the same person, and in fact it seems unlikely that many people would have the skill sets required for both. Going further, it seems like grammar engineering would be a less than efficient use of the time of someone who has the skills to do original fieldwork.

[15]Note that this could even be done before the manual annotation step of the treebank construction process: the

analyses.

The Montage project itself was never funded, but nonetheless there is progress in the direction of this vision, including:

- Collaborative annotation and descriptive grammar authoring environments, including GALOES (Nordhoff 2007), TypeCraft (Beermann and Mihaylov 2009) and Digital Grammar (Drude, this volume)
- The Grammar Matrix customization system (Bender et al. 2010, cf. Section 5.1)
- The Redwoods methodology for dynamic treebank construction (Oepen et al. 2004, cf. Section 2.2)
- Treebank search using Fangorn (Ghodke and Bird 2010, cf. Section 3.1)
- Machine learning algorithms that learn typological properties from IGT (e.g., Lewis and Xia 2007)

The Grammar Matrix makes it feasible to create interesting implemented grammars for languages without large computational resources, while the Redwoods methodology makes treebank development practical. Fangorn makes the treebanks useful as resources for readers of descriptive grammars. The longer term goal of semi-automatic grammar implementation is supported by the Grammar Matrix and the work of Lewis and Xia (2007) which suggests that it might be possible to learn answers to questions like those in the Grammar Matrix questionnaire on the basis of (sufficiently large) sets of IGT (with sufficiently meticulous glossing).

6 Conclusion

In this paper we have presented a vision of how electronic descriptive grammars can be enriched with implemented grammars and treebanks, and described how such a vision is supported by current technology as well as what future developments could add further value. Following the discursive approach of Bird and Simons (2003) and Nordhoff (2008), we have explored the ways in which implemented grammars and treebanks can help to meet the values and associated maxims proposed regarding producing the most useful possible language resources. To our knowledge, no such treebank-enhanced descriptive grammar yet exists, but we hope to see them emerge through the collaboration of field linguists and grammar engineers.

References

Abney, Steven. 1996. Statistical methods and linguistics. In J. L. Klavans and P. Resnik (Eds.), *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*. Cambridge, MA: MIT Press.

treebank search tools described above work equally with undisambiguated sets of trees.

Bagish, Henry. 1983. Confessions of a former cultural relativist. In E. Angeloni (Ed.), *Anthropology Annual Editions 83/84*, 87–112. Guilford, CT: Dushkin Publishing Group.

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*. Berlin, Germany: Mouton de Gruyter.

Beermann, Dorothee, and Pavel Mihaylov. 2009. TypeCraft: Linguistic data and knowledge sharing, open access and linguistic methodology. Unpublished ms., Paper presented at the Workshop on Small Tools in Cross-linguistic Research, University of Utrecht. The Netherlands.

Beesley, Kenneth R., and Lauri Karttunen. 2003. *Finite State Morphology*. Stanford CA: CSLI Publications.

Bender, Emily M. 2008a. Evaluating a crosslinguistic grammar resource: A case study of Wambaya. In *Proceedings of ACL08:HLT*, 977–985, Columbus, OH.

Bender, Emily M. 2008b. Grammar engineering for linguistic hypothesis testing. In N. Gaylord, A. Palmer, and E. Ponvert (Eds.), *Proceedings of the Texas Linguistics Society X Conference: Computational Linguistics for Less-Studied Languages*, 16–36, Stanford, CA. CSLI Publications.

Bender, Emily M. 2010. Reweaving a grammar for Wambaya: A case study in grammar engineering for linguistic hypothesis testing. *Linguistic Issues in Language Technology* 3:1–34.

Bender, Emily M., Scott Drellishak, Antske Fokkens, Laurie Poulson, and Safiyyah Saleem. 2010. Grammar customization. *Research on Language & Computation* 8(1):1–50.

Bender, Emily M., Dan Flickinger, Jeff Good, and Ivan A. Sag. 2004. Montage: Leveraging advances in grammar engineering, linguistic ontologies, and mark-up for the documentation of underdescribed languages. In *Proceedings of the Workshop on First Steps for Language Documentation of Minority Languages: Computational Linguistic Tools for Morphology, Lexicon and Corpus Compilation, LREC 2004*, Lisbon, Portugal.

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 *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, 8–14, Taipei, Taiwan.

Bender, Emily M., Dan Flickinger, and Stephan Oepen. 2011. Grammar engineering and linguistic hypothesis testing: Computational support for complexity in syntactic analysis. In E. M. Bender and J. E. Arnold (Eds.), *Language from a Cognitive Perspective: Grammar, Usage and Processing*, 5–29. Stanford, CA: CSLI Publications.

