Resource sharing among HPSG and LTAG communities by a method of grammar conversion from FB-LTAG to HPSG

Naoki Yoshinaga **Yusuke Miyao**

Department of Information Science, Graduate school of Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, 113-0033, Japan
{yoshinag, yusuke}@is.s.u-tokyo.ac.jp

Kentaro Torisawa

School of Information Science, Japan Advanced Institute of Science and Technology
Asahidai 1-1, Tatsunokuchi-cho, Noumi-gun, Ishikawa, 923-1292, Japan
Information and Human Behavior, PRESTO, Japan Science and Technology Corporation
Kawaguchi Hon-cho 4-1-8, Kawaguchi-shi, Saitama, 332-0012, Japan
torisawa@jaist.ac.jp

Jun'ichi Tsujii

Department of Computer Science, Graduate school of Information Science and Technology, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, 113-0033, Japan
CREST, JST (Japan Science and Technology Corporation)
Kawaguchi Hon-cho 4-1-8, Kawaguchi-shi, Saitama, 332-0012, Japan
tsujii@is.s.u-tokyo.ac.jp

Abstract

This paper describes *the RenTAL system*, which enables sharing resources in LTAG and HPSG formalisms by a method of grammar conversion from an FB-LTAG grammar to a strongly equivalent HPSG-style grammar. The system is applied to the latest version of the XTAG English grammar. Experimental results show that the obtained HPSG-style grammar successfully worked with an HPSG parser, and achieved a drastic speed-up against an LTAG parser. This system enables to share not only grammars and lexicons but also parsing techniques.

1 Introduction

This paper describes an approach for sharing resources in various grammar formalisms such as Feature-Based Lexicalized Tree Adjoining Grammar (FB-LTAG[1]) (Vijay-Shanker, 1987; Vijay-Shanker and Joshi, 1988) and Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) by a method of grammar conversion. *The RenTAL system* automatically converts an FB-LTAG grammar into a strongly equivalent HPSG-style grammar (Yoshinaga and Miyao, 2001). Strong equivalence means that both grammars generate exactly equivalent parse results, and that we can share the LTAG grammars and lexicons in HPSG applications. Our system can reduce considerable workload to develop a huge resource (grammars and lexicons) from scratch.

Our concern is, however, not limited to the sharing of grammars and lexicons. Strongly equivalent grammars enable the sharing of *ideas* developed in each formalism. There have been many studies on *parsing techniques* (Poller and Becker, 1998; Flickinger et al., 2000), ones on *disambiguation models* (Chiang, 2000; Kanayama et al., 2000), and ones on *programming/grammar-development environ-*

[1] In this paper, we use the term LTAG to refer to FB-LTAG, if not confusing.

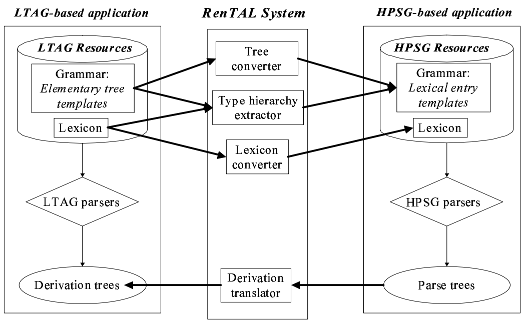

Figure 1: The RenTAL System: Overview

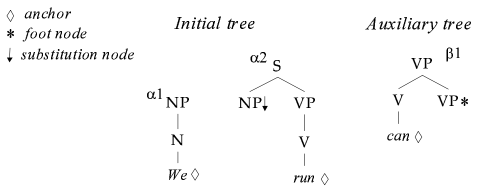

Figure 2: Elementary trees

ment (Sarkar and Wintner, 1999; Doran et al., 2000; Makino et al., 1998). These works are restricted to each closed community, and the relation between them is not well discussed. Investigating the relation will be apparently valuable for both communities.

In this paper, we show that the strongly equivalent grammars enable the sharing of "*parsing techniques*", which are dependent on each computational framework and have never been shared among HPSG and LTAG communities. We apply our system to the latest version of the XTAG English grammar (The XTAG Research Group, 2001), which is a large-scale FB-LTAG grammar. A parsing experiment shows that an efficient HPSG parser with the obtained grammar achieved a significant speed-up against an existing LTAG parser (Yoshinaga et al., 2001). This result implies that parsing techniques for HPSG are also beneficial for LTAG parsing. We can say that the grammar conversion enables us to share HPSG parsing techniques in LTAG parsing.

