Compact non-left-recursive grammars using the selective left-corner transform and factoring*

Mark Johnson **Brian Roark**
Cognitive and Linguistic Sciences, Box 1978
Brown University
Mark_Johnson@Brown.edu Brian_Roark@Brown.edu

Abstract

The left-corner transform removes left-recursion from (probabilistic) context-free grammars and unification grammars, permitting simple top-down parsing techniques to be used. Unfortunately the grammars produced by the standard left-corner transform are usually much larger than the original. The selective left-corner transform described in this paper produces a transformed grammar which simulates left-corner recognition of a user-specified set of the original productions, and top-down recognition of the others. Combined with two factorizations, it produces non-left-recursive grammars that are not much larger than the original.

1 Introduction

Top-down parsing techniques are attractive because of their simplicity, and can often achieve good performance in practice (Roark and Johnson, 1999). However, with a left-recursive grammar such parsers typically fail to terminate. The left-corner grammar transform converts a left-recursive grammar into a non-left-recursive one: a top-down parser using a left-corner transformed grammar simulates a left-corner parser using the original grammar (Rosenkrantz and Lewis II, 1970; Aho and Ullman, 1972). However, the left-corner transformed grammar can be significantly larger than the original grammar, causing numerous problems. For example, we show below that a probabilistic context-free grammar (PCFG) estimated from left-corner transformed Penn WSJ tree-bank trees exhibits considerably greater sparse data problems than a PCFG estimated in the usual manner, simply because the left-corner transformed grammar contains approximately 20 times more productions. The transform described in this paper produces a grammar approximately the same size as the input grammar, which is not as adversely affected by sparse data.

* This research was supported by NSF awards 9720368, 9870676 and 9812169. We would like to thank our colleagues in BLLIP (Brown Laboratory for Linguistic Information Processing) and Bob Moore for their helpful comments on this paper.

Left-corner transforms are particularly useful because they can preserve annotations on productions (more on this below) and are therefore applicable to more complex grammar formalisms as well as CFGs; a property which other approaches to left-recursion elimination typically lack. For example, they apply to left-recursive unification-based grammars (Matsumoto et al., 1983; Pereira and Shieber, 1987; Johnson, 1998a). Because the emission probability of a PCFG production can be regarded as an annotation on a CFG production, the left-corner transform can produce a CFG with weighted productions which assigns the same probabilities to strings and transformed trees as the original grammar (Abney et al., 1999). However, the transformed grammars can be much larger than the original, which is unacceptable for many applications involving large grammars.

The selective left-corner transform reduces the transformed grammar size because only those productions which appear in a left-recursive cycle need be recognized left-corner in order to remove left-recursion. A top-down parser using a grammar produced by the selective left-corner transform simulates a *generalized left-corner parser* (Demers, 1977; Nijholt, 1980) which recognizes a user-specified subset of the original productions in a left-corner fashion, and the other productions top-down.

Although we do not investigate it in this paper, the selective left-corner transform should usually have a smaller search space relative to the standard left-corner transform, all else being equal. The partial parses produced during a top-down parse consist of a single connected tree fragment, while the partial parses produced produced during a left-corner parse generally consist of several disconnected tree fragments. Since these fragments are only weakly related (via the "link" constraint described below), the search for each fragment is relatively independent. This may be responsible for the observation that exhaustive left-corner parsing is less efficient than top-down parsing (Covington, 1994). Informally, because the selective left-corner transform recognizes only a subset of productions in a left-corner fashion, its partial parses contain fewer tree discontiguous

fragments and the search may be more efficient.

While this paper focuses on reducing grammar size to minimize sparse data problems in PCFG estimation, the modified left-corner transforms described here are generally applicable wherever the original left-corner transform is. For example, the selective left-corner transform can be used in place of the standard left-corner transform in the construction of finite-state approximations (Johnson, 1998a), often reducing the size of the intermediate automata constructed. The selective left-corner transform can be generalized to head-corner parsing (van Noord, 1997), yielding a selective head-corner parser. (This follows from generalizing the selective left-corner transform to Horn clauses).

