Natural Language Engineering **6** (1): 1–14. Printed in the United Kingdom

© 2000 Cambridge University Press

Journal of Natural Language Engineering # 6 (1). Special Issue on Efficient Processing with HPSG (March 2000)

Introduction to this Special Issue

STEPHAN OEPEN
Computational Linguistics, Saarland University,
Im Stadtwald, 66123 Saarbrücken, Germany
e-mail: `oe@coli.uni-sb.de`

DAN FLICKINGER
Center for the Study of Language and Information, Stanford University,
Stanford, CA 94305, USA
e-mail: `dan@csli.stanford.edu`

HANS USZKOREIT
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH and
Computational Linguistics, Saarland University
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
e-mail: `hansu@dfki.de`

JUN-ICHI TSUJII
Department of Information Science, Graduate School of Science, University of Tokyo and
Language Engineering, University of Science and Technology in Manchester,
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033 Japan
e-mail: `tsujii@is.s.u-tokyo.ac.jp`

(*Received 6 March 2000*)

Abstract

This volume reports on recent achievements in the domain of HPSG-based parsing. Research groups at Saarbrücken, CSLI Stanford, and the University of Tokyo have worked on grammar development and processing systems that allow the use of HPSG-based processing in practical application contexts. Much of the research reported here has been collaborative, and all of the work shares a commitment to producing comparable results on wide-coverage grammars with substantial test suites. The focus of this special issue is deliberately narrow, in order to allow detailed technical reports on the results obtained among the collaborating groups. Thus, the volume cannot aim at providing a complete survey on the current state of the field.

This introduction summarizes the research background for the work reported in the volume and puts the major new approaches and results into perspective. Relationships to similar efforts pursued elsewhere are included, along with a brief summary of the research and development efforts reflected in the volume, the joint reference grammar, and the common sets of reference data.

Relevant only to some extent. State-of-the-art parsers are moving away from complex feature structure systems.

(Anonymous NAACL 2000 Reviewer)

1 Do We Need (Deep) Linguistic Processing?

Much like the global economy, the stock exchange, and *haute couture*, natural language engineering (the field) exhibits a cyclic progression of dominating paradigms and development currents. And although Natural Language Engineering (the journal) has demonstrated a respectable degree of independence from the streams of fashion, we view the production of this Special Issue as an indicator of a new development: the return of precise linguistic grammars and constraint-based processing for practical applications.

The goal of capturing linguistic knowledge—providing a model of the system of language in a form suitable for computer-based, algorithmic processing—has always been among the central concerns of Computational Linguistics and Natural Language Processing (NLP). Formal clarity, descriptive adequacy, declarativity, modularity, re-usability, and related concepts have been desiderata for NLP theories and systems from the very beginning. (Context-free) Phrase structure grammar (Chomsky, 1959), augmented transition networks (Woods, 1970), definite clause grammars (Pereira & Warren, 1980), chart parsing (Younger, 1967; Kay, 1973), feature structures and unification (Kay, 1979), taxonomic logics (Brachman & Schmolze, 1985), and constraint-based approaches to grammar and processing (Sells, 1985; Shieber, 1986) mark some of the milestones in the development of the field. The 1980s saw an immense increase in the number of research projects and development efforts (some in industrial environments) working on the production of declarative grammatical resources and suitable processing techniques, many of them aiming for (often very complex) query processing, dialogue system, or machine translation applications. This traditional strain of NLP is now often referred to as 'deep' processing.

The 1996 final report of the European Expert Advisory Group (EAGLES) on Linguistic Formalisms lists about a dozen implemented grammar development and processing environments (Uszkoreit et al., 1996).[1] Head-Driven Phrase Structure Grammar (HPSG; Pollard & Sag, 1994) and to a slightly lesser extent Lexical Functional Grammar (LFG; Dalrymple, Kaplan, Maxwell, & Zaenen, 1995) and Tree Adjoining Grammar (TAG; Joshi, 1987) are the predominant paradigms according to the EAGLES survey. In retrospect, it may seem little has changed in the past five years. HPSG and LFG continue to be the most widely accepted unification-based theories of grammar within Computational Linguistics and are gaining ground as non-transformational alternatives to Chomskyan grammar in formal and theoretical Linguistics proper. The majority of established grammar development environments are still around, though some have disappeared or lost importance, and we are not aware of new developments started recently.