Figure 1 depicts a brief sketch of the RenTAL system. The system consists of the following four modules: *Tree converter*, *Type hierarchy extractor*, *Lexicon converter* and *Derivation translator*. The tree converter module is a core module of the system, which is an implementation of the grammar conversion algorithm given in Section 3. The type hierarchy extractor module extracts the symbols of the node, features, and feature values from the LTAG elementary tree templates and lexicon, and construct the type hierarchy from them. The lexicon converter module converts LTAG elementary tree templates into HPSG lexical entries. The derivation translator module takes HPSG parse trees, and map them to LTAG derivation trees. All modules other than the last one are related to the conversion process from LTAG into HPSG, and the last one enables to obtain LTAG analysis from the obtained HPSG analysis.

Tateisi et al. also translated LTAG into HPSG (Tateisi et al., 1998). However, their method depended on translator's intuitive analysis of the original grammar. Thus the translation was manual and grammar dependent. The manual translation demanded considerable efforts from the translator, and obscures the equivalence between the original and obtained grammars. Other works (Kasper et al., 1995; Becker and Lopez, 2000) convert HPSG grammars into LTAG grammars. However, given the greater expressive power of HPSG, it is impossible to convert an arbitrary HPSG grammar into an LTAG grammar. Therefore, a conversion from HPSG into LTAG often requires some restrictions on the HPSG grammar to suppress its generative capacity. Thus, the conversion loses the equivalence of the grammars, and we cannot gain the above advantages.

Section 2 reviews the source and the target grammar formalisms of the conversion algorithm. Section 3 describes the conversion algorithm which the core module in the RenTAL system uses. Section 4 presents the evaluation of the RenTAL system through experiments with the XTAG English grammar. Section 5 concludes this study and addresses future works.

2 Background

2.1 Feature-Based Lexicalized Tree Adjoining Grammar (FB-LTAG)

LTAG (Schabes et al., 1988) is a grammar formalism that provides syntactic analyses for a sentence by composing *elementary trees* with two opera-

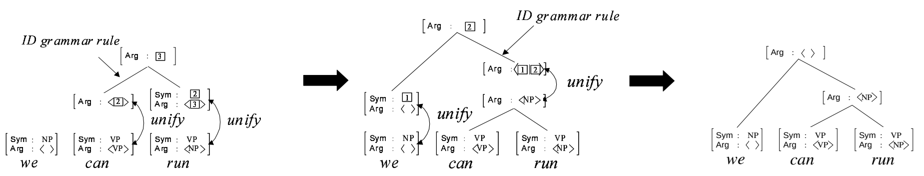

Figure 6: Parsing with an HPSG grammar

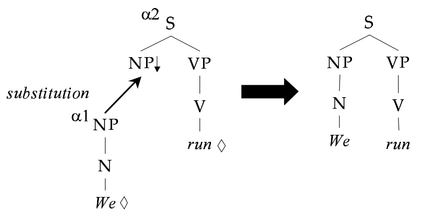

Figure 3: Substitution

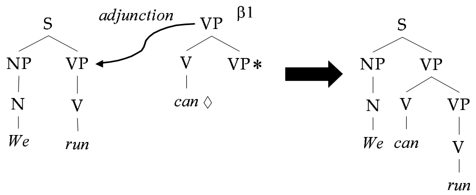

Figure 4: Adjunction

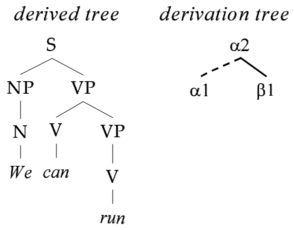

Figure 5: Derived trees and derivation trees

tions called *substitution* and *adjunction*. Elementary trees are classified into two types, *initial trees* and *auxiliary trees* (Figure 2). An elementary tree has at least one leaf node labeled with a terminal symbol called an *anchor* (marked with $\diamond$). In an auxiliary tree, one leaf node is labeled with the same symbol as the root node and is specially marked as a *foot node* (marked with $*$). In an elementary tree, leaf nodes with the exception of anchors and the foot node are called *substitution nodes* (marked with $\downarrow$).

