An open source grammar development environment and broad-coverage English grammar using HPSG

Ann Copestake*, **Dan Flickinger***

*CSLI, Ventura Hall, Stanford University
Stanford, CA 94305-4115, USA
{aac, danf}@csli.stanford.edu

Abstract

The LinGO (Linguistic Grammars Online) project's English Resource Grammar and the LKB grammar development environment are language resources which are freely available for download for any purpose, including commercial use (see http://lingo.stanford.edu). Executable programs and source code are both included. In this paper, we give an outline of the LinGO English grammar and LKB system, and discuss the ways in which they are currently being used. The grammar and processing system can be used independently or combined to give a central component which can be exploited in a variety of ways. Our intention in writing this paper is to encourage more people to use the technology, which supports collaborative development on many levels.

1. Introduction

Despite the advances that have been made in automatic grammar learning, there are many applications which require hand-built or partially hand-built grammars for acceptable precision. However, developing precise large-coverage grammars and lexicons of natural languages is a very time-consuming activity. In this paper we give an overview of the LinGO (Linguistic Grammars Online) project's English Resource Grammar (ERG) and the LKB grammar development environment which are freely available resources for use in research, teaching and commercial applications. The LKB is a general-purpose system that can process the ERG and other grammars written in a typed-feature structure formalism. The ERG is a broad-coverage grammar of English in the Head-Driven Phrase Structure Grammar (HPSG) framework which can be used in a range of applications. Besides the ERG, some smaller grammars intended for teaching purposes are also distributed. We discuss the steps we are taking to make the ERG and LinGO processing system as generally usable as possible, and describe some of the collaborative work which has been undertaken.

There are a number of properties of the LinGO resources which, taken in combination, make them unique in terms of currently available NLP systems:

Availability The LKB system and the LinGO grammars are freely available on the Web as open source (http://lingo.stanford.edu): they may be used and modified by both academic and commercial groups. The LKB system is implemented in Common Lisp,[1] but is also distributed as a standalone application for Linux, Windows 98/NT and Sun Solaris platforms, and can be installed and used by people with no specialist programming knowledge.

Explicit formalization and declarativity The formalism assumed by the LKB system (Copestake, in press) is a typed feature structure logic similar to (Carpenter, 1992). Because the LinGO ERG has been developed using a core subset of this logic and is completely declarative, it can be processed on a number of systems (see §2.2. and §4. for more details). The LKB system necessarily assumes a particular formalism, but it follows the philosophy of PATR (Shieber, 1986) in aiming to be independent of a particular linguistic framework. Although the LKB's most extensive use has been for HPSGs, it has also been used for quite large categorial grammars (Sanfilippo, 1993).

Bidirectionality The LKB system includes a parser and a generator and the LinGO ERG itself is bidirectional.

Linguistic motivation The ERG uses the HPSG framework, which is quite popular among theoretical linguists and for which numerous detailed analyses of phenomena in many different languages exist. As we discuss in §3.2., this is an important factor in successful collaborative grammar development.

Scale and Efficiency The LinGO ERG is a broad coverage grammar (see §2.2. for details). However, at least in comparison with most previous typed feature structure systems, processing is highly efficient (see §2.2. for some details).

Semantic representation The Minimal Recursion Semantics (MRS) approach used in the ERG (Copestake et al., 1999) is fully supported by a module associated with the LKB system (see 3.3.).

Evaluation tools The LKB system can be used in conjunction with the [incr tsdb()] test-suite machinery (Oepen et al., 1997; Oepen and Flickinger, 1998) which provides extensive fine-grained diagnosis and profiling capabilities (see 3.1.).

The LinGO resources are somewhat comparable in scope to systems developed by companies such as SRI (Alshawi, 1992) and Xerox (the Xerox Linguistics Environment, XLE, developed at PARC), which are not generally available. Although there are several constraint-based

[1] Specifically, the current version runs on Macintosh Common Lisp, Allegro and Lispworks and on CMU Common Lisp without graphics.

grammar development systems for which source is freely available, of which ALE (Carpenter and Penn, 1999) has probably the widest distribution, the grammars distributed with them are relatively small compared to the LinGO ERG. The XTAG grammar (The XTAG Research Group, 1995) is of more comparable size to the LinGO ERG, but does not include semantics and the XTAG parser is "not for the casual user" according to the XTAG website. The LinGO technology is perhaps more similar to the Alvey Natural Language Tools (ANLT) (Briscoe et al., 1987), which has been widely used in NLP. However a fee is charged for the ANLT, the formalism is not as close to a currently popular linguistic framework, and no generator is available. Finally, a distinctive feature of the LinGO effort is the unusual degree of cross-center and international collaboration (see the acknowledgments in §6.). In §3. we will discuss various issues that arise when we attempt to facilitate collaborative development and common use of the resources, but first we will give more details of their current state.