At the same time, however, the 1990s and especially the past five years have seen a shift of emphasis: a large number of current NLP applications focus on a slightly different, linguistically often less demanding problem than (proto-)typical systems ten years earlier. Precise, in-depth syntactic and semantic analysis are far

[1] The complete report can be accessed on-line from the EAGLES home page at Pisa; see '`http://www.ilc.pi.cnr.it/EAGLES96/home.html`'.

less important in text retrieval, message understanding, or information extraction contexts than they are for a dialogue or machine translation system, for example. Instead, the applicability to large amounts of naturally-occurring input (typically text), overall system coverage and robustness, domain-oriented processing, and general fitness for a specific task are among the primary requirements for what have come to be known as 'shallow' (text) processing systems. The DARPA-sponsored TREC and MUC conferences—a series of competitive, task-oriented system evaluation meetings—have made at least two significant contributions to the field: (i) because the common evaluation metric is strictly black-box and task-driven, a diversity of approaches ranging from finite-state to probabilistic (and often hybrid) systems were encouraged; and (ii) given the large funding body behind the evaluations, public research efforts, especially in the US, were polarized between working either within the shallow processing paradigm, or deliberately outside of it.

Shallow processing techniques have produced useful results in some classes of applications, but they have not met the full range of needs for NLP, particularly where precise interpretation is important, or where the variety of linguistic expression is large relative to the amount of training data available. For such applications, especially ones involving (non-trivial) semantic processing and language generation such as machine translation, automated response systems, or speech prostheses, the quality of each output from the system will be judged against a readily accessible human standard. While robustness remains important, it is in tension with the user expectation of correct, natural results from the system, and deep processing can provide informed estimates of correctness, either because a given linguistic expression is within the scope of the grammar, or because it falls outside of the grammatical coverage in some quite specific respects. These measures of how confident the system is of its results can be of real use, both in avoiding deceptive or confusing output, and in ranking logically correct outputs when the available context is not rich enough to resolve ambiguous expressions.

2 Why is (Deep) Linguistic Processing a Hard Problem?

Linguistic expressions taken out of context are incomplete and ambiguous, since the speaker counts on the hearer to supply common sense and world knowledge as part of the understanding process. Not only is a lot left unsaid, but many words in what is said have multiple meanings, and the ways they are combined give rise to even more possible meanings for a given utterance. Yet humans succeed very well in processing natural language, apparently unaware of most of the logically possible interpretations of what they say or hear.

Contemporary NLP systems cannot hope to have access to the vast amount of real world knowledge that humans enjoy, nor can they expect to reason very well about the modest amount of knowledge that is formally represented in current machines. But contemporary systems can exploit the rich and steadily growing store of detailed linguistic knowledge to at least identify those interpretations of an utterance which are logically possible, and to avoid false understandings. Linguists can provide precise descriptions of the words of a language, and of the rules that govern how they can be combined to produce meaningful utterances. Implementing such lexicons and grammars in an NLP system requires sustained collaboration between the theoretical linguist and the grammar writer, since even the formal tools for representing linguistic knowledge undergo steady refinement. And the grammar writer must often find engineering solutions to fill in gaps in the body of theoretical work on a language, since in any NLP application there are quite ordinary expressions that remain unanalyzed within a given theoretical framework.

Deep processing of language necessarily involves making a great number of fine-grained distinctions about how the words and phrases of a language relate form to meaning, and this level of detail can prove to be expensive computationally. Within the HPSG framework adopted for the grammars reported on in this volume, the descriptions of linguistic signs (both words and phrases) are large, and will only get larger as more of the language is analyzed. The size and nature of these signs presents an interesting challenge for the NLP system developer who wants to meet the efficiency requirements of a given application. Let us substantiate these observations with a few real-world numbers obtained from the LinGO grammar (Flickinger, this volume) and using the PET parser (Callmeier, this volume): each feature structure built in the parser, on average, has some three hundred internal nodes, each of around 80 bytes in size (including outgoing arcs). While parsing a representative sample (viz. the '*blend*' test set described below), the unifier on average executes more than four thousand top-level unifications per sentence (in average total time of less than a second), which corresponds to close to one hundred megabytes of memory that are being visited (i.e. dereferenced, not necessarily allocated). Not surprisingly, nearly forty per cent of total parsing time are spent in the unifier, and another forty five per cent in feature structure copying.

While it is this issue of efficient processing that provides the focus for the papers in this volume, we note that consumption of time and space are not the only challenges facing the developers of a useful deep NLP system. Competing with the desire for efficiency are the goals of (i) broader coverage of the linguistic expressions needed for a given application; (ii) avoiding false analyses of utterances (which can easily arise as coverage grows); (iii) correctly ranking the alternatives for utterances that the grammar finds ambiguous; and (iv) retaining a close connection between the implemented grammar and the theoretical work that informs its design.