Substitution replaces a substitution node with another initial tree (Figure 3). Adjunction grafts an auxiliary tree with the root node and foot node labeled x onto an internal node of another tree with the same symbol x (Figure 4). FB-LTAG (Vijay-Shanker, 1987; Vijay-Shanker and Joshi, 1988) is an extension of the LTAG formalism. In FB-LTAG, each node in the elementary trees has a feature structure, containing grammatical constraints on the node. Figure 5 shows a result of LTAG analysis, which is described not only by *derived trees* (i.e., parse trees) but also by *derivation trees*. A derivation tree is a structural description in LTAG and represents the history of combinations of elementary trees.

There are several grammars developed in the FB-LTAG formalism, including the XTAG English grammar, a large-scale grammar for English (The XTAG Research Group, 2001). The XTAG group (Doran et al., 2000) at the University of Pennsylvania is also developing Korean, Chinese, and Hindi grammars. Development of a large-scale French grammar (Abeillé and Candito, 2000) has also started at the University of Pennsylvania and is expanded at University of Paris 7.

2.2 Head-Driven Phrase Structure Grammar (HPSG)

An HPSG grammar consists of *lexical entries* and *ID grammar rules*, each of which is described with typed feature structures (Carpenter, 1992). A lexical entry for each word expresses the characteristics of the word, such as the subcategorization frame and the grammatical category. An ID grammar rule represents a relation between a mother and its daughters, and is independent of lexical characteristics. Figure 6 illustrates an example of bottom-up parsing with an HPSG grammar. First, lexical entries for "*can*" and "*run*" are unified respectively with the daughter feature structures of

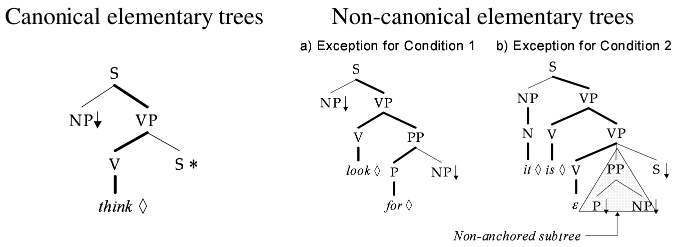

Figure 7: A canonical elementary tree and exceptions

an ID grammar rule. The feature structure of the mother node is determined as a result of these unifications. The center of Figure 6 shows a rule application to "*can run*" and "*we*".

There are a variety of works on efficient parsing with HPSG, which allow the use of HPSG-based processing in practical application contexts (Flickinger et al., 2000). Stanford University is developing the English Resource Grammar, an HPSG grammar for English, as a part of the Linguistic Grammars Online (LinGO) project (Flickinger, 2000). In practical context, German, English, and Japanese HPSG-based grammars are developed and used in the Verbmobil project (Kay et al., 1994). Our group has developed a wide-coverage HPSG grammar for Japanese (Mitsuishi et al., 1998), which is used in a high-accuracy Japanese dependency analyzer (Kanayama et al., 2000).

3 Grammar conversion

The grammar conversion from LTAG to HPSG (Yoshinaga and Miyao, 2001) is the core portion of the RenTAL system. The conversion algorithm consists of:

1. Conversion of *canonical elementary trees* to HPSG lexical entries.
2. Definition of ID grammar rules to emulate substitution and adjunction.
3. Conversion of non-canonical elementary trees to canonical ones.

The left-hand side of Figure 7 shows a canonical elementary tree, which satisfies the following conditions:

Condition 1 A tree must have only one anchor.

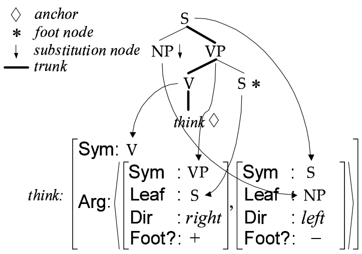

Figure 8: A conversion from a canonical elementary tree into an HPSG lexical entry

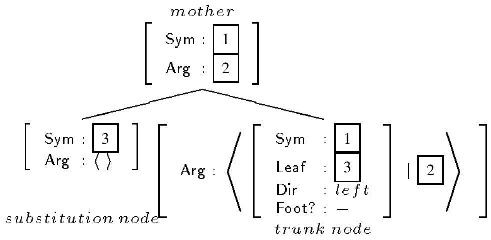

Figure 9: Left substitution rule

Condition 2 All branchings in a tree must contain trunk nodes.