2. The LinGO resources

2.1. The LKB system: efficient technology for grammar development

The LKB system was originally developed at Cambridge University as part of the EU ACQUILEX projects as a way of representing and validating highly structured lexical information (Copestake, 1992). At this point, LKB stood for Lexical Knowledge Base, and the LKB system was a tool for construction of such knowledge bases. As such, efficiency of parsing large grammars was not a primary consideration. When we decided to utilize the LKB system at CSLI as a more general tool for grammar development, several core components were reimplemented and efficiency greatly improved (Malouf et al., 2000; Oepen and Carroll, 2000).

A primary consideration in our efficiency work has been to avoid any techniques which require lengthy grammar compilation. Reasonable efficiency is essential for grammar development systems, since it must be possible to quickly validate changes made to the grammar on extensive test suites (see 3.1.). But it is crucial that the edit-test-debug cycle be kept as short as possible, which implies that the time required to load and reload a grammar must be short. The ERG currently loads in about 20 seconds with a cached lexicon, with about another 30 seconds being necessary if the database used for the lexicon has to be constructed from scratch.[2] Some indication of parsing times can be gathered from Table 1.

The functionality of the LKB system has also been extended in several ways since its use on ACQUILEX, most notably:

1. semantic processing using MRS (Copestake et al., 1999), see §3.3.
2. addition of an efficient tactical generator (Carroll et al., 1999),
3. integration with the [incr tsdb()] test-suite machinery, discussed briefly in 3.1.
4. order independent default unification (Lascarides and Copestake, 1999; Malouf, 1999)

In addition, the LKB system incorporates extensive visualization tools and various debugging aids in an integrated graphical user interface, which can be customized for users with different levels of experience. A small number of user-settable parameters allow for different styles of grammar.

2.2. The LinGO English grammar

The LinGO ERG was originally developed and still runs on the PAGE system (Kiefer et al., 1999). The central part of the grammar is a type hierarchy, which is used to structure the lexicon and the lexical and grammar rules. The grammar currently contains over 15,000 lines of code (excluding the lexicon, the hand-built portion of which is roughly 30,000 lines) and has involved a total of around eleven person-years of effort. Table 1 gives some indication of coverage. We have included some sentences that the grammar will not accept or generate, since the grammar is intended for applications where precision is required. The particular examples here are all taken from the Verbmobil test corpus (see 2.3.).

The hand-built core lexicon is supplemented by entries automatically constructed from COMLEX information (Grishman et al., 1994). Because the ERG lexicon is structured by means of a detailed type hierarchy, such a mapping is reasonably easy to construct. However, one major source of problems is that COMLEX does not provide subcategorization information for adjectives and nouns that is adequate for the ERG. We are currently tackling this with a mixture of hand-coding and semi-automatic acquisition techniques. The lack of an adequate source of multi-word entries is also proving to be a serious bottleneck, which we hope to address in future work.

2.3. Current applications of the LinGO technology developed at CSLI

The first application of the LinGO ERG was in the Verbmobil spoken language machine translation project (Wahlster, 1997). CSLI has been responsible for building the English grammar for the deep-processing component of Verbmobil, which utilizes a semantic transfer approach. The grammar therefore contains good coverage of the constructions most frequently found in the Verbmobil data which concerns meeting scheduling and travel reservations: it can currently produce semantic representations for about 83 per cent of the utterances in a corpus of transcriptions of some 10,000 utterances, which vary in length from one word to more than thirty words. The hand-built lexicon of around 5000 words is somewhat tuned to this domain. The other aspect of the grammar which is partially domain-specific are the weights which are associated with rules and lexical entries in order to guide the parser so that it can order possible analyses. In this application, some utterances are ungrammatical because of restarts etc, or because of errors in speech recognition. These are dealt with in Verbmobil by a component which attempts to reconstruct

[2] The figures we give here, and in the rest of this paper, are for Franz Allegro Common Lisp 5.0 (Linux) on a 450MHz Pentium PC with 512 megabytes of memory.

Sentence	Processing time (secs)		Number of	Number of
	First parse	All parses	parses	edges
Would either Monday or Tuesday morning of the next week work?	0.3	0.6	2	671
How did you say your Wednesday was, the twenty ninth?	0.6	0.8	11	658
Tuesday, the only time I would have would be at three in the afternoon.	2.7	4.0	78	2964
So if we can not make it on Thursday afternoon, we will have to, you know, look for something.	0.5	4.2	96	3464
Now that we have finished our last meeting, we need to arrange another one within the next two weeks.	0.4	17.4	128	8469
*So actually next the whole of next week is gone.	N/A	0.2	0	123
*And it is already been pushed off more than two weeks.	N/A	1.1	0	650
*How does Wednesday the twenty fourth after one o'clock?	N/A	2.7	0	1222

Table 1: Examples of LinGO grammar coverage. Processing times are with the current internal version of the LKB system.

a valid semantic representation from the fragments that the parser produces (Kasper et al., 1999).