3 Multilateral Collaboration: Our Setup

In early 1994, research groups at Saarbrücken[2] and CSLI Stanford[3] started to collaborate on the development of large-scale HPSG grammars, suitable grammar engi-

[2] See '`http://www.dfki.de/lt/`' and '`http://www.coli.uni-sb.de/`' for information on the DFKI Language Technology Laboratory and the Computational Linguistics Department at Saarland University, respectively.

[3] The '`http://lingo.stanford.edu/`' web pages list HPSG-related projects and people involved at CSLI, and also provide an on-line demonstration of the LKB system and LinGO grammar. As a Stanford visiting scholar, John Carroll of Sussex University had a great deal of influence on the efficient reimplementation and optimization of core LKB components and (now back at home in Sussex) continues to be an active LKB developer.

neering platforms, and efficient processors. Since the early 1990s, the Saarbrücken group had been developing an HPSG-based dialogue system, including a highly expressive typed feature formalism, a medium-coverage grammar of German, and an application prototype for distributed email-based appointment scheduling (Uszkoreit et al., 1994; Krieger & Schäfer, 1994; Erbach et al., 1995). CSLI, on the other hand, had long been among the driving forces in the theoretical development of the HPSG theory of grammar, and could at the same time build on system and grammar building experience gained in the Hewlett-Packard NL and the EU-funded ACQUILEX projects (Flickinger, Nerbonne, Sag, & Wassow, 1987; Copestake, 1992). The close collaboration developed when both sites started participating in Verb*mobil* (Wahlster, 1997), a distributed project on spoken dialogue translation[4] comprising more than twenty groups, and adopting HPSG as the common grammar model for deep processing. The English grammar was developed at Stanford, whereas the German grammar and core processing environment was contributed by DFKI Saarbrücken; Saarland University supplied the Japanese Grammar and robust semantics. The multi-site efforts in grammar-based analysis were coordinated by Hans Uszkoreit. Collaboration has greatly increased productivity, resulted in a mutual exchange of knowledge and technology, and helped building a collection of grammar development environments, several highly engineered parsers (Kiefer, Krieger, Carroll, & Malouf, 1999), and an efficient generator (Carroll, Copestake, Flickinger, & Poznanski, 1999). In 1998, the grammar formalisms and parsing group at Tokyo University[5] joined the consortium and now supplies additional expertise on (abstract-machine-based) compilation of typed feature structures, Japanese HPSG, and grammar transformation and approximation techniques (Torisawa & Tsujii, 1996; Makino, Yoshida, Torisawa, & Tsujii, 1998; Tateisi, Torisawa, Miyao, & Tsujii, 1998).

The primary goal of this multilateral collaboration is to synchronize efforts on the development and deployment of efficient, large-scale HPSG processors, thereby enhancing the effectiveness of each group in doing its own focused research. Grounded in these common goals, the sites have agreed on a joint descriptive formalism and reference grammar and are now engaged in a constructive competition for premium processing performance within this framework.

3.1 Converging on a Joint Formalism and Reference Grammar

Given a broad acceptance of unification-based approaches to computational grammar—and in particular of the HPSG and LFG frameworks—it may seem from the outside that the formal foundations of (typed) feature structures have long been established. While this may well be true from a mathematical point of view (Rounds & Kasper, 1986; Carpenter, 1992), it is less so seen from the implementation per-

[4] Verb*mobil* was funded by the German Federal Ministry of Education, Science, Research, and Technology (BMBF) under Grant 01 IV 701 V0.

[5] Information on the Tokyo laboratory, founded and managed by Professor Jun-ichi Tsujii, can be found at '`http://www.is.s.u-tokyo.ac.uk/`'.

spective. The main degree of variation here is not in different interpretations of individual concepts but in the particular choice of descriptive devices that a token system makes from a set of options and alternatives that has been growing continuously. Open- vs. closed-world reasoning, single vs. multiple inheritance, various approaches to disjunction and negation (in different flavours), set-valued feature structures, the precise semantics of the type system, and the inclusion of implicational or relational constraints are some of the dimensions that, when applied to the systems listed in the above mentioned 1996 EAGLES survey, for example, make each implementation distinct in the range of formal devices that it has to offer.

