Comparison between CFG filtering techniques for LTAG and HPSG

Naoki Yoshinaga†
† University of Tokyo
7-3-1 Hongo, Bunkyo-ku,
Tokyo, 113-0033, Japan
yoshinag@is.s.u-tokyo.ac.jp

Kentaro Torisawa‡
‡ Japan Advanced Institute
of Science and Technology
1-1 Asahidai, Tatsunokuchi,
Ishikawa, 923-1292, Japan
torisawa@jaist.ac.jp

Jun'ichi Tsujii†∗
∗ CREST, JST (Japan Science
and Technology Corporation)
Hon-cho 4-1-8, Kawaguchi-shi,
Saitama, 332-0012, Japan
tsujii@is.s.u-tokyo.ac.jp

Abstract

An empirical comparison of CFG filtering techniques for LTAG and HPSG is presented. We demonstrate that an approximation of HPSG produces a more effective CFG filter than that of LTAG. We also investigate the reason for that difference.

1 Introduction

Various parsing techniques have been developed for lexicalized grammars such as Lexicalized Tree Adjoining Grammar (LTAG) (Schabes et al., 1988), and Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994). Along with the independent development of parsing techniques for individual grammar formalisms, some of them have been adapted to other formalisms (Schabes et al., 1988; van Noord, 1994; Yoshida et al., 1999; Torisawa et al., 2000). However, these realizations sometimes exhibit quite different performance in each grammar formalism (Yoshida et al., 1999; Yoshinaga et al., 2001). If we could identify an algorithmic difference that causes performance difference, it would reveal advantages and disadvantages of the different realizations. This should also allow us to integrate the advantages of the realizations into one generic parsing technique, which yields the further advancement of the whole parsing community.

In this paper, we compare CFG filtering techniques for LTAG (Harbusch, 1990; Poller and Becker, 1998) and HPSG (Torisawa et al., 2000; Kiefer and Krieger, 2000), following an approach to parsing comparison among different grammar formalisms (Yoshinaga et al., 2001). The key idea of the approach is to use *strongly equivalent grammars*, which generate equivalent parse results for the same input, obtained by a grammar conversion as demonstrated by Yoshinaga and Miyao (2001). The parsers with CFG filtering predict possible parse trees by a CFG approximated from a given grammar. Comparison of those parsers are interesting because effective CFG filters allow us to bring the empirical time complexity of the parsers close to that of CFG parsing. Investigating the difference between the ways of context-free (CF) approximation of LTAG and HPSG will thereby enlighten a way of further optimization for both techniques.

We performed a comparison between the existing CFG filtering techniques for LTAG (Poller and Becker, 1998) and HPSG (Torisawa et al., 2000), using strongly equivalent grammars obtained by converting LTAGs extracted from the Penn Treebank (Marcus et al., 1993) into HPSG-style. We compared the parsers with respect to the size of the approximated CFG and its effectiveness as a filter.

2 Background

In this section, we introduce a grammar conversion (Yoshinaga and Miyao, 2001) and CFG filtering (Harbusch, 1990; Poller and Becker, 1998; Torisawa et al., 2000; Kiefer and Krieger, 2000).

2.1 Grammar conversion

The grammar conversion consists of a conversion of LTAG elementary trees to HPSG lexical entries and an emulation of substitution and adjunction by

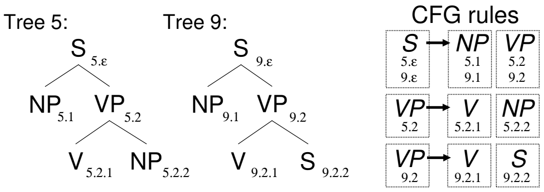

Figure 1: Extraction of CFG from LTAG

pre-determined grammar rules. An LTAG elementary tree is first converted into *canonical elementary trees* which have only one anchor and whose subtrees of depth n (≥ 1) contain at least one anchor. A canonical elementary tree is then converted into an HPSG lexical entry by regarding the leaf nodes as arguments and by storing them in a stack.