After this paper was accepted for publication we learnt of Moore (2000), which addresses the issue of grammar size using very similar techniques to those proposed here. The goals of the two papers are slightly different: Moore's approach is designed to reduce the total grammar size (i.e., the sum of the lengths of the productions), while our approach minimizes the number of productions. Moore (2000) does not address left-corner tree-transforms, or questions of sparse data and parsing accuracy that are covered in section 3.

2 The selective left-corner and related transforms

This section introduces the selective left-corner transform and two additional factorization transforms which apply to its output. These transforms are used in the experiments described in the following section. As Moore (2000) observes, in general the transforms produce a non-left-recursive output grammar only if the input grammar G does not contain unary cycles, i.e., there is no nonterminal A such that $A \rightarrow_G^+ A$.

2.1 The selective left-corner transform

The selective left-corner transform takes as input a CFG $G = (V, T, P, S)$ and a set of *left-corner productions* $L \subseteq P$, which contains no epsilon productions; the non-left-corner productions $P - L$ are called *top-down productions*. The *standard left-corner transform* is obtained by setting L to the set of all non-epsilon productions in P. The *selective left-corner transform* of G with respect to L is the CFG $\mathcal{LC}_L(G) = (V_1, T, P_1, S)$, where:

$$V_1 \;=\; V \cup \{D\text{–}X : D \in V, X \in V \cup T\}$$

and P_1 contains all instances of the schemata 1. In these schemata, $D \in V$, $w \in T$, and lower case greek letters range over $(V \cup T)^*$. The $D\text{–}X$ are new nonterminals; informally they encode a parse state in which an D is predicted top-down and an X has been found left-corner, so $D\text{–}X \Rightarrow^*_{\mathcal{LC}_L(G)} \gamma$ only if $D \Rightarrow^*_G X\gamma$.

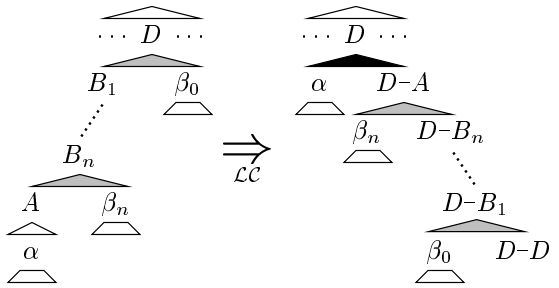

Figure 1: Schematic parse trees generated by the original grammar G and the selective left-corner transformed grammar $\mathcal{LC}_L(G)$. The shaded local trees in the original parse tree correspond to left-corner productions; the corresponding local trees (generated by instances of schema 1c) in the selective left-corner transformed tree are also shown shaded. The local tree colored black is generated by an instance of schema 1b.

$$\begin{array}{lll} D \;\rightarrow\; w\; D\text{–}w & & (1a) \\ D \;\rightarrow\; \alpha\; D\text{–}A & \text{where } A \rightarrow \alpha \in P - L & (1b) \\ D\text{–}B \;\rightarrow\; \beta\; D\text{–}C & \text{where } C \rightarrow B\,\beta \in L & (1c) \\ D\text{–}D \;\rightarrow\; \epsilon & & (1d) \end{array}$$

The schemata function as follows. The productions introduced by schema 1a start a left-corner parse of a predicted nonterminal D with its leftmost terminal w, while those introduced by schema 1b start a left-corner parse of D with a left-corner A, which is itself found by the top-down recognition of production $A \rightarrow \alpha \in P - L$. Schema 1c extends the current left-corner B up to a C with the left-corner recognition of production $C \rightarrow B\,\beta$. Finally, schema 1d matches the top-down prediction with the recognized left-corner category.

Figure 1 schematically depicts the relationship between a chain of left-corner productions in a parse tree generated by G and the chain of corresponding instances of schema 1c. The left-corner recognition of the chain starts with the recognition of α, the right-hand side of a top-down production $A \rightarrow \alpha$, using an instance of schema 1b. The left-branching chain of left-corner productions corresponds to a right-branching chain of instances of schema 1c; the left-corner transform in effect converts left recursion into right recursion. Notice that the top-down predicted category D is passed down this right-recursive chain, effectively multiplying each left-corner productions by the possible top-down predicted categories. The right recursion terminates with an instance of schema 1d when the left-corner and top-down categories match.