Although the individual systems developed within our consortium often supply extra functionality, the groups have converged on a common descriptive formalism—a conservative blend of Carpenter (1992), Copestake (1992), and Krieger & Schäfer (1994)—that allows grammars[6] to be processed by five different platforms. But this joint formalism is by no means the mere intersection (or, loosely speaking, the smallest region of overlap) between the environments represented among the participating groups; instead, the selection of formal and descriptive devices was guided by two major concerns: (i) linguistic adequacy, grounded in nearly three decades of joint experience in building large-scale HPSG-type grammars, and (ii) processing requirements, informed by earlier work on efficient implementations. The decision to eliminate (explicit) disjunction from the linguistic specification language, for example, is motivated by theoretical and engineering considerations alike. Flickinger (this volume) argues that a grammatical stipulation that makes disjunctive information explicit in underspecified types in the grammatical ontology (rather than by disjunctive enumeration) can be interpreted as a stronger model of what (co-)variation the grammar actually foresees. At the same time, moving to a purely conjunctive feature logic allowed the adaptation and fine-tuning of existing, very efficient unification techniques (Malouf, Carroll, and Copestake, this volume) that avoid expensive backtracking and duplication of redundant structure.

The joint descriptive formalism can be informally characterized as a closed-world, conjunctive-only, multiple inheritance type system that enforces strong typing and strict appropriateness, but allows types to be associated with arbitrary (complex) constraints that are inherited and applied both at compile and at run-time (e.g. when two types unify to a more specific, constraint-introducing subtype). HPSG well-formedness principles, immediate dominance schemata, and constituent ordering constraints are all spelled out in the type hierarchy (and cross-multiplied), yielding a set of phrase structure schemata that can be interpreted as rewrite rules over complex (typed feature structure) categories by a suitable parser or generator. A precise mathematical specification of this formalism as it is assumed throughout the volume is given in the Appendix (Copestake, this volume). And although our conservative choice of descriptive devices is fairly restrictive—in particular when compared to a general-purpose inference and type deduction system like TFS (Emele, 1994),

[6] In the HPSG universe (and accordingly the present volume) the term 'grammar' is typically used holistically, referring to the linguistic system comprised of (at least) the type hierarchy, lexicon, and rule apparatus.

for example—it has enabled the development of several large grammars as well as the implementation of HPSG processing systems that perform with previously unmatched efficiency.

The LinGO grammar, a multi-purpose, broad-coverage grammar of English developed at CSLI and to our best knowledge the largest HPSG implementation currently available, serves as a common reference for all three groups (while of course the sites continue development of additional grammars for English, German, Japanese, and other languages). The grammar primarily serves as a representative sample of the common approach to linguistic description and the joint specification language, rather than as a fixed target to which systems are being tuned. As each site regularly evaluates their system(s) against other, only abstractly similar grammars, and since it has often been confirmed that the techniques evolving from the collaboration proved beneficial beyond the LinGO grammar, the contributions in this volume can be taken as a representative report on this particular line of research in HPSG processing. Flickinger (this volume) provides details on the LinGO grammar, including reasoning about some of the design decisions made in the underlying formalism; unless stated otherwise, all contributions in the volume refer to the August 1999 LinGO version, which was frozen as a common reference point.

With one hundred thousand lines of source, roughly eight thousand types, an average feature structure size of some three hundred nodes, twenty seven lexical and thirty seven phrase structure rules, and some six thousand lexical (stem) entries, the LinGO grammar presents a fine challenge for processing systems; a multiple-inheritance ontology with several thousand types, for example, is a rare configuration, even in large-scale object-oriented applications. While scaling the systems to the rich set of constraints embodied in the LinGO grammar and improving processing and constraint resolution techniques, the groups have regularly exchanged benchmarking results, in particular at the level of individual components, and discussed benefits and disadvantages of particular encodings and algorithms. Precise comparison has been found to be essential in this process and has facilitated a degree of cross-fertilization that has proved beneficial for all participants.

3.2 The Reference Data

For comparison and benchmarking purposes with the LinGO grammar three test suites and development corpora were chosen: (i) the CSLI test suite derived from the original Hewlett-Packard data (Flickinger et al., 1987), (ii) a small collection of transcribed dialogue utterances collected in the Verb*mobil* project, and (iii) a larger extract from recent Verb*mobil* corpora that was selected pseudo-randomly to achieve a balanced distribution of one hundred samples for each input length below twenty words. Some salient properties of these test sets are summarized in Table 1.[7]

[7] While wellformedness and item length are properties of the test data proper, the indicators for average ambiguity and feature structure (fs) size were obtained using the current release version of the LinGO grammar, frozen in August 1999. Here and in the tables to come the symbol '$\sharp$' indicates absolute numbers, while 'ϕ' denotes average values.