Trunk nodes are nodes on a *trunk*, which is a path from an anchor to the root node (the thick lines in Figure 7) (Kasper et al., 1995). Condition 1 guarantees that a canonical elementary tree has only one trunk, and Condition 2 guarantees that each branching consists of a trunk node, a leaf node, and their mother (also a trunk node). The right-hand side of Figure 7 shows elementary trees violating the conditions.

Canonical elementary trees can be directly converted to HPSG lexical entries by regarding each leaf node as a subcategorization element of the anchor, and by encoding them into a list. Figure 8 shows an example of the conversion. By following the trunk from the anchor "*think*" to the

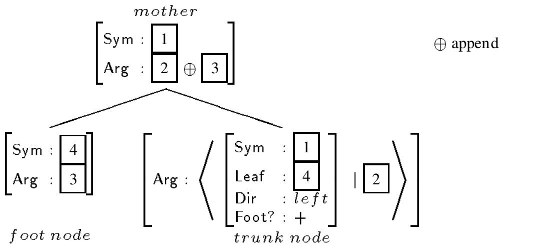

Figure 10: Left adjunction rule

root node labeled S, we store each branching in a list. As shown in Figure 8, each branching is specified by a leaf node and the mother node. A feature Sym represents the non-terminal symbol of the mother node. Features Leaf, Dir, Foot? represent the leaf node; the non-terminal symbol, the direction (on which side of the trunk node the leaf node is), and the type (whether a foot node or a substitution node), respectively.

Figures 9 and 10 show ID grammar rules to emulate substitution and adjunction. These grammar rules are independent of the original grammar because they don't specify any characteristics specific to the original grammar.

In the substitution rule, the Sym feature of the substitution node must have the value of the Leaf feature $\boxed{3}$ of the trunk node. The Arg feature of the substitution node must be a null list, because the substitution node must be unified only with the node corresponding to the root node of the initial tree. The substitution rule percolates the tail elements $\boxed{2}$ of the Arg feature of a trunk node to the mother in order to continue constructing the tree.

In the adjunction rule, the Sym feature of a foot node must have the same value as the Leaf feature $\boxed{4}$. The value of the Arg feature of the mother node is a concatenation list of both Arg features $\boxed{2}$ and $\boxed{3}$ of its daughters because we first construct the tree corresponding to the adjoining tree and next continue constructing the tree corresponding to the adjoined tree. The value "$+$" or "$-$" of the Foot? feature explicitly determines whether the next rule application is the adjunction rule or the substitution rule.

Figure 11 shows an instance of rule applications. The thick line indicates the adjoined tree ($\alpha 1$) and the dashed line indicates the adjoining

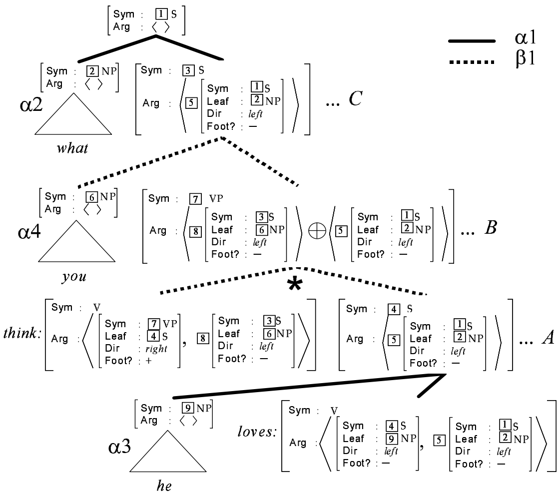

Figure 11: An example of rule applications

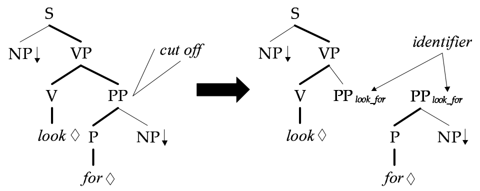

Figure 12: Division of a multi-anchored elementary tree into single-anchored trees

tree ($\beta 2$). The adjunction rule is applied to construct the branching marked with $\star$, where "*think*" takes as an argument a node whose Sym feature's value is S. By applying the adjunction rule, the Arg feature of the mother node (B) becomes a concatenation list of both Arg features of $\beta 1$ ($\boxed{8}$) and $\alpha 1$ ($\boxed{5}$). Note that when the construction of $\beta 1$ is completed, the Arg feature of the trunk node (C) will be its former state (A). We can continue constructing $\alpha 1$ as if nothing had happened.