Our second major application effort is to adapt the LKB and the ERG for use in a speech prosthesis system, to be used by people with disabilities to speed up their language production so they can carry on more natural conversations in a highly flexible manner. We are investigating an approach to generation which combines the grammar with corpus-based word-frequency data and conversational templates. Applied to speech prostheses, it enables the production of full sentences from minimal user input in a context-sensitive way (Copestake, 1997). We expect that this approach can also be applied more generally for efficient production of formulaic text like the structured reports used widely in business and government. It may also have utility in computer-aided language learning, both for people who are not fully literate, and those for whom English is not their first language. The main difference from the Verbmobil project, as far as grammar development is concerned, is that this system requires an open-ended lexicon.

2.4. Teaching

While the LinGO ERG aims for broad, precise coverage in realistic applications, the LKB platform is also being used with smaller grammars more relevant for educational purposes. Courses using the LKB have already been taught for (undergraduate and graduate) students at Stanford, Ohio State, North Carolina, SUNY Buffalo and Essex, and also at the European Summer School for Logic, Linguistics and Information (ESSLLI-98, ESSLLI-00). Undergraduate students at CSLI[3] have also implemented the grammar developed in an introductory syntax textbook (Sag and Wasow, 1999) in the LKB system, and we are distributing this with the aim that it be used as a basis for hands-on exercises and experimentation.

The requirements for a teaching system are in some ways more stringent than those for research. Although efficiency is generally less important (since teaching grammars tend to be small and to involve relatively little ambiguity), software reliability, debugging tools and ease of use are more important, especially since beginning students will typically be unsure of the formalism and are easily discouraged if a system behaves in an unexpected or unintuitive manner. This means that detailed and extensive testing is required on versions of the system which are used for teaching. Primarily because of this, improvements in the internal version of the LKB are typically not reflected in the public versions for some months, although we may move to an approach where all releases are available, but some are flagged as tested/major revisions suitable for general use.

[3]Chiefly Matt Kodama, Ryan Ginstrom, Christopher Callison-Burch and Scott Guffey.

3. Issues in collaborative development of grammars

There has been a large amount of discussion recently about the advantages of an open source model of software development and the ways in which this is facilitated by the increase in use of the Internet, so we will not repeat any of those general points here. Instead we will concentrate on issues specific to the collaborative development of grammars and the extent to which they can be genuinely multi-purpose.

3.1. Evaluation technology

Evaluation methodology as it pertains to the LinGO resources has been discussed at some length in previous papers (Oepen and Flickinger, 1998; Oepen and Callmeier, 2000), so here we just highlight a few points. Obviously a grammar must be tested in a manner which is realistic for any application for which it is being used. This generally involves collecting a testbed of inputs (utterances for parsing, semantic representations for conventional generation) and checking that the grammar produces acceptable output in a reasonable time. Although efficiency is partly the responsibility of the developer of the platform on which the grammar is run, it is also partly the responsibility of the grammar developer, since even the most efficient system can be made unacceptably slow. Regression testing ensures that coverage is not lost.

However, for development of a grammar that is intended for use in multiple applications, it is crucial to ensure that

coverage is not unduly domain- or application- specific. This means that the application-specific test suite has to be supplemented by one which contains (hand-constructed) examples which illustrate particular grammatical phenomena. For instance, the test sentence shown below would indicate whether the grammar could handle extraction from prepositional phrases:

Which office does Jones sleep in?

The sentences are not supposed to be realistic but must minimize ambiguity and use standardized vocabulary and phrases in order to make the results as clearcut as possible. This test suite should also contain negative examples, to ensure that the grammar does not overgenerate. The CSLI test suite (which is largely drawn from the HP test suite) contains around 1350 sentences (about 400 of which are ungrammatical). Again, this is used both as a target and for regression testing.

The issue of evaluating the results of parsing a test suite is complex, since hand-checking each result is too time-consuming. We use derivation trees as one way of summarizing the results of a parse which is useful for regression checking on the grammar, and especially for ensuring that revisions to the parser have not created bugs. The MRS representation allows a flexible way of checking semantic structures for equivalence: ensuring that the underspecified semantics is well-formed by constructing a scoped representation is also detects some grammar bugs.

Currently, the only way of checking generation in the absence of an application which constructs input representations is to analyze a test sentence and then attempt to generate from the results. We are actively investigating more satisfactory methods, but discussion of this leads into issues of defining interfaces for generation, which we cannot explore adequately in this paper.

3.2. Collaborative grammar coding

Any grammar development environment should provide explicit support for collaborative development. At a very basic level, the LKB facilitates code manipulation by allowing the grammar source to be split into multiple files, on the basis of functionality, for instance. We maintain source files for grammars using the standard CVS source control system, which allows multiple people to work on the same file and automatically merges different versions if the edits do not overlap. Although there is a possibility of introducing errors in this way, we have found the process works well if developers check in and update their source reasonably frequently.