Set	Aggregate	total items ♯	word string φ	lexical entries φ	total results ♯	parser analyses φ	passive edges φ
'csli'	wellformed	918	6·45	15·3	732	2·16	115
	illformed	375	6·11	14·9	85	2·31	84
'aged'	wellformed	96	8·41	23·1	72	7·00	292
'blend'	wellformed	1910	11·13	32·1	1008	51·39	1181
	illformed	142	11·05	34·2	24	20·33	611

Table 1. *Reference data sets used throughout the volume.*

Looking at the degrees of lexical (i.e. the ratio between columns five and four), global (column seven), and local (approximated in column eight by the number of passive edges created in pure bottom-up parsing) ambiguity, the three test sets range from very short and unambiguous to mildly long and highly ambiguous. Contrasting columns six and three (i.e. items accepted by the grammar vs. total numbers of well- or ill-formed items) provides a measure of grammatical coverage and overgeneration, respectively.[8]

The *'blend'* test set is a good indicator of maximal input complexity that the available parsers can currently process (in plausible amounts of time and memory). See the benchmarking results presented by Callmeier (this volume) for precise performance data on this test set. For improved comparability, all systems were allowed to impose an upper limit on the number of passive edges built in non-predictive bottom-up parsing; using a limit of twenty thousand edges resulted in the exclusion from the comparison of sixty-seven items from the original *'blend'* set.

3.3 Benchmarking and Comparison

In system development and optimization, subtle algorithmic and implementational decisions often have a significant impact on system performance, so monitoring system evolution very closely is crucial. System performance, however, cannot be adequately characterized merely by measurements of overall processing time (and perhaps memory usage). Properties of (i) individual modules (in a classical setup, especially the unifier, type system, and parser), (ii) the grammar being used, and (iii) the input presented to the system all interact in complex ways. In order to obtain an analytical understanding of strengths and weaknesses of a particular configuration, finer-grained records are required. Among the participating groups (and in particular during the production of this volume) a common approach to

[8] Coverage on the *'blend'* corpus is comparatively low, as this test set became available only after the LinGO reference version had been frozen, and in particular some frequent lexical items are missing from the grammar.

benchmarking and comparison has served as a 'clearing house' in the production and exchange of comparable, reproducible data sets.

The methodology was introduced using the term *competence & performance profiling* (by analogy to standard software engineering techniques) by Oepen & Flickinger (1998); a competence & performance profile is defined as a rich, precise, and structured snapshot of system behaviour at a given development point. The production, maintenance, and inspection of profiles is supported by a specialized software package (called [incr tsdb()][9]) that supplies a uniform data model, an application program interface to the grammar-based processors, and graphical facilities for profile analysis and comparison. Profiles are stored in a relational database which accumulates a precise record of system evolution, and which serves as the basis for flexible report generation, visualization, and data analysis via basic descriptive statistics. Oepen & Carroll (this volume) review some of the details of the profiling approach used within the consortium, inasmuch as they are relevant to this Special Issue. Additionally, complete profiles for most of the contributions in the volume are available on-line; see below.

4 Scope of this Volume — Related Work

The research contributing to this volume was first presented at an internal working meeting of the three cooperating groups (held in Berlin, Germany, in March 1999) and subsequently as part of a topical workshop (held at Schloß Dagstuhl, Germany, in October of the same year).[10] The current collection documents a large body of practical research and engineering, ranging from linguistic adaptation of the grammatical specification (Flickinger, this volume), over improved constraint resolution (Malouf et al., this volume, and Miyao, Makino, Torisawa, and Tsujii, this volume) and parsing strategies (Oepen & Carroll, this volume), down to the compilation of a context-free approximation for large-scale HPSG grammars (Torisawa, Nishida, Miyao, and Tsujii, this volume); also, a comparative assessment of what broad progress has been achieved through the synthesis of several of these techniques, compared to system performance in 1996, is included (Callmeier, this volume). The volume presents a representative snapshot of where the joint effort on efficient HPSG processing has taken us so far, and at the same time provides a good summary of previously unpublished implementation experience. Given this narrow focus, the Special Issue cannot serve as a survey of the state-of-the-art in HPSG processing, let alone constraint-based grammar in general. There are, in fact, a large number of ongoing activities, some directly related to work reported here, and others similar in spirit, which we cannot reflect in this volume.

Closely related projects pursued at DFKI Saarbrücken (and involving the LinGO reference grammar) which are not represented in our current collection include

[9] See '`http://www.coli.uni-sb.de/itsdb/`' for the (draft) [incr tsdb()] user manual, pronunciation guidelines, and instructions on obtaining and installing the package.

[10] We are grateful to Verb*mobil* for financial support of the March meeting and to the Dagstuhl Foundation for supporting the October workshop.