We can perform a comparison between LTAG and HPSG parsers using strongly equivalent grammars obtained by the above conversion. This is because strongly equivalent grammars can be a substitute for the same grammar in different grammar formalisms.

2.2 CFG filtering techniques

An initial offline step of CFG filtering is performed to approximate a given grammar with a CFG. The obtained CFG is used as an efficient device to compute the necessary conditions for parse trees.

The CFG filtering generally consists of two steps. In phase 1, the parser first predicts possible parse trees using the approximated CFG, and then filters out irrelevant edges by a top-down traversal starting from roots of successful context-free derivations. In phase 2, it then eliminates invalid parse trees by using constraints in the given grammar. We call the remaining edges that are used for the phase 2 parsing *essential edges*.

The parsers with CFG filtering used in our experiments follow the above parsing strategy, but are different in the way the CF approximation and the elimination of impossible parse trees in phase 2 are performed. In the following sections, we briefly describe the CF approximation and the elimination of impossible parse trees in each realization.

2.2.1 CF approximation of LTAG

In CFG filtering techniques for LTAG (Harbusch, 1990; Poller and Becker, 1998), every branching of elementary trees in a given grammar is extracted as a CFG rule as shown in Figure 1.

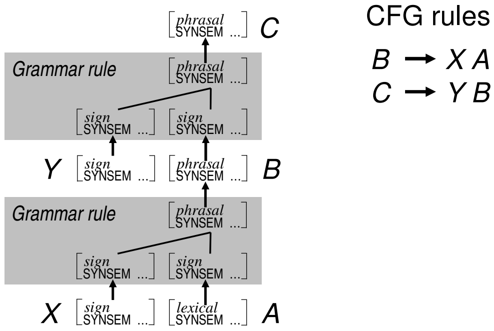

Figure 2: Extraction of CFG from HPSG

Because the obtained CFG can reflect only local constraints given in each local structure of the elementary trees, it generates invalid parse trees that connect local trees in different elementary trees. In order to eliminate such parse trees, a link between branchings is preserved as *a node number* which records a unique node address (a subscript attached to each node in Figure 1). We can eliminate these parse trees by traversing essential edges in a bottom-up manner and recursively propagating *ok-flag* from a node number x to a node number y when a connection between x and y is allowed in the LTAG grammar. We call this propagation *ok-prop*.

2.2.2 CF approximation of HPSG

In CFG filtering techniques for HPSG (Torisawa et al., 2000; Kiefer and Krieger, 2000), the extraction process of a CFG from a given HPSG grammar starts by recursively instantiating daughters of a grammar rule with lexical entries and generated feature structures until new feature structures are not generated as shown in Figure 2. We must impose restrictions on values of some features (*i.e.*, ignoring them) and/or the number of rule applications in order to guarantee the termination of the rule application. A CFG is obtained by regarding each initial and generated feature structures as nonterminals and transition relations between them as CFG rules.

Although the obtained CFG can reflect local and global constraints given in the whole structure of lexical entries, it generates invalid parse trees because they do not reflect upon constraints given by the values of features that are ignored in phase 1. These parse trees are eliminated in phase 2 by applying a grammar rule that corresponds to the applied CFG rule. We call this rule application *rule-app*.

Table 1: The size of extracted LTAGs (tree templates) and approximated CFGs (above: the number of nonterminals; below: the number of rules)

Grammar	G_2	$G_{2\text{-}4}$	$G_{2\text{-}6}$	$G_{2\text{-}8}$	$G_{2\text{-}10}$	$G_{2\text{-}21}$
LTAG	1,488	2,412	3,139	3,536	3,999	6,085
CFG_{PB}	65	66	66	66	67	67
	716	954	1,090	1,158	1,229	1,552
CFG_{TNT}	1,989	3,118	4,009	4,468	5,034	7,454
	18,323	35,541	50,115	58,356	68,239	118,464