Figure 2 shows how top-down productions from G are recognized using $\mathcal{LC}_L(G)$. When the se-

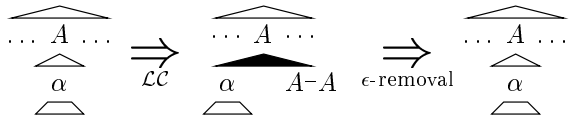

Figure 2: The recognition of a top-down production $A \rightarrow \alpha$ by $\mathcal{LC}_L(G)$ involves a left-corner category A–A, which immediately rewrites to ϵ. One-step ϵ-removal applied to $\mathcal{LC}_L(G)$ produces a grammar in which each top-down production $A \rightarrow \alpha$ corresponds to a production $A \rightarrow \alpha$ in the transformed grammar.

lective left-corner transform is followed by a one-step ϵ-removal transform (i.e., composition or partial evaluation of schema 1b with respect to schema 1d (Johnson, 1998a; Abney and Johnson, 1991; Resnik, 1992)), each top-down production from G appears unchanged in the final grammar. Full ϵ-removal yields the grammar given by the schemata below.

D	$\rightarrow w\ D$–w	
D	$\rightarrow w$	where $D \Rightarrow_L^+ w$
D	$\rightarrow \alpha\ D$–A	where $A \rightarrow \alpha \in P - L$
D	$\rightarrow \alpha$	where $D \Rightarrow_L^\star A, A \rightarrow \alpha \in P - L$
D–B	$\rightarrow \beta\ D$–C	where $C \rightarrow B\ \beta \in L$
D–B	$\rightarrow \beta$	where $D \Rightarrow_L^\star C, C \rightarrow B\ \beta \in L$

Moore (2000) introduces a version of the left-corner transform called LC_{LR}, which applies only to productions with left-recursive parent and left child categories. In the context of the other transforms that Moore introduces, it seems to have the same effect in his system as the selective left-corner transform does here.

2.2 Selective left-corner tree transforms

There is a 1-to-1 correspondence between the parse trees generated by G and $\mathcal{LC}_L(G)$. A tree t is generated by G iff there is a corresponding t' generated by $\mathcal{LC}_L(G)$, where each occurrence of a top-down production in the derivation of t corresponds to exactly one local tree generated by occurrence of the corresponding instance of schema 1b in the derivation of t', and each occurrence of a left-corner production in t corresponds to exactly one occurrence of the corresponding instance of schema 1c in t'. It is straightforward to define a 1-to-1 tree transform $\mathcal{T}_L$ mapping parse trees of G into parse trees of $\mathcal{LC}_L(G)$ (Johnson, 1998a; Roark and Johnson, 1999). In the empirical evaluation below, we estimate a PCFG from the trees obtained by applying $\mathcal{T}_L$ to the trees in the Penn WSJ tree-bank, and compare it to the PCFG estimated from the original tree-bank trees. A stochastic top-down parser using the PCFG estimated from the trees produced by $\mathcal{T}_L$ simulates a stochastic generalized left-corner parser, which is a generalization of a standard stochastic left-corner parser that permits productions to be recognized top-down as well as left-corner (Manning and Carpenter, 1997). Thus investigating the properties of PCFG estimated from trees transformed with $\mathcal{T}_L$ is an easy way of studying stochastic push-down automata performing generalized left-corner parses.

2.3 Pruning useless productions

We turn now to the problem of reducing the size of the grammars produced by left-corner transforms. Many of the productions generated by schemata 1 are useless, i.e., they never appear in any terminating derivation. While they can be removed by standard methods for deleting useless productions (Hopcroft and Ullman, 1979), the relationship between the parse trees of G and $\mathcal{LC}_L(G)$ depicted in Figure 1 shows how to determine ahead of time the new nonterminals D–X that can appear in useful productions of $\mathcal{LC}_L(G)$. This is known as a *link constraint*.