We have successfully taught many students how to use the LKB system and develop small grammars, but the learning curve required to understand the LinGO ERG well enough to collaborate on it is very steep. Several people have contributed substantially to the LinGO grammar (see §6.), but only four of them have been in a position to do large-scale work on the core grammar, though several more people have been involved in more peripheral activities, such as adding lexicon.

We believe that our experience is reasonably representative. [4] It is often suggested that the problem with grammar engineering is that there is a lack of modularity, but it is not clear to us that this is correct. In software engineering generally, there are two conflicting goals: it is desirable to divide a task into components with hidden internal structure which can be developed independently of each other, but it is also desirable to avoid duplication of functionality. Different programming languages emphasize different paradigms: for instance, Modula-2 provided strong support for hiding data and functions but the object-oriented programming language C++ emphasizes commonality instead (Stroustrup, 1991). Information-hiding is often referred to as modularity in the software engineering literature: this is a much stronger sense than the idea of simply dividing up code. Some paradigms are more appropriate than others for specific application areas: e.g., Stroustrup argues that object-oriented programming (OOP) is more suited to graphics than to classical arithmetic.

In our experience with the LinGO ERG and previous grammar design work, commonality completely eclipses information-hiding in grammar design. While generalization in software engineering is motivated by practical considerations of avoiding errors and time-wasting due to redundancy, in grammar engineering there is an additional theoretical reason since a primary research aim in linguistics is capturing generalizations. Information-hiding is almost the antithesis of this, since it inherently involves having some parts of the representation which are only used in specified subsystems of the grammar. Consider the discovery that a feature which is used in the description of long-distance dependencies correlates with a phenomenon in morphology. This would be regarded as good news by a grammar developer and not as a failure of modularity, because it is a generalization that enhances the predictive power of the system. Furthermore, information-hiding modules are only useful in software development if they can be defined in the initial design, but they are inherently inflexible and therefore do not work well for more exploratory programming.

Because we cannot isolate individual linguistic phenomena, we cannot expect someone to work on an analysis without some knowledge of the rest of the grammar. But there are other notions of modularity. As with OOP, the inheritance hierarchy allows developers to work on expanding leaves without affecting the more general nodes. Some developers' tasks primarily involve classification. For instance, a lexicon can be extended by someone with little knowledge of the grammar because they can copy the classes allocated to words they know are similar. Similarly, a relatively untrained developer can add morphological rules, even though the morphology component cannot be a module in the information-hiding sense, because HPSG is a monostratal theory.

The LKB system has extensive tools for developing inheritance hierarchies. Unlike any other feature structure based system, it incorporates a fully order-independent version of default unification (Lascarides and Copestake,

[4] Although there is not all that much published work on grammar development: (Butt et al., 1999) is an exception.

1999). Currently the main LinGO ERG does not use defaults but Malouf has developed a version that does (Malouf, 1999), which demonstrates that it is possible to considerably simplify the type hierarchy. There is a trade-off since the use of defaults means the formalism is more complex and less like HPSG as it is standardly presented. However, we believe that overall defaults make the grammar much easier to understand. In fact, extensive use is made of the default formalism in recent theoretical work on HPSG (Ginzburg and Sag, to appear).

Despite the many aids to grammar engineering that have been developed, we think that to some extent it just has to be accepted that it really is inherently difficult. Grammar developers are attempting to construct a model which mirrors (some aspects of) a real-world phenomenon which is not at all well understood. Thus grammar development is much more of a research enterprise than conventional software engineering is. It necessarily involves frequent redesign and recoding of large components of the grammar (within the boundaries imposed by the initial linguistic framework).

As a first step to being able to work on the LinGO ERG, any potential grammar developer needs to have a good understanding of linguistics in general and HPSG in particular. This imposes a high initial barrier, especially because the HPSG framework changes rapidly: knowledge of the standard references (Pollard and Sag, 1987; Pollard and Sag, 1994) is only of limited use. Furthermore, even though the formalism implemented in the LKB system is relatively close to the formalism assumed in most theoretical work on HPSG, there is still a considerable element of design necessary in order to actually get an analysis to work. However, the most time-consuming part of grammar development is not implementing the well-worked out phenomena described in existing work on HPSG, but extending the analysis to deal with the constructions which are needed to handle real text or speech. This involves considerable linguistic research, before any attempt at coding can be made. It is also necessary to ensure that the result can be integrated with the rest of the framework, which may involve non-trivial modification to the core component. The positive side of this is that people who have sufficient understanding of linguistics to be good grammar developers find the research component interesting and motivating. Similarly, this is a good reason to develop a grammar using a framework that is relatively popular among linguists. In fact, to the extent that the ERG is an embodiment of research on HPSG, it is a desirable goal for students to gain detailed understanding of a substantial portion of it, rather than being limited to a small module. Several linguistics papers which have resulted from work on the LinGO ERG (Bender and Flickinger, 1999; Bender and Flickinger, in press; Smith, 1999).