Table 2: Parsing performance (sec.) with the strongly equivalent grammars for Section 2 of WSJ

Parser	G_2	$G_{2\text{-}4}$	$G_{2\text{-}6}$	$G_{2\text{-}8}$	$G_{2\text{-}10}$	$G_{2\text{-}21}$
PB	1.4	9.1	17.4	24.0	34.2	124.3
TNT	0.044	0.097	0.144	0.182	0.224	0.542

3 Comparison with CFG filtering

In this section, we compare a pair of CFG filtering techniques for LTAG (Poller and Becker, 1998) and HPSG (Torisawa et al., 2000) described in Section 2.2.1 and 2.2.2. We hereafter refer to *PB* and *TNT* for the C++ implementations of the former and a valiant[1] of the latter, respectively.[2]

We first acquired LTAGs by a method proposed in Miyao et al. (2003) from Sections 2-21 of the Wall Street Journal (WSJ) in the Penn Treebank (Marcus et al., 1993) and its subsets.[3] We then converted them into strongly equivalent HPSG-style grammars using the grammar conversion described in Section 2.1. Table 1 shows the size of CFG approximated from the strongly equivalent grammars. G_x, CFG_{PB}, and CFG_{TNT} henceforth refer to the LTAG extracted from Section x of WSJ and CFGs approximated from G_x by *PB* and *TNT*, respectively. The size of CFG_{TNT} is much larger than that of CFG_{PB}. By investigating parsing performance using these CFGs, we show that the larger size of CFG_{TNT} resulted in better parsing performance.

Table 2 shows the parse time with 254 sentences of length n (≤ 10) from Section 2 of WSJ (the average length is 6.72 words).[4] This result shows not only that *TNT* achieved a drastic speed-up against *PB*, but also that performance difference between them increases with the larger size of the grammars.

In order to estimate the degree of CF approximation, we measured the number of essential (inactive) edges of phase 1. Table 3 shows the number of the essential edges. The number of essential edges produced by *PB* is much larger than that produced by *TNT*. We then investigated the effect on phase 2 as caused by the different number of the essential edges. Table 4 shows the success rate of ok-prop and rule-app. The success rate of rule-app is 100%,[5] whereas that of ok-prop is quite low.[6] These results indicate that CFG_{TNT} is superior to CFG_{PB} with respect to the degree of the CF approximation.

We can explain the reason for this difference by investigating how *TNT* approximates HPSG-style grammars converted from LTAGs. As described in Section 2.1, the grammar conversion preserves the whole structure of each elementary tree (precisely, a canonical elementary tree) in a stack, and grammar rules manipulate a head element of the stack. A generated feature structure in the approximation process thus corresponds to the whole unprocessed portion of a canonical elementary tree. This implies that successful context-free derivations obtained by CFG_{TNT} basically involve elementary trees in which all substitution and adjunction have succeeded. However, CFG_{PB} (also a CFG produced by the other work (Harbusch, 1990)) cannot avoid generating invalid parse trees that connect two lo-

Table 3: The numbers of essential edges with the strongly equivalent grammars for Section 02 of WSJ

Parser	G_2	$G_{2\text{-}4}$	$G_{2\text{-}6}$	$G_{2\text{-}8}$	$G_{2\text{-}10}$	$G_{2\text{-}21}$
PB	791	1,435	1,924	2,192	2,566	3,976
TNT	63	121	174	218	265	536

Table 4: The success rate (%) of phase 2 operations

Operations	G_2	$G_{2\text{-}4}$	$G_{2\text{-}6}$	$G_{2\text{-}8}$	$G_{2\text{-}10}$	$G_{2\text{-}21}$
ok-prop (*PB*)	38.5	34.3	33.1	32.3	31.7	31.0
rule-app (*TNT*)	100	100	100	100	100	100

[1] All daughters of rules are instantiated in the approximation.

[2] In phase 1, *PB* performs Earley (Earley, 1970) parsing while *TNT* performs CKY (Younger, 1967) parsing.