Bender, Emily M., and Jeff Good. 2005. Implementation for discovery: A bipartite lexicon to support morphological and syntactic analysis. In *Proceedings from the Panels of the Forty-First Meeting of the Chicago Linguistic Society: Volume 41-2.*

Bierwisch, Manfred. 1963. *Grammatik des deutschen Verbs*. Vol. II of *Studia Grammatica*. Akademie Verlag.

Bird, Steven, Yi Chen, Susan B. Davidson, Haejoong Lee, and Yifeng Zheng. 2006. Designing and evaluating an XPath dialect for linguistic queries. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, 52–62, Washington, DC.

Bird, Steven, and Gary Simons. 2003. Seven dimensions of portability for language documentation and description. *Language* 79(3):557–582.

Black, H. Andrew, and Gary F. Simons. 2008. The SIL FieldWorks Language Explorer approach to morphological parsing. In N. Gaylord, A. Palmer, and E. Ponvert (Eds.), *Proceedings of the Texas Linguistics Society X Conference: Computational Linguistics for Less-Studied Languages*, 37–55, Stanford, CA. CSLI Publications.

Carroll, John, Ann Copestake, Daniel Flickinger, and Victor Poznanski. 1999. An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation*, 86–95, Toulouse, France.

Carter, David. 1997. The treebanker: a tool for supervised training of parsed corpora. In *Proceedings of a Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, 9–15, Madrid, Spain.

Copestake, Ann. 2000. Appendix: Definitions of typed feature structures. *Natural Language Engineering* 6:109–112.

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(4):281–332.

Cristofaro, Sonia. 2006. The organization of reference grammars: A typologist user's point of view. In F. Ameka, A. Dench, and N. Evans (Eds.), *Catching Language: The Standing Challenge of Grammar Writing*, 137–170. Berlin, Germany: Mouton de Gruyter.

Crouch, Dick, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell, and Paula Newman. 2001. XLE documentation. Unpublished ms., On-line documentation, Palo Alto Research Center (PARC).

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

Dridan, Rebecca, and Timothy Baldwin. 2010. Unsupervised parse selection for HPSG. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, 694–704, Boston, USA.

Drude, Sebastian. this volume. Digital grammars — integrating the Wiki/CMS approach with language archiving technology and TEI. In S. Nordhoff (Ed.), *Electronic Grammaticography*. Honolulu: University of Hawai'i Press.

Farrar, Scott, and William D. Lewis. 2007. The GOLD community of practice: An infrastructure for linguistic data on the web. *Language Resources and Evaluation* 41(1):45–60.

Flickinger, Dan. 2011. Accuracy v. robustness in grammar engineering. In E. M. Bender and J. E. Arnold (Eds.), *Language from a Cognitive Perspective: Grammar, Usage and Processing*, 31–50. Stanford, CA: CSLI Publications.

Fokkens, Antske. 2011. Metagrammar engineering: Towards systematic exploration of implemented grammars. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 1066–1076, Portland, OR.

Ghodke, Sumukh, and Steven Bird. 2010. Fast query for large treebanks. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 267–275, Los Angeles, CA.

Good, Jeff. 2004. The descriptive grammar as a (meta)database. In *Proceedings of the E-MELD Workshop 2004: Linguistic Databases and Best Practice*, Detroit, Michigan.

Good, Jeff. this volume. Deconstructing descriptive grammars. In S. Nordhoff (Ed.), *Electronic Grammaticography*. Honolulu: University of Hawai'i Press.

Hashimoto, Chikara, Francis Bond, Takaaki Tanaka, and Melanie Siegel. 2007. Semi-automatic documentation of an implemented linguistic grammar augmented with a treebank. *Language Resources and Evaluation (Special Issue on Asian Language Technology)* 42(2):117–126.

Johnson, Mark, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Annual Meeting of the ACL*, 535–541, College Park, MD.

Kiefer, Bernd, Hans-Ulrich Krieger, John Carroll, and Rob Malouf. 1999. A bag of useful techniques for efficient and robust parsing. In *Proceedings of the 37th Annual Meeting of the ACL*, 473–480, College Park, MD.

Lai, Catherine, and Steven Bird. 2004. Querying and updating treebanks: A critical survey and requirements analysis. In *Proceedings of the Australasian Language Technology Workshop*, 139–146, Sydney, Australia.