Multi-anchored elementary trees, which violate Condition 1, are divided into multiple canonical elementary trees. We call the cutting nodes in the divided trees *cut-off nodes* (Figure 12). Note that a cut-off node is marked by an *identifier* to preserve a co-occurrence relation among the multiple anchors. Figure 12 shows an example of the conversion of a multi-anchored elementary tree for a compound expression "*look for*". We first select an anchor "*look*" as the syntactic head, and traverse the tree along the trunk from the root node S to the anchor "*look*". We then cut off the multi-

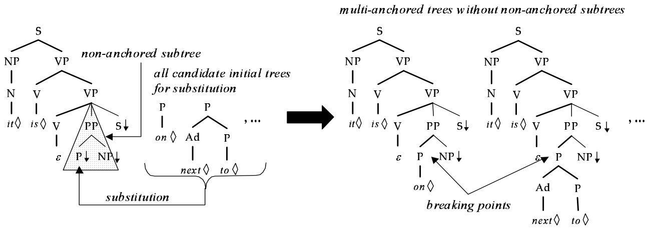

Figure 13: Combination of a non-anchored subtree into anchored trees

anchored elementary tree at the node PP, and cut-off nodes PP in resulting single-anchored trees are marked by an identifier $look\_for$.

Non-canonical elementary trees violating Condition 2 have a *non-anchored subtree* which is a subtree of depth 1 or above with no anchor. A non-anchored subtree is converted into multi-anchored trees by substituting the deepest node (Figure 13). Substituted nodes are marked as *breaking points* to remember that the nodes originate from the substitution nodes. In the resulting trees, all subtrees are anchored so that we can apply the above conversion algorithms. Figure 13 shows a conversion of a non-canonical elementary tree for *it-cleft*. A substitution node P in the non-anchored subtree is selected, and is substituted by each initial tree. The substituted node P in resulting multi-anchored trees are marked as breaking points.

The above algorithm gives the conversion of LTAG, and it can be easily extended to handle an FB-LTAG grammar by merely storing a feature structure of each node into the Sym feature and Leaf feature together with the non-terminal symbol. Feature structure unification is executed by ID grammar rules.

The strong equivalence is assured because only substitution/adjunction operations performed in LTAG are performed with the obtained HPSG-style grammar. This is because each element in the Arg feature selects only feature structures corresponding to trees which can substitute/be adjoined by each leaf node of an elementary tree. By following a history of rule applications, each combination of elementary trees in LTAG derivation trees can be readily recovered. The strong equivalence holds also for conversion of non-canonical elementary trees. For trees violating Condition 1, we can distinguish the cut-off nodes from the substitution nodes owing to identifiers, which recover the co-occurrence relation in the original elementary trees between the divided trees. For trees violating Condition 2, we can identify substitution nodes in a combined tree because they are marked as breaking points, and we can consider the combined tree as two trees in the LTAG derivation.

Table 1: The classification of elementary tree templates in the XTAG English grammar (LTAG) and converted lexical entry templates corresponding to them (HPSG): $\mathcal{A}$: canonical elementary trees, $\mathcal{B}$: elementary trees violating only Condition 1, $\mathcal{C}$: elementary trees violating only Condition 2, $\mathcal{D}$: elementary trees violating both conditions

Grammar	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	**Total**
LTAG	326	764	54	50	1,194
HPSG	326	1,992	1,083	2,474	5,875

4 Experiments

The RenTAL system is implemented in *LiLFeS* (Makino et al., 1998)[2]. LiLFeS is one of the fastest inference engines for processing feature structure logic, and efficient HPSG parsers have already been built on this system (Nishida et al., 1999; Torisawa et al., 2000). We applied our system to the XTAG English grammar (The XTAG Research Group, 2001)[3], which is a large-scale FB-LTAG grammar for English.