For (P)CFGs there is a particularly simple link constraint: D–X appears in useful productions of $\mathcal{LC}_L(G)$ only if $\exists \gamma \in (V \cup T)^\star . D \Rightarrow_L^\star X\gamma$. If epsilon removal is applied to the resulting grammar, D–X appears in useful productions only if $\exists \gamma \in (V \cup T)^+ . D \Rightarrow_L^\star X\gamma$. Thus one only need generate instances of the left-corner schemata which satisfy the corresponding link constraints.

Moore (2000) suggests an additional constraint on nonterminals D–X that can appear in useful productions of $\mathcal{LC}_L(G)$: D must either be the start symbol of G or else appear in a production $A \rightarrow \alpha D \beta$ of G, for any $A \in V$, $\alpha \in \{V \cup T\}^+$ and $\beta \in \{V \cup T\}^\star$. It is easy to see that the productions that Moore's constraint prohibits are useless. There is one nonterminal in the tree-bank grammar investigated below that has this property, namely LST. However, in the tree-bank grammar none of the productions expanding LST are left-recursive (in fact, the first child is always a preterminal), so Moore's constraint does not affect the size of the transformed grammars investigated below.

While these constraints can dramatically reduce both the number of productions and the size of the parsing search space of the transformed grammar, in general the transformed grammar $\mathcal{LC}_L(G)$ can be quadratically larger than G. There are two causes for the explosion in grammar size. First, $\mathcal{LC}_L(G)$ contains an instance of schema 1b for each top-down production $A \rightarrow \alpha$ and each D such that $\exists \gamma . D \Rightarrow_L^\star A\gamma$. Second, $\mathcal{LC}_L(G)$ contains an instance of schema 1c for each left-corner production $C \rightarrow \beta$ and each D such that $\exists \gamma . D \Rightarrow_L^\star C\gamma$. In effect, $\mathcal{LC}_L(G)$ contains one copy of each production for each possible left-corner ancestor. Section 2.5 describes further factorizations of the productions of $\mathcal{LC}_L(G)$ which mitigate these causes.

2.4 Optimal choice of L

Because $\Rightarrow_L^\star$ increases monotonically with $\Rightarrow_L$ and hence L, we typically reduce the size of $\mathcal{LC}_L(G)$ by making the left-corner production set L as small as possible. This section shows how to find the unique minimal set of left-corner productions L such that $\mathcal{LC}_L(G)$ is not left-recursive.

Assume $G = (V, T, P, S)$ is pruned (i.e., P contains no useless productions) and that there is no $A \in V$ such that $A \rightarrow_G^+ A$ (i.e., G does not generate recursive unary branching chains). For reasons of space we also assume that P contains no ϵ-productions, but this approach can be extended to deal with them if desired. A production $A \rightarrow B\beta \in P$ is *left-recursive* iff $\exists \gamma \in (V \cup T)^\star . B \Rightarrow_P^\star A\gamma$, i.e., P rewrites B into a string beginning with A. Let L_0 be the set of left-recursive productions in G. Then we claim (1) that $\mathcal{LC}_{L_0}(G)$ is not left-recursive, and (2) that for all $L \subset L_0$, $\mathcal{LC}_L(G)$ is left-recursive.

Claim 1 follows from the fact that if $A \Rightarrow_{L_0} B\gamma$ then $A \Rightarrow_P B\gamma$ and the constraints in section 2.3 on useful productions of $\mathcal{LC}_{L_0}(G)$. Claim 2 follows from the fact that if $L \subset L_0$ then there is a chain of left-recursive productions that includes a top-down production; a simple induction on the length of the chain shows that $\mathcal{LC}_L(G)$ is left-recursive.

This result justifies the common practice in natural language left-corner parsing of taking the terminals to be the preterminal part-of-speech tags, rather than the lexical items themselves. (We did not attempt to calculate the size of such a left-corner grammar in the empirical evaluation below, but it would be much larger than any of the grammars described there). In fact, if the preterminals are distinct from the other nonterminals (as they are in the tree-bank grammars investigated below) then L_0 does not include any productions beginning with a preterminal, and $\mathcal{LC}_{L_0}(G)$ contains no instances of schema 1a at all. We now turn our attention to the other schemata of the selective left-corner grammar transform.