3.3. MRS representation

The MRS representation is used in the LinGO ERG and, in a simplified form, in several teaching grammars.[5] MRS is discussed in detail in (Copestake et al., 1999). Its most salient feature from the current perspective is that it is a flat representation which uses explicit pointers to encode scope effects which are represented by recursive structures in more conventional formal semantic representations. The use of pointers, called *handles* in MRS, gives us an easy way of underspecifying quantifier scope. We have tried to give an intuitive idea of this in Figure 1, although space limitations preclude a detailed discussion. The LKB system contains a separate module for MRS processing, which is independent of the rest of the LKB and has also been used with PAGE. This module allows extraction of MRS structures from a feature structure representation, scoping, comparison, display etc. It also forms part of the LKB generator which accepts MRS structures as input.

We believe that MRS is a particularly suitable semantic representation for a general-purpose grammar because, while it allows encoding of a full predicate logic representation with generalized quantifiers, it is trivial to ignore scope representations and work with a subpart of the MRS which is more like propositional logic. This is very useful for machine translation because it simplifies the specification of the vast majority of transfer rules, while allowing for the explicit representation of scope, if that should be necessary. MRS thus offers the advantages of the forms of flat semantics advocated in (Phillips, 1993) and (Trujillo, 1995) without their loss of expressive power.

The way in which we use MRS in the ERG is designed to be as uncommitted to specific semantic theories as possible, giving a very surface-oriented form of semantics. A general-purpose grammar should encode enough information in the semantics to ensure that an application never needs to see the syntactic representation. In our case, the interface to an application is defined purely on the level of the MRS representation.[6] In other respects, the semantic representation should be as neutral as is consistent with the demands of the syntax-semantics interface.

MRS allows other forms of underspecification besides scopal relationships, since the MRS relations are in a type hierarchy. A concrete use of this is that in MRS one can specify a relation which is general between various temporal uses of prepositions, such as *on* and *in* as they occur in *on Saturday morning*, *in the morning* etc. Preposition choice here appears to be conventionalized, especially since *Saturday morning* and *the morning* could actually denote the same thing.

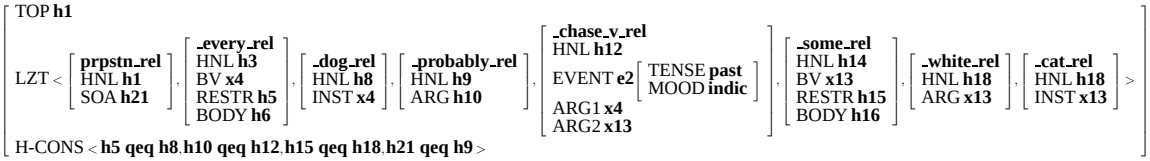
One advantage of underspecification in general is that it allows a more robust approach to generation. For instance, the module which constructs the MRS representation need not make a choice about the preposition in the case discussed above. This avoids the possibility it will make the

[5] A similar underspecified representation for use with typed feature structures known (a bit misleadingly) as Underspecified MRS (UMRS) was developed for a German grammar at IBM Heidelberg. UMRS is described in (Egg and Lebeth, 1995; Egg, 1998). (Abb et al., 1996) discuss the use of UMRS in semantic transfer (also see (Copestake et al., 1995) for semantic transfer using MRS).

[6] We actually relax this slightly in order to treat fragments of sentences since we currently regard an atomic abbreviation of the syntax of the phrase, such as NP, PP etc, as part of the interface. In the future, we will also need to consider discourse information and information structure more carefully.

every dog probably chased some white cat

Full notation:



Unscoped form, abbreviated notation:
$\langle h1, \{h1\!:\text{prpstn}(h21), h3\!:\text{every}(x4, h5, h6), h8\!:\text{dog}(x4), h9\!:\text{probably}(h10), h12\!:\text{chase}(e2, x4, x13),$
$h14\!:\text{some}(x13, h15, h16), h18\!:\text{white}(x13), h18\!:\text{cat}(x13)\},$
$\{h5 =_q h8, h10 =_q h12, h15 =_q h18, h21 =_q h9\}\rangle$