Lewis, William D., and Fei Xia. 2007. Automatically identifying computationally relevant typological features. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, 685–690, Hyderabad, India.

Lezius, Wolfgang, and Esther König. 2000. Towards a search engine for syntactically annotated corpora. In *KONVENS 2000 / Sprachkommunikation, Vorträge der gemeinsamen Veranstaltung 5. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS), 6. ITG-Fachtagung "Sprachkommunikation"*, 113–116, Berlin, Germany.

Lønning, Jan Tore, Stephan Oepen, Dorothee Beermann, Lars Hellan, John Carroll, Helge Dyvik, Dan Flickinger, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, Victoria Rosén, and Erik Velldal. 2004. LOGON. A Norwegian MT effort. In *Proceedings of the Workshop in Recent Advances in Scandinavian Machine Translation*, Uppsala, Sweden.

MacKinlay, Andrew, Timothy Baldwin, Dan Flickinger, and Rebecca Dridan. 2011. Using external treebanks to filter parse forests for parse selection and treebanking. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011)*, 246–254, Chiang Mai, Thailand.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

Meurers, W. Detmar, Gerald Penn, and Frank Richter. 2002. A web-based instructional platform for constraint-based grammar formalisms and parsing. In *Proceedings of the ACL 2002 workshop on Effective Tools and Methodologies for Teaching NLP and CL*, 18–25, Philadelphia, PA.

Müller, Stefan. 1999. *Deutsche Syntax deklarativ: Head-Driven Phrase Structure Grammar für das Deutsche*. Tübingen, Germany: Max Niemeyer Verlag.

Müller, Stefan, and Walter Kasper. 2000. HPSG analysis of German. In W. Wahlster (Ed.), *Verbmobil: Foundations of Speech-to-Speech Translation*, 238–253. Berlin, Germany: Springer.

Noonan, Michael. 2006. Grammar writing for a grammar-reading audience. *Studies in Language* 30:351–365.

Nordhoff, Sebastian. 2007. Growing a grammar with galoes. Unpublished ms., Paper presented at the DoBeS workshop.

Nordhoff, Sebastian. 2008. Electronic reference grammars for typology: Challenges and solutions. *Language Documentation & Conservation* 2:296–324.

Nordlinger, Rachel. 1998. *A Grammar of Wambaya, Northern Australia*. Canberra: Research School of Pacific and Asian Studies, The Australian National University.

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, and Daniel P. Flickinger. 1998. Towards systematic grammar profiling. Test suite technology ten years after. *Journal of Computer Speech and Language* 12 (4) (Special Issue on Evaluation):411–436.

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.

Resnik, Philip, Aaron Elkiss, Ellen Lau, and Heather Taylor. 2005. The web in theoretical linguistics research: Two case studies using the linguist's search engine. In *31st Meeting of the Berkeley Linguistics Society*, 265–276, Berkeley, USA.

Rice, Keren. 2006. A typology of good grammars. *Studies in Language* 30:385–415.

Riezler, Stefan, Tracy Holloway King, Richard S. Crouch, John T Maxwell, and Ronald M. Kaplan. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the ACL and 3rd Annual Meeting of the NAACL (ACL-02)*, 7–12, Philadelphia, PA.

Rohde, Douglas L. T. 2005. *TGrep2 User Manual Version 1.15*. `http://tedlab.mit.edu/~dr/TGrep2/tgrep2.pdf`.

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.

Tanaka, Takaaki, Francis Bond, Stephan Oepen, and Sanae Fujita. 2005. High precision treebanking: blazing useful trees using POS information. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 330–337, Ann Arbor, MI.

Toutanova, Kristina, Chistopher D. Manning, Dan Flickinger, and Stephan Oepen. 2005. Stochastic HPSG parse selection using the Redwoods corpus. *Journal of Research on Language and Computation* 3(1):83–105.

Tukey, John W. 1977. *Exploratory Data Analysis*. Reading, MA: Addison-Wesley.

Weber, David. 2006. Thoughts on growing a grammar. *Studies in Language* 30:417–444.

Zaefferer, Dietmar. 1998. Einleitung: Allgemeine Vergleichbarkeit als Herausforderung für die Sprachbeschreibung. In D. Zaefferer (Ed.), *Deskriptive Grammatik und Allgemeiner Sprachvergleich*, 1–5. Tübingen, Germany: Niemeyer.

Zhang, Yi, and Valia Kordoni. 2010. Discriminant ranking for efficient treebanking. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, 1453–1461, Beijing, China.