2.5 Factoring the output of $\mathcal{LC}_L$

This section defines two factorizations of the output of the selective left-corner grammar transform that can dramatically reduce its size. These factorizations are most effective if the number of productions is much larger than the number of nonterminals, as is usually the case with tree-bank grammars.

The *top-down factorization* decomposes schema 1b by introducing new nonterminals D', where $D \in V$, that have the same expansions that D does in G. Using the same interpretation for variables as in schemata 1, if $G = (V, T, P, S)$ then $\mathcal{LC}_L^{(td)}(G) = (V_{td}, T, P_{td}, S)$, where:

$$V_{td} \;=\; V_1 \cup \{D' : D \in V\}$$

and P_{td} contains all instances of the schemata 1a, 3a, 3b, 1c and 1d.

$$\begin{array}{lll} D \;\rightarrow\; A'\; D{-}A & \text{where } A \rightarrow \alpha \in P - L & \text{(3a)} \\ A' \;\rightarrow\; \alpha & \text{where } A \rightarrow \alpha \in P - L & \text{(3b)} \end{array}$$

Notice that the number of instances of schema 3a is less than the square of the number of nonterminals and that the number of instances of schema 3b is the number of top-down productions; the sum of these numbers is usually much less than the number of instances of schema 1b.

Top-down factoring plays approximately the same role as "non-left-recursion grouping" (NLRG) does in Moore's (2000) approach. The major difference is that NLRG applies to all productions $A \rightarrow B\beta$ in which B is not left-recursive, i.e., $\nexists \gamma . B \Rightarrow_P^+ B\gamma$, while in our system top-down factorization applies to those productions for which $\nexists \gamma . B \Rightarrow_P^\star A\gamma$, i.e., the productions not directly involved in left recursion.

The *left-corner factorization* decomposes schema 1c in a similar way using new nonterminals $D{\backslash}X$, where $D \in V$ and $X \in V \cup T$. $\mathcal{LC}_L^{(lc)}(G) = (V_{lc}, T, P_{lc}, S)$, where:

$$V_{lc} \;=\; V_1 \cup \{D{\backslash}X : D \in V, X \in V \cup T\}$$

and P_{lc} contains all instances of the schemata 1a, 1b, 4a, 4b and 1d.

$$\begin{array}{lll} D{-}B \;\rightarrow\; C{\backslash}B\; D{-}C & \text{where } C \rightarrow B\,\beta \in L & \text{(4a)} \\ C{\backslash}B \;\rightarrow\; \beta & \text{where } C \rightarrow B\,\beta \in L & \text{(4b)} \end{array}$$

The number of instances of schema 4a is bounded by the number of instances of schema 1c and is typically much smaller, while the number of instances of schema 4b is precisely the number of left-corner productions L.

Left-corner factoring seems to correspond to one step of Moore's (2000) "left factor" (LF) operation. The left factor operation constructs new nonterminals corresponding to common prefixes of arbitrary length, while left-corner factoring effectively only factors the first nonterminal symbol on the right hand side of left-corner productions. While we have not done experiments, Moore's left factor operation would seem to reduce the total number of symbols in the transformed grammar at the expense of possibly introducing additional productions, while our left-corner factoring reduces the number of productions.

These two factorizations can be used together in the obvious way to define a grammar transform $\mathcal{LC}_L^{(td,lc)}$, whose productions are defined by schemata 1a, 3a, 3b, 4a, 4b and 1d. There are corresponding tree transforms, which we refer to as $\mathcal{T}_L^{(td)}$, etc., below. Of course, the pruning constraints described in section 2.3 are applicable with these factorizations, and corresponding invertible tree transforms can be constructed.