(i) the adaptation of the Platform for Advanced Grammar Engineering (PAGE)—the environment that was initially used for LinGO and other Verb*mobil* grammar engineering—for the robust and efficient parsing of word recognizer output in Verb*mobil* (Kiefer et al., 1999); (ii) the development of a compiler for LinGO-type grammars that—by virtue of an intermediate abstract-machine-based representation—generates ANSI C program code for an efficient parser (CHIC: Compiling HPSG into C; Ciortuz, 2000); and ongoing research on (iii) context-free approximation techniques for (large-scale) HPSG grammars (Kiefer & Krieger, 2000) and on (iv) the acquisition of domain-specific, stochastic tree substitution grammars from parsing corpus data (Neumann & Flickinger, 1999).

Taking a slightly wider perspective for a brief moment, we see related work being pursued at several sites in Europe and the US. Among others, the Department of Linguistics at Tübingen University (Germany) continues research on formalism and grammar development (in the ConTroll system; Götz & Meurers, 1997), though with a different focus: unlike our own consortium, the Tübingen group explores a logically very rich and advanced formalism that facilitates the direct encoding of HPSG principles and well-formedness constraints as they were articulated in the original HPSG theory (Pollard & Sag, 1987, 1994). The theoretical and formal development of the framework, accordingly, are primary concerns for the basic research done at Tübingen, whereas the construction of large-scale grammars and efficient processors take more of a back-seat position.[11] In a similar—theory- more than application-driven—vein the Linguistics Department at Ohio State University (USA) is investigating linearization-based extensions to HPSG (Kasper, Calcagno, & Davis, 1998), which aim at addressing the 'free' word order challenges encountered in languages like German, the Slavic language family, and others. Again, primary emphasis in this and similar efforts is not on the engineering and scaling aspects, but on advancing the underlying linguistic theory.

This is very different in the work carried out within the multi-site OVIS (public transport information system) project at the University of Groningen (The Netherlands); van Noord, Bouma, Koeling, & Nederhof (1999) demonstrate that a robust analysis component based on a linguistically sophisticated grammar (inspired by HPSG) can compete with a probabilistic, 'data-oriented' (DOP) parser. In fact, for the limited OVIS domain, the grammatical analysis module outperforms the shallow processor in both accuracy and its demand for computational resources. This is made possible by, among others, restricting the linguistic formalism to a subset of definite clause grammar (DCG), specialized and robust word graph (pre-)processing, and thorough parser engineering (van Noord, 1997).

We have seen comparatively few reports on (the use or extension of) systems like ALEP, CUF, ProFit, or TFS for several years, although these platforms doubtlessly

[11] Incidentally, Gerald Penn, one of the developers of the Attribute Logic Engine (ALE), was based at Tübingen University until recently where he was engaged in an extension to ALE (called TRALE) that integrates a restricted amount of constraint resolution of general implicational constraints at run-time from ConTroll into an efficient logic programming and grammar parsing and generation implementation. TRALE development has not been completed yet; see '`http://www.sfs.nphil.uni-tuebingen.de/~gpenn/ale.html`'.

continue to be used in research and educational environments. Thus, it appears that the wealth of HPSG-related projects and approaches observed in the early 1990s has in the meantime coalesced into a smaller number of synchronized and focussed (and in some cases comparatively large) research and development initiatives. Our own experience strongly suggests that this tendency of convergence can be beneficial both to consortium members and to the wider community.

Perhaps the closest similarity to the work reported in this Special Issue can be found in a development within the LFG community, where the Lisp-based, only moderately efficient Grammar Writers Workbench has effectively been replaced with a very efficient reimplementation in ANSI C, the Xerox Linguistic Environment (XLE) developed at Xerox PARC. The design and realization of the XLE was guided by extensive grammar engineering and system implementation experience; restricting the LFG formalism somewhat and re-engineering of central algorithms resulted in a net speed-up of more than an order of magnitude. The XLE platform has facilitated the development of parallel, large-scale grammars for English, French, and German (Butt, King, Niño, & Segond, 1999) (with other languages underway) and has—by virtue of its previously unmatched processing performance—enhanced and energized language engineering work in LFG.

From the limited parser performance data presented in Butt et al. (1999) it seems that the XLE performs on a scale broadly equivalent to the current best system(s) within our consortium (see Callmeier, this volume): medium-complexity input of ten to twenty words, say, is analysed in average parsing times of around or less than one second per sentence. Obviously, more detailed and systematic comparison will be required between the two frameworks.