Scoped forms:
$\text{prpstn}(\text{probably}(\text{every}(x, \text{dog}(x), \text{some}(y, \text{white}(y) \wedge \text{cat}(y), \text{chase}(x,y)))))$
$\text{prpstn}(\text{every}(x, \text{dog}(x), \text{probably}(\text{some}(y, \text{white}(y) \wedge \text{cat}(y), \text{chase}(x,y)))))$
$\text{prpstn}(\text{every}(x, \text{dog}(x), \text{some}(y, \text{white}(y) \wedge \text{cat}(y), \text{probably}(\text{chase}(x,y)))))$
$\text{prpstn}(\text{probably}(\text{some}(y, \text{white}(y) \wedge \text{cat}(y), \text{every}(x, \text{dog}(x), \text{chase}(x,y)))))$
$\text{prpstn}(\text{some}(y, \text{white}(y) \wedge \text{cat}(y), \text{probably}(\text{every}(x, \text{dog}(x), \text{chase}(x,y)))))$
$\text{prpstn}(\text{some}(y, \text{white}(y) \wedge \text{cat}(y), \text{every}(x, \text{dog}(x), \text{probably}(\text{chase}(x,y)))))$

Figure 1: Example of MRS representation produced by the ERG

'wrong' choice (i.e., one not licensed by the grammar). One danger with this is that the input might be too underspecified, but the approach we are taking to this is to let corpus data act as an oracle to guide a generator. Specifically, in the context of the speech prosthesis project mentioned in §2.3., the input to the generator may underspecify closed-class words, such as determiners. Although the grammar provides some constraints (e.g. *much* may not occur with plural nouns), a totally underspecified determiner would generally result in generation of far more strings than the application could plausibly require. Frequency information from corpora can be used to make a best guess in an application such as this, where complete precision is not a requirement.

We need to do more work to define appropriate general-purpose interfaces to the semantics, especially for applications that use the grammar for generation. However, we believe MRS will eventually be highly suitable as a representation for a shared resource grammar.

4. Efficiency improvements via shared resources

The LinGO ERG has been central in a collaborative effort to improve technology for processing HPSG (and similar formalisms). Until recently, it has been argued that high-level grammar formalisms are too slow for real-world applications. Groups from the DFKI, Saarland University, the University of Tokyo and CSLI have collaborated on improving efficiency by using the LinGO grammar as a common reference point. All the sites used the [incr tsdb()] system to measure various parameters in parsing with the ERG on common test-suites. The LKB system was used as a baseline to validate correctness. It was also used to preprocess the type hierarchy and lexicon to avoid the necessity for each group to write a parser for the syntax used in the ERG. Practically speaking, this is much faster than attempting to get groups to agree on a common syntax for the grammar definition files. Similarly, the other systems have made use of various expanded forms of the grammar that can be output by the LKB, enabling them to bypass processing stages which are irrelevant for core parser comparison, such as expansion of the type hierarchy to form a semi-lattice and morphological processing.

Some small changes to the ERG were made to allow this comparison: for instance, there were a few places where the LKB's assumption that feature structures may not contain cycles was used in order to rule out otherwise valid unifications. However, it was not difficult to remove these cases and the ERG itself is now neutral with respect to cyclicity assumptions in that it does not generate cyclic structures (modulo bugs).

The combination of the LinGO ERG and the [incr tsdb()] test-suite machinery has enabled much more detailed cross-platform performance evaluation on realistic grammars than has previously been possible. The result has been a combining of processing techniques developed by the various groups to mutual benefit. (Kiefer et al., 1999) is a partial report on some of this work; (Flickinger et al., 2000) contains detailed discussions by most of the participants in this collaboration. The LKB now incorporates techniques that were adapted from other systems, especially PAGE and PET (Callmeier, 2000).

5. Other collaborations and future work

In addition to the work on processing efficiency and performance profiling described above, the ERG and the LKB also form the basis of well-established collaborations on extraction of stochastic lexicalized tree grammars (Neumann, 1997) and the integration with discourse models and pragmatics (Copestake and Lascarides, 1998). Emerging joint projects include the adaptation of the ERG lexical type hierarchy to large English and Japanese lexicons developed for MT in collaboration with Francis Bond and other researchers from NTT. We are actively investigating possible

uses of the LinGO resources with other CSLI industrial affiliate companies.

Both the LinGO grammar and the LKB continue to be actively developed. Besides work described above, our current and planned research is focused in the following areas:

- extending the lexicon through adaptation of existing lexical resources and through acquisition from corpora
- using corpora as an additional information source in language generation
- incorporating statistical data derived from corpora to cope with the ambiguity inherent in wide-coverage grammars
- building exploratory grammars for other languages and investigating the extent to which grammars can be shared
- representing dialect variation in both the lexicon and the grammar
- developing discourse models which interact with the grammar in particular to allow treatment of fragments

6. Acknowledgments

Development of the LKB system was originally supported by ACQUILEX projects BRA-3030 and 7315 under the Esprit program (grant to Cambridge University). More recent research has been supported by the National Science Foundation under grant number IRI-9612682 (grant to Stanford University). Development of the LinGO ERG was supported by NSF IRI-9612682 and by German Federal Ministry of Education, Science, Research and Technology (BMBF) in the framework of the Verbmobil Project under Grant FKZ:01iV401.