3 Empirical Results

To examine the effect of the transforms outlined above, we experimented with various PCFGs induced from sections 2–21 of a modified Penn WSJ tree-bank as described in Johnson (1998b) (i.e., labels simplified to grammatical categories, ROOT nodes added, empty nodes and vacuous unary branches deleted, and auxiliaries retagged as AUX or AUXG). We ignored lexical items, and treated the part-of-speech tags as terminals. As Bob Moore pointed out to us, the left-corner transform may produce left-recursive grammars if its input grammar contains unary cycles, so we removed them using the a transform that Moore suggested. Given an initial set of (non-epsilon) productions P, the transformed grammar contains the following productions, where the $A^\natural$ are new non-terminals:

$$\begin{array}{ll} A \to \alpha & \text{where } A \to \alpha \in P, A \not\Rightarrow_P^+ A \\ A \to D^\natural & \text{where } A \Rightarrow_P^\star D \Rightarrow_P^+ A \\ A^\natural \to \alpha & \text{where } A \to \alpha \in P, A \Rightarrow_P^+ A, \alpha \not\Rightarrow_P^* A \end{array}$$

This transform can be extended to one on PCFGs which preserves derivation probabilities. In this section, we fix P to be the productions that result after applying this unary cycle removal transformation to the tree-bank productions, and G to be the corresponding grammar.

Tables 1 and 2 give the sizes of selective left-corner grammar transforms of G for various values of the left-corner set L and factorizations, without and with epsilon-removal respectively. In the tables, L_0 is the set of left-recursive productions in P, as defined in section 2.4. N is the set of productions in P whose left-hand sides do not begin with a part-of-speech (POS) tag; because POS tags are distinct from other nonterminals in the tree-bank, N is an easily identified set of productions guaranteed to include L_0. The tables also gives the sizes of maximum-likelihood PCFGs estimated from the trees resulting from applying the selective left-corner tree transforms $\mathcal{T}$ to the tree-bank, breaking unary cycles as described above. For the parsing experiments below we always deleted empty nodes in the output of these tree transforms; this corresponds to epsilon removal in the grammar transform.

First, note that $\mathcal{LC}_P(G)$, the result of applying the standard left-corner grammar transform to G, has approximately 20 times the number of productions that G has. However $\mathcal{LC}_{L_0}^{(td,lc)}(G)$, the result of applying the selective left-corner grammar transformation with factorization, has approximately 1.4 times the number of productions that G has. Thus the methods described in this paper can in fact dramatically reduce the size of left-corner transformed grammars. Second, note that $\mathcal{LC}_N^{(td,lc)}(G)$ is not much larger than $\mathcal{LC}_{L_0}^{(td,lc)}(G)$. This is because N is not much larger than L_0, which in turn is because most pairs of non-POS nonterminals A, B are mutually left-recursive.

	none	(td)	(lc)	(td,lc)
G	15,040			
$\mathcal{LC}_P$	346,344		30,716	
$\mathcal{LC}_N$	345,272	113,616	254,067	22,411
$\mathcal{LC}_{L_0}$	314,555	103,504	232,415	21,364
$\mathcal{T}_P$	20,087		17,146	
$\mathcal{T}_N$	19,619	16,349	19,002	15,732
$\mathcal{T}_{L_0}$	18,945	16,126	18,437	15,618

Table 1: Sizes of PCFGs inferred using various grammar and tree transforms after pruning with link constraints without epsilon removal. Columns indicate factorization. In the grammar and tree transforms, P is the set of productions in G (i.e., the standard left-corner transform), N is the set of all productions in P which do not begin with a POS tag, and L_0 is the set of left-recursive productions.

	none	(td)	(lc)	(td,lc)
$\mathcal{LC}_P$	564,430		38,489	
$\mathcal{LC}_N$	563,295	176,644	411,986	25,335
$\mathcal{LC}_{L_0}$	505,435	157,899	371,102	23,566
$\mathcal{T}_P$	22,035		17,398	
$\mathcal{T}_N$	21,589	16,688	20,696	15,795
$\mathcal{T}_{L_0}$	21,061	16,566	20,168	15,673

Table 2: Sizes of PCFGs inferred using various grammar and tree transforms after pruning with link constraints with epsilon removal, using the same notation as Table 1.