5 The Virtual Appendix

Besides the Appendix that gives a mathematical summary of the typed feature formalism assumed throughout the volume (Copestake, this volume), this Special Issue provides a second, virtual appendix that is not included in the printed distribution. The virtual appendix gives access to the raw data collections (profiles) used in the individual contributions to the volume; profiles are available for public download from the following address

`http://www.coli.uni-sb.de/nlesi/`

Complete raw data is provided for interested readers who want to study a specific property or aspect of the profiles in more detail than can be given as part of the manuscripts. Additionally, the collection of on-line profiles may facilitate comparison across manuscripts (i.e. between different systems and techniques), beyond the relative assessments that some of the authors already give. Although profile inspection, analysis, and comparison may be simplified using the [incr tsdb()] software package (see above), the data is represented in ASCII files, suitable for manipulation using standard text processing utilities (like, for example, `grep(1)`, `wc(1)`, `awk(1)` and others).

Acknowledgements

The guest editors for this Special Issue are very grateful to authors and external reviewers, who have all demonstrated an outstanding degree of cooperativity, attention to detail, and flexibility. This has enabled us to complete camera-ready production for the volume within a fairly ambitious, tight schedule of only seven months. Roberto Garigliano and Michelle Hickey served as our JNLE contacts. We are especially indebted to John Carroll, a JNLE editorial board member, who brokered our initial arrangement with JNLE and has helped in producing this volume in many ways. Ulrich Callmeier helpfully supplied the LaTeX macros used in volume-internal bibliographic references.

From the eight papers we had solicited, five were accepted as a result of external reviewing. For one manuscript, authors were invited to revise and resubmit the paper; reviewing of the revised version allowed us to include this sixth contribution in the end. We would like to thank the external reviewers who have greatly helped us in shaping this volume; we were very pleased with the quality, accuracy, and level of detail in the reviews that we received. External reviewing for manuscripts in this collection was performed by:

- Gosse Bouma, Groningen
- Bob Carpenter, SpeechWorks
- David Carter, Speech Machines
- Martin Emele, Stuttgart
- Gregor Erbach, FTW Vienna
- Mark Gawron, SRI Stanford
- Tibor Kiss, Bochum
- Suresh Manandhar, York
- Guido Minnen, Sussex
- Gertjan van Noord, Groningen
- Gerald Penn, Bell Laboratories
- Hadar Shemtov, Xerox Parc
- Koiti Hasida, ETL Tsukuba

References

Brachman, R. J., & Schmolze, J. G. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, *9*, 171 – 216.

Butt, M., King, T. H., Niño, M.-E., & Segond, F. (1999). *A grammar writer's cookbook.* Stanford, CA: CSLI Publications.

Carpenter, B. (1992). *The logic of typed feature structures.* Cambridge, UK: Cambridge University Press.

Carroll, J., Copestake, A., Flickinger, D., & Poznanski, V. (1999). An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation* (pp. 86 – 95). Toulouse, France.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, *2*, 173 – 167.

Ciortuz, L. (2000). *Compiling HPSG into C* (DFKI Research Report). Saarbrücken, Germany: Deutsches Forschungszentrum für Künstliche Intelligenz GmbH.

Copestake, A. (1992). The ACQUILEX LKB. Representation issues in semi-automatic acquisition of large lexicons. In *Proceedings of the 3rd ACL Conference on Applied Natural Language Processing* (pp. 88 – 96). Trento, Italy.

Dalrymple, M., Kaplan, R. M., Maxwell III, J. T., & Zaenen, A. (Eds.). (1995). *Formal issues in Lexical-Functional Grammar.* Stanford, CA: Cambridge University Press.

Emele, M. C. (1994). The typed feature structure representation formalism. In *Proceedings of the International Workshop on Sharable Natural Language Resources.* Nara, Japan.

Erbach, G., Kraan, M. van der, Manandhar, S., Ruessink, H., Thiersch, C., & Skut, W. (1995). Extending unification formalisms. In *Proceedings of the 2nd Language Engineering Convention.* London, UK.

Flickinger, D., Nerbonne, J., Sag, I. A., & Wassow, T. (1987). *Toward evaluation of NLP systems* (Technical Report). Hewlett-Packard Laboratories. (Distributed at the 24th Annual Meeting of the Association for Computational Linguistics)

Götz, T., & Meurers, W. D. (1997). The ConTroll system as large grammar development platform. In *Proceedings of the Workshop on Computational Environments for Grammar Development and Linguistic Engineering* (pp. 38–45). Madrid, Spain.

Joshi, A. K. (1987). An introduction to Tree Adjoining Grammars. In A. Manaster-Ramer (Ed.), *Mathematics of language* (pp. 87–115). Amsterdam, The Netherlands: John Benjamins.

Kasper, R. T., Calcagno, M., & Davis, P. C. (1998). Know when to hold 'em. Shuffling deterministically in a parser for nonconcatenative grammars. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics* (pp. 663–669). Montreal, Canada.