[2]The RenTAL system is available at: http://www-tsujii.is.s.u-tokyo.ac.jp/rental/

[3]We used the grammar attached to the latest distribution of an LTAG parser which we used for the parsing experiment. The parser is available at: ftp://ftp.cis.upenn.edu/pub/xtag/lem/lem-0.13.0.i686.tgz

Table 2: Parsing performance with the XTAG English grammar for the ATIS corpus.

Parser	Parse Time (sec.)
lem	19.64
TNT	0.77

The XTAG English grammar consists of 1,194[4] elementary tree templates and around 45,000 lexical items[5]. We successfully converted all the elementary tree templates in the XTAG English grammar to HPSG lexical entry templates. Table 1 shows the classifications of elementary tree templates of the XTAG English grammar, according to the conditions we introduced in Section 3, and also shows the number of corresponding HPSG lexical entry templates. Conversion took about 25 minutes CPU time on a 700 Mhz Pentium III Xeon with four gigabytes main memory.

The original and the obtained grammar generated exactly the same number of derivation trees in the parsing experiment with 457 sentences from the ATIS corpus (Marcus et al., 1994)[6] (the average length is 6.32 words). This result empirically attested the strong equivalence of our algorithm.

Table 2 shows the average parsing time with the LTAG and HPSG parsers. In Table 2, *lem* refers to the LTAG parser (Sarkar et al., 2000), ANSI C implementation of the two-phase parsing algorithm that performs the head corner parsing (van Noord, 1994) without features (phase 1), and then executes feature unification (phase 2). *TNT* refers to the HPSG parser (Torisawa et al., 2000), C++ implementation of the two-phase parsing algorithm that performs filtering with a compiled CFG (phase 1) and then executes feature unification (phase 2). Table 2 clearly shows that the HPSG parser is significantly faster than the LTAG parser. This result implies that parsing techniques for HPSG are also beneficial for LTAG parsing. We can say that the grammar conversion enables us to share HPSG parsing techniques in LTAG parsing. Another paper (Yoshinaga et al., 2001) describes the detailed analysis on the factor of the difference of parsing performance.

[4]We eliminated 32 elementary trees because the LTAG parser cannot produce correct derivation trees with them.

[5]These lexical items are a subset of the original XTAG English grammar distribution.

[6]We eliminated 59 sentences because of a time-out of the parsers, and 61 sentences because the LTAG parser does not produce correct derivation trees because of bugs in its preprocessor.

5 Conclusion

We described the RenTAL system, a grammar converter from FB-LTAG to HPSG. The grammar conversion guarantees the strong equivalence, and hence we can obtain an HPSG-style grammar equivalent to existing LTAG grammars. Experimental result showed that the system enabled to share not only LTAG grammars, but also HPSG parsing techniques. This system will enable a variety of resource sharing such as the sharing of the programming/grammar-development environment (Makino et al., 1998; Sarkar and Wintner, 1999) and grammar extraction methods from bracketed corpora (Xia, 1999; Chen and Vijay-Shanker, 2000; Neumann, 1998). Although our system connects only FB-LTAG and HPSG, we believe that our approach can be extended to other formalisms such as Lexical-Functional Grammar (Kaplan and Bresnan, 1982).

Acknowledgment The authors are indebted to Mr. Anoop Sarkar for his help in using his parser in our experiment. The authors would like to thank anonymous reviewers for their valuable comments and criticisms on this paper.

References

Anne Abeillé and Marie-Hélène Candito. 2000. FTAG: A Lexicalized Tree Adjoining Grammar for French. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars: Formal, Computational and Linguistic Aspects*, pages 305–329. CSLI publications.

Tilman Becker and Patrice Lopez. 2000. Adapting HPSG-to-TAG compilation to wide-coverage grammars. In *Proc. of TAG+5*, pages 47–54.

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

John Chen and K. Vijay-Shanker. 2000. Automated extraction of TAGs from the Penn Treebank. In *Proc. of IWPT 2000*.

David Chiang. 2000. Statistical parsing with an automatically-extracted Tree Adjoining Grammar. In *Proc. of ACL 2000*, pages 456–463.