Turning now to the PCFGs estimated after applying tree transforms, we notice that grammar size does not increase nearly so dramatically. These PCFGs encode a maximum-likelihood estimate of the state transition probabilities for various stochastic generalized left-corner parsers, since a top-down parser using these grammars simulates a generalized left-corner parser. The fact that $\mathcal{LC}_P(G)$ is 17 times larger than the PCFG inferred after applying $\mathcal{T}_P$ to the tree-bank means that most of the possible transitions of a standard stochastic left-corner parser are not observed in the tree-bank training data. The state of a left-corner parser does capture some linguistic generalizations (Manning and Carpenter, 1997; Roark and Johnson, 1999), but one might still expect sparse-data problems. Note that $\mathcal{LC}_{L_0}^{(td,lc)}$ is only 1.4 times larger than $\mathcal{T}_{L_0}^{(td,lc)}$, so we expect less serious sparse data problems with the factored selective left-corner transform.

We quantify these sparse data problems in two ways using a held-out test corpus, viz., all sentences in section 23 of the tree-bank. First, table 3 lists the number of sentences in the test corpus that fail to receive a parse with the various PCFGs mentioned

Transform	none	(td)	(lc)	(td, lc)
none	0			
$\mathcal{T}_P$	2		0	
$\mathcal{T}_N$	2	0	2	0
$\mathcal{T}_{L_0}$	0	0	0	0

Table 3: The number of sentences in section 23 that do not receive a parse using various grammars estimated from sections 2–21.

Transform	none	(td)	(lc)	(td, lc)
none	514			
$\mathcal{T}_P$	665		535	
$\mathcal{T}_N$	664	543	639	518
$\mathcal{T}_{L_0}$	640	547	615	522
$\mathcal{T}_{P\,\epsilon}$	719		539	
$\mathcal{T}_{N\,\epsilon}$	718	554	685	521
$\mathcal{T}_{L_0\,\epsilon}$	706	561	666	521

Table 4: The number of productions found in the transformed trees of sentences in section 23 that do not appear in the corresponding transformed trees from sections 2–21. (The subscript epsilon indicates epsilon removal was applied).

above. This is a relatively crude measure, but correlates roughly with the ratios of grammar sizes, as expected.

Second, table 4 lists the number of productions found in the tree-transformed test corpus that do not appear in the correspondingly transformed trees of sections 2–21. What is striking here is that the number of missing productions after either of the transforms $\mathcal{T}_{L_0}^{(td,lc)}$ or $\mathcal{T}_N^{(td,lc)}$ is approximately the same as the number of missing productions using the untransformed trees, indicating that the factored selective left-corner transforms cause little or no additional sparse data problem. (The relationship between local trees in the parse trees of G and $\mathcal{LC}_L(G)$ mentioned earlier implies that left-corner tree transformations will not decrease the number of missing productions).

We also investigate the accuracy of the maximum-likelihood parses (MLPs) obtained using the PCFGs estimated from the output of the various left-corner tree transforms.[1] We searched for these parses using an exhaustive CKY parser. Because the parse trees of these PCFGs are isomorphic to the derivations of the corresponding stochastic generalized left-corner parsers, we are in fact evaluating different kinds of stochastic generalized left-corner parsers inferred from sections 2–21 of the tree-bank. We used

[1] We did not investigate the grammars produced by the various left-corner grammar transforms. Because a left-corner grammar transform $\mathcal{LC}_L$ preserves production probabilities, the highest scoring parses obtained using the weighted CFG $\mathcal{LC}_L(G)$ should be the highest scoring parses obtained using G transformed by $\mathcal{T}_L$.

	none	(td)	(lc)	(td, lc)
none	70.8,75.3			
$\mathcal{T}_P$	75.8,77.7		74.8,76.9	
$\mathcal{T}_N$	75.8,77.6	73.8,75.8	75.5,77.8	72.8,75.4
$\mathcal{T}_{L_0}$	75.8,77.4	73.0,74.7	75.6,77.8	72.9,75.4

Table 5: Labelled recall and precision scores of PCFGs estimated using various tree-transforms in a transform-detransform framework using test data from section 23.

the transform-detransform framework described in Johnson (1998b) to evaluate the parses, i.e., we applied the appropriate inverse tree transform $\mathcal{T}^{-1}$ to detransform the parse trees produced using the PCFG estimated from trees transformed by $\mathcal{T}$. By calculating the labelled precision and recall scores for the detransformed trees in the usual manner, we can systematically compare the parsing accuracy of different kinds of stochastic generalized left-corner parsers.