Kay, M. (1973). The MIND system. In R. Randall (Ed.), *Natural language processing* (pp. 155–188). New Yorck, NY: Algorithmic Press.

Kay, M. (1979). Functional grammar. In *Proceedings of the 5th Annual Meeting of the Berkeley Linguistics Society* (pp. 137–144).

Kiefer, B., & Krieger, H.-U. (2000). A context-free approximation of Head-driven Phrase Structure Grammar. In *Proceedings of the 6th International Workshop on Parsing Technologies* (pp. 135–146). Trento, Italy.

Kiefer, B., Krieger, H.-U., Carroll, J., & Malouf, R. (1999). A bag of useful techniques for efficient and robust parsing. In *Proceedings of the 37th Meeting of the Association for Computational Linguistics* (pp. 473–480). College Park, MD.

Krieger, H.-U., & Schäfer, U. (1994). $\mathcal{TDL}$ — A type description language for constraint-based grammars. In *Proceedings of the 15th International Conference on Computational Linguistics* (pp. 893–899). Kyoto, Japan.

Makino, T., Yoshida, M., Torisawa, K., & Tsujii, J. ichi. (1998). LiLFeS — towards a practical HPSG parser. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics* (pp. 807–11). Montreal, Canada.

Neumann, G., & Flickinger, D. (1999). *Learning stochastic lexicalized tree grammars from HPSG* (DFKI Research Report). Saarbrücken, Germany: Deutsches Forschungszentrum für Künstliche Intelligenz GmbH.

van Noord, G. (1997). An efficient implementation of the head-corner parser. *Computational Linguistics*, *23 (3)*, 425–456.

van Noord, G., Bouma, G., Koeling, R., & Nederhof, M.-J. (1999). Robust grammatical analysis for spoken dialogue systems. *Natural Language Engineering, 5 (1)*, 45 – 93.

Oepen, S., & Flickinger, D. P. (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.

Pereira, F. C. N., & Warren, D. H. D. (1980). Definite clause grammars for language analysis. A survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence, 13*, 231 – 278.

Pollard, C., & Sag, I. A. (1987). *Information-based syntax and semantics. Volume 1: Fundamentals.* Chicago, IL and Stanford, CA: Center for the Study of Language and Information. (Distributed by The University of Chicago Press)

Pollard, C., & Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar.* Chicago, IL and Stanford, CA: The Univeristy of Chicago Press and CSLI Publications.

Rounds, W. C., & Kasper, R. T. (1986). A complete logical calculus for record structures representing linguistic information. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science.* Cambridge, MA.

Sells, P. (1985). *Lectures on contemporary syntactic theories.* Stanford, CA: Center for the Study of Language and Information.

Shieber, S. M. (1986). *An introduction to unification-based approaches to grammar.* Stanford, CA: Center for the Study of Language and Information.

Tateisi, Y., Torisawa, K., Miyao, Y., & Tsujii, J. ichi. (1998). Translating the XTAG English grammar to HPSG. In *Proceedings of the 4th Workshop on Tree-adjoining Grammars and Related Frameworks (TAG+)* (pp. 172 – 175). Philadelphia, PA.

Torisawa, K., & Tsujii, J. (1996). Computing phrasal signs in HPSG prior to parsing. In *Proceedings of the 16th International Conference on Computational Linguistics* (pp. 949 – 955). Kopenhagen, Denmark.

Uszkoreit, H., Backofen, R., Busemann, S., Diagne, A. K., Hinkelman, E. A., Kasper, W., Kiefer, B., Krieger, H.-U., Netter, K., Neumann, G., Oepen, S., & Spackman, S. P. (1994). DISCO — an HPSG-based NLP system and its application for appointment scheduling. In *Proceedings of the 15th International Conference on Computational Linguistics.* Kyoto, Japan.

Uszkoreit, H., Becker, T., Backofen, R., Calder, J., Capstick, J., Dini, L., Dörre, J., Erbach, G., Estival, D., Manandhar, S., Mineur, A.-M., van Noord, G., & Oepen, S. (1996). *The EAGLES Formalisms working group. Final report* (Technical Report). Saarbrücken, Germany: Deutsches Forschungszentrum für Künstliche Intelligenz GmbH.

Wahlster, W. (1997). *VerbMobil — Erkennung, Analyse, Transfer, Generierung und Synthese von Spontansprache* (VerbMobil Report # 198). Saarbrücken, Germany: Deutsches Forschungszentrum für Künstliche Intelligenz GmbH.

Woods, W. (1970). Transition network grammars for natural language analysis. *Communications of the ACM, 13*, 591 – 596.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control, 10*, 189 – 208.