Christy Doran, Beth Ann Hockey, Anoop Sarkar, B. Srinivas, and Fei Xia. 2000. Evolution of the XTAG system. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars: Formal, Computational and Linguistic Aspects*, pages 371–403. CSLI publications.

Dan Flickinger, Stephen Oepen, Jun'ichi Tsujii, and Hans Uszkoreit, editors. 2000. *Natural Language Engineering – Special Issue on Efficient Processing with HPSG: Methods, Systems, Evaluation.* Cambridge University Press.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering – Special Issue on Efficient Processing with HPSG: Methods, Systems, Evaluation*, 6(1):15–28.

Hiroshi Kanayama, Kentaro Torisawa, Yutaka Mitsuisi, and Jun'ichi Tsujii. 2000. Hybrid Japanese parser with hand-crafted grammar and statistics. In *Proc. of COLING 2000*, pages 411–417.

Ronald Kaplan and Joan Bresnan. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. The MIT Press.

Robert Kasper, Bernd Kiefer, Klaus Netter, and K. Vijay-Shanker. 1995. Compilation of HPSG to TAG. In *Proc. of ACL '94*, pages 92–99.

M. Kay, J. Gawron, and P. Norvig. 1994. *Verbmobil: A Translation System for Face-to-Face Dialog*. CSLI Publications.

Takaki Makino, Minoru Yoshida, Kentaro Torisawa, and Jun'ichi Tsujii. 1998. LiLFeS — towards a practical HPSG parsers. In *Proc. of COLING–ACL '98*, pages 807–811.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Yutaka Mitsuishi, Kentaro Torisawa, and Jun'ichi Tsujii. 1998. HPSG-style underspecified Japanese grammar with wide coverage. In *Proc. of COLING–ACL '98*, pages 876–880.

Güter Neumann. 1998. Automatic extraction of stochastic lexcalized tree grammars from treebanks. In *Proc. of TAG+4*, pages 120–123.

Kenji Nishida, Kentaro Torisawa, and Jun'ichi Tsujii. 1999. An efficient HPSG parsing algorithm with array unification. In *Proc. of NLPRS '99*, pages 144–149.

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

Peter Poller and Tilman Becker. 1998. Two-step TAG parsing revisited. In *Proc. of TAG+4*, pages 143–146.

Anoop Sarkar and Shuly Wintner. 1999. Typing as a means for validating feature structures. In *Proc.of CLIN '99*, pages 159–167.

Anoop Sarkar, Fei Xia, and Aravind Joshi. 2000. Some experiments on indicators of parsing complexity for lexicalized grammars. In *Proc. of COLING 2000*, pages 37–42.

Yves Schabes, Anne Abeille, and Aravind K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proc. of 12th COLING '92*, pages 578–583.

Yuka Tateisi, Kentaro Torisawa, Yusuke Miyao, and Jun'ichi Tsujii. 1998. Translating the XTAG English grammar to HPSG. In *Proc. of TAG+4*, pages 172–175.

The XTAG Research Group. 2001. A Lexicalized Tree Adjoining Grammar for English. http://www.cis.upenn.edu/~xtag/.

Kentaro Torisawa, Kenji Nishida, Yusuke Miyao, and Jun'ichi Tsujii. 2000. An HPSG parser with CFG filtering. *Natural Language Engineering – Special Issue on Efficient Processing with HPSG: Methods, Systems, Evaluation*, 6(1):63–80.

Gertjan van Noord. 1994. Head corner parsing for TAG. *Computational Intelligence*, 10(4):525–534.

K. Vijay-Shanker and Aravind K. Joshi. 1988. Feature structures based Tree Adjoining Grammars. In *Proc. of 12th COLING '92*, pages 714–719.

K. Vijay-Shanker. 1987. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, Department of Computer & Information Science, University of Pennsylvania.

Fei Xia. 1999. Extracting Tree Adjoining Grammars from bracketed corpora. In *Proc. of NLPRS '99*, pages 398–403.

Naoki Yoshinaga and Yusuke Miyao. 2001. Grammar conversion from FB-LTAG to HPSG. In *Proc. of ESSLLI 2001 Student Session*. To appear.

Naoki Yoshinaga, Yusuke Miyao, Kentaro Torisawa, and Jun'ichi Tsujii. 2001. Efficient LTAG parsing using HPSG parsers. In *Proc. of PACLING 2001*. To appear.