[3] The elementary trees in the LTAGs are binarized.

[4] We used a subset of the training corpus to avoid the complication of using default lexical entries for unknown words.

[5] This means that the extracted LTAGs should be compatible with CFG and were completely converted to CFGs by *TNT*.

[6] Similar results were obtained in preliminary experiments using the XTAG English grammar (The XTAG Research Group, 2001) without features (parse time (sec.)/success rate (%) for *PB* and *TNT* were 15.3/30.6 and 0.606/71.2 with the same sentences), though space limitations preclude complete results.

cal structures where adjunction takes place between them. We measured with $G_{2\text{-}21}$ the proportion of the number of ok-prop between two node numbers of nodes that take adjunction and its success rate. It occupied 87% of the total number of ok-prop and its success rate was only 22%. These results suggest that the global contexts in a given grammar is essential to obtain an effective CFG filter.

It should be noted that the above investigation also tells us another way of CF approximation of LTAG. We first define a unique way of tree traversal such as head-corner traversal (van Noord, 1994) on which we can perform a sequential application of substitution and adjunction. We then recursively apply substitution and adjunction on that traversal to an elementary tree and a generated tree structure. Because the processed portions of generated tree structures are no longer used later, we regard the unprocessed portions of the tree structures as nonterminals of CFG. We can thereby construct another CFG filtering for LTAG by combining this CFG filter with an existing LTAG parsing algorithm (van Noord, 1994).

4 Conclusion and future direction

We presented an empirical comparison of LTAG and HPSG parsers with CFG filtering. We compared the parsers with strongly equivalent grammars obtained by converting LTAGs extracted from the Penn Treebank into HPSG-style. Experimental results showed that the existing CF approximation of HPSG (Torisawa et al., 2000) produced a more effective filter than that of LTAG (Poller and Becker, 1998). By investigating the different ways of CF approximation, we concluded that the global constraints in a given grammar is essential to obtain an effective filter.

We are going to integrate the advantage of the CF approximation of HPSG into that of LTAG in order to establish another CFG filtering for LTAG. We will also conduct experiments on trade-offs between the degree of CF approximation and the size of approximated CFGs as in Maxwell III and Kaplan (1993).

Acknowledgment

We thank Yousuke Sakao for his help in profiling TNT parser and anonymous reviewers for their helpful comments. This work was partially supported by JSPS Research Fellowships for Young Scientists.

References

J. Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 6(8):451–455.

K. Harbusch. 1990. An efficient parsing algorithm for Tree Adjoining Grammars. In *Proc. of ACL*, pages 284–291.

B. Kiefer and H.-U. Krieger. 2000. A Context-Free approximation of Head-Driven Phrase Structure Grammar. In *Proc. of IWPT*, pages 135–146.

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

J. T. Maxwell III and R. M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–590.

Y. Miyao, T. Ninomiya, and J. Tsujii. 2003. Lexicalized grammar acquisition. In *Proc. of EACL companion volume*, pages 127–130.

C. Pollard and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

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

Y. Schabes, A. Abeillé, and A. K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proc. of COLING*, pages 578–583.

The XTAG Research Group. 2001. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-01-03, IRCS, University of Pennsylvania.

K. Torisawa, K. Nishida, Y. Miyao, and J. Tsujii. 2000. An HPSG parser with CFG filtering. *Natural Language Engineering*, 6(1):63–80.

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

M. Yoshida, T. Ninomiya, K. Torisawa, T. Makino, and J. Tsujii. 1999. Efficient FB-LTAG parser and its parallelization. In *Proc. of PACLING*, pages 90–103.

N. Yoshinaga and Y. Miyao. 2001. Grammar conversion from LTAG to HPSG. In *Proc. of ESSLLI Student Session*, pages 309–324.

N. Yoshinaga, Y. Miyao, K. Torisawa, and J. Tsujii. 2001. Efficient LTAG parsing using HPSG parsers. In *Proc. of PACLING*, pages 342–351.

D. H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 2(10):189–208, February.