The main developers of the LKB system are Ann Copestake, John Carroll (University of Sussex), Rob Malouf (University of Groningen) and Stephan Oepen (Saarland University). The distributed system also incorporates code written by Bernie Jones (while at Cambridge), Dan Flickinger and John Bowler. Ted Briscoe and Antonio Sanfilippo had a great deal of influence on the design of the original system for ACQUILEX, Dan Flickinger played a similar role for the new version. Ulrich Callmeier (DFKI and Saarland University) has done extensive testing and uncovered numerous bugs and infelicities. Victor Poznanski (Sharp Laboratories of Europe) helped develop the generation algorithm, Guido Minnen (Sussex) and Stefan Thater (Saarland University) have investigated corpus-derived constraints on generation.

The LinGO ERG was mainly developed by Dan Flickinger, Rob Malouf, Emily Bender and Jeff Smith (San Jose State University). Several Stanford students (Brady Clark, Judith Tonhauser, Kathryn Campbell-Kibler, Martina Faller, Ash Asudeh, Susanne Riehemann) and visiting students (Jesse Tseng, Edinburgh; Ken Bame, Ohio State; Judith Eckle-Kohler, Stuttgart; Martine Smets, currently at Sussex) have also done detailed work, including building the lexicon, developing test suites, isolating phenomena found in corpora and developing analyses in the HPSG formalism. Ivan Sag, Tom Wasow and Carl Pollard provided theoretical insights. Ana Quirino Simoes (Bielefeld), Chris Callison-Burch, Aline Villavicencio (Cambridge) and Judith Baur (Saarland University) have developed other grammars which have led to improvements in the LKB system and alternative analyses for the ERG. See http://lingo.stanford.edu/members.html for a full list of LinGO project affiliates.

Invaluable input has of course also been provided by the individuals mentioned or cited in the text and by other people too numerous to thank individually, including researchers on the ACQUILEX and LinGO projects, at the Copenhagen Business School, Saarland University, University of Essex, Edinburgh University, University of Cambridge and Stanford University and students on a course taught by Copestake, Flickinger and Oepen at ESSLLI-98.

7. References

Abb, Bernd, Bianka Buschbeck-Wolf, and Christel Tschernitschek, 1996. Abstraction and underspecification in semantic transfer. In *Proceedings of the Second Conference of the Association for Machine Translation in the Americas (AMTA-96)*. Montreal.

Alshawi, Hiyan (ed.), 1992. *The Core Language Engine*. Cambridge, MA: MIT Press.

Bender, Emily and Dan Flickinger, 1999. Peripheral constructions and core phenomena. In Andreas Kathol, Jean-Pierre Koenig, and Gert Webelhuth (eds.), *Lexical and Constructional Aspects of Linguistic Explanation*. CSLI Publications, pages 199–214.

Bender, Emily and Dan Flickinger, in press. Diachronic evidence for extended Argument Structure. In Gosse Bouma, Erhard Hinrichs, Geert-Jan Kruijff, and Richard Oehrle (eds.), *Constraints and Resources in Natural Language Syntax and Semantics*. CSLI Publications.

Briscoe, Ted, Claire Grover, Bran Boguraev, and John Carroll, 1987. A formalism and environment for the development of a large grammar of English. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI-87)*. Milan, Italy.

Butt, Miriam, Tracy Holloway King, María-Eugenia Niño, and Frédérique Segond, 1999. *A Grammar Writer's Cookbook*. Stanford: CSLI Publications.

Callmeier, Ulrich, 2000. PET — A platform for experimentation with efficient HPSG processing techniques. In *(Flickinger et al., 2000)*.

Carpenter, Bob, 1992. *The Logic of Typed Feature Structures*. Cambridge, UK: Cambridge University Press.

Carpenter, Bob and Gerald Penn, 1999. ALE: The Attribute Logic Engine / user's guide version 3.2. University of Tübingen, <http://whorf.sfs.nphil.uni-tuebingen.de/~gpenn/ale/files/guide.tex>.

Carroll, John, Ann Copestake, Dan Flickinger, and Victor Poznanski, 1999. An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG'99)*. Toulouse, France.

Copestake, Ann, 1992. The ACQUILEX LKB: representation issues in semi-automatic acquisition of large lexicons. *Proceedings of the 3rd Conference on Applied Natural Language Processing, Trento, Italy*:88–96.

Copestake, Ann, 1997. Augmented and alternative NLP techniques for augmentative and alternative communication. In *Proceedings of the ACL workshop on Natural Language Processing for Communication Aids*. Madrid.

Copestake, Ann, in press. *Implementing typed feature structure grammars*. CSLI Publications, Stanford.

Copestake, Ann, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag, 1995. Translation using Minimal Recursion Semantics. In *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-95)*. Leuven, Belgium.

Copestake, Ann, Dan Flickinger, and Ivan Sag, 1999. Minimal recursion semantics: An introduction. Ms., CSLI, Stanford University.

Copestake, Ann and Alex Lascarides, 1998. Resolving underspecified values with discourse information. Paper presented at the workshop on models of underspecification and the representation of meaning, Bad Teinach, Germany.

Egg, Marcus, 1998. *Wh*-questions in Underspecified Minimal Recursion Semantics. *Journal of Semantics*, 15:1:37–82.

Egg, Marcus and Kai Lebeth, 1995. Semantic underspecification and modifier attachment. Intergrative Ansätze in der Computerlinguistik. Beiträge zur 5. Fachtagung für Computerlinguistik der DGfS.

Flickinger, Daniel, Stephan Oepen, Hans Uszkoreit, and Jun'ichi Tsujii (eds.), 2000. *Journal of Natural Language Engineering. Special Issue on Efficient Processing with HPSG: Methods, Systems, Evaluation*. Cambridge, UK: Cambridge University Press. In preparation.

Ginzburg, Jonathan and Ivan A. Sag, to appear. *English Interrogative Constructions*. CSLI Publications.

Grishman, Ralph, Catherine Macleod, and Adam Meyers, 1994. Comlex syntax: building a computational lexicon. In *Proceedings of International Conference on Computational Linguistics, COLING-94*. Kyoto, Japan.

Kasper, Walter, Bernd Kiefer, Hans-Ulrich Krieger, C.J. Rupp, and Karsten Worm, 1999. Charting the depths of robust speech parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*. University of Maryland, USA.

Kiefer, Bernd, Hans-Ulrich Krieger, John Carroll, and Robert Malouf, 1999. A bag of useful techniques for efficient and robust parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. University of Maryland, USA.

Lascarides, Alex and Ann Copestake, 1999. Default representation in constraint-based frameworks. *Computational Linguistics*, 25:55–106.

Malouf, Robert, 1999. Practical default inheritance in constraint-based grammars. Paper presented at Ohio State University.

Malouf, Robert, John Carroll, and Ann Copestake, 2000. Efficient feature structure operations without compilation. In *(Flickinger et al., 2000)*.

Neumann, Günter, 1997. Applying explanation-based learning to control and speeding-up natural language generation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL-EACL 97)*. Madrid.

Oepen, Stephan and Ulrich Callmeier, 2000. Measure for measure: Parser cross-fertilization. Towards increased component comparability and exchange. In *Proceedings of the 6th International Workshop on Parsing Technologies*. Trento, Italy.

Oepen, Stephan and John Carroll, 2000. Ambiguity packing in constraint-based parsing. Practical results. In *Proceedings of the 1st Conference of the North American Chapter of the ACL*. Seattle, WA.

Oepen, Stephan and Daniel P. Flickinger, 1998. Towards systematic grammar profiling. Test suite technology ten years after. *Journal of Computer Speech and Language: Special Issue on Evaluation*, 12 (4):411–437.

Oepen, Stephan, Klaus Netter, and Judith Klein, 1997. TSNLP — Test Suites for Natural Language Processing. In John Nerbonne (ed.), *Linguistic Databases*. Stanford: CSLI Lecture Notes 77, CSLI Publications, pages 13–36.

Phillips, J.D., 1993. Generation of text from logical formulae. *Machine Translation*, 8 (4):209–235.

Pollard, Carl and Ivan A. Sag, 1987. *An information-based approach to syntax and semantics: Volume 1 fundamentals*. Stanford: CSLI Lecture Notes 13, CSLI Publications.

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

Sag, Ivan A. and Thomas Wasow, 1999. *Syntactic Theory: A Formal Introduction*. CSLI Publications, Stanford.

Sanfilippo, Antonio, 1993. LKB encoding of lexical knowledge from machine-readable dictionaries. In Edward J. Briscoe, Ann Copestake, and Valeria de Paiva (eds.), *Inheritance, defaults and the lexicon*. Cambridge University Press, Cambridge, England.

Shieber, Stuart, 1986. *An introduction to unification-based approaches to grammar*. CSLI Lecture Notes 4, Stanford.

Smith, Jeffrey D., 1999. English number names in HPSG. In Gert Webelhuth, Andreas Kathol, and Jean-Pierre Koenig (eds.), *Lexical and Constructional Aspects of Linguistic Explanation*. Stanford: CSLI Publications.

Stroustrup, Bjarne, 1991. *The C++ programming language — second edition*. Addison Wesley.

The XTAG Research Group, 1995. A Lexicalized Tree Adjoining Grammar for English. Technical report, IRCS Report 95-03, University of Pennsylvania.

Trujillo, Arturo, 1995. Lexicalist machine translation of spatial prepositions. PhD dissertation, University of Cambridge.

Wahlster, Wolfgang, 1997. VerbMobil — Erkennung, Analyse, Transfer, Generierung und Synthese von Spontansprache. VerbMobil Report 198, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Saarbrücken, Germany.