Table 5 presents the results of this comparison. As reported previously, the standard left-corner grammar embeds sufficient non-local information in its productions to significantly improve the labelled precision and recall of its MLPs with respect to MLPs of the PCFG estimated from the untransformed trees (Manning and Carpenter, 1997; Roark and Johnson, 1999). Parsing accuracy drops off as grammar size decreases, presumably because smaller PCFGs have fewer adjustable parameters with which to describe this non-local information. There are other kinds of non-local information which can be incorporated into a PCFG using a transform-detransform approach that result in an even greater improvement of parsing accuracy (Johnson, 1998b). Ultimately, however, it seems that a more complex approach incorporating back-off and smoothing is necessary in order to achieve the parsing accuracy achieved by Charniak (1997) and Collins (1997).

4 Conclusion

This paper presented factored selective left-corner grammar transforms. These transforms preserve the primary benefits of the left-corner grammar transform (i.e., elimination of left-recursion and preservation of annotations on productions) while dramatically ameliorating its principal problems (grammar size and sparse data problems). This should extend the applicability of left-corner techniques to situations involving large grammars. We showed how to identify the minimal set L_0 of productions of a grammar that must be recognized left-corner in order for the transformed grammar not to be left-recursive. We also proposed two factorizations of the output of the selective left-corner grammar transform which further reduce grammar size, and showed that there is only a minor increase in grammar size when the

factored selective left-corner transform is applied to a large tree-bank grammar. Finally, we exploited the tree transforms that correspond to these grammar transforms to formulate and study a class of stochastic generalized left-corner parsers.

This work could be extended in a number of ways. For example, in this paper we assumed that one would always choose a left-corner production set that includes the minimal set L_0 required to ensure that the transformed grammar is not left-recursive. However, Roark and Johnson (1999) report good performance from a stochastically-guided top-down parser, suggesting that left-recursion is not always fatal. It might be possible to judiciously choose a left-corner production set *smaller* than L_0 which eliminates pernicious left-recursion, so that the remaining left-recursive cycles have such low probability that they will effectively never be used and a stochastically-guided top-down parser will never search them.

References

Stephen Abney and Mark Johnson. 1991. Memory requirements and local ambiguities of parsing strategies. *Journal of Psycholinguistic Research*, 20(3):233–250.

Steven Abney, David McAllester, and Fernando Pereira. 1999. Relating probabilistic grammars and automata. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 542–549, San Francisco. Morgan Kaufmann.

Alfred V. Aho and Jeffery D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling; Volume 1: Parsing*. Prentice-Hall, Englewood Cliffs, New Jersey.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Menlo Park. AAAI Press/MIT Press.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *The Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, San Francisco. Morgan Kaufmann.

Michael A. Covington. 1994. *Natural Language Processing for Prolog Programmers*. Prentice Hall, Englewood Cliffs, New Jersey.

A. Demers. 1977. Generalized left-corner parsing. In *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, 1977 ACM SIGACT/SIGPLAN*, pages 170–182.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

Mark Johnson. 1998a. Finite state approximation of unification grammars using left-corner grammar transforms. In *The Proceedings of the 36th Annual Conference of the Association for Computational Linguistics (COLING-ACL)*, pages 619–623. Morgan Kaufmann.

Mark Johnson. 1998b. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Christopher D. Manning and Bob Carpenter. 1997. Probabilistic parsing using left-corner models. In *Proceedings of the 5th International Workshop on Parsing Technologies*, pages 147–158, Massachusetts Institute of Technology.

Yuji Matsumoto, Hozumi Tanaka, Hideki Hirakawa, Hideo Miyoshi, and Hideki Yasukawa. 1983. BUP: A bottom-up parser embedded in Prolog. *New Generation Computing*, 1(2):145–158.

Robert C. Moore. 2000. Removing left recursion from context-free grammars. In *Proceedings of 1st Annual Conference of the North American Chapter of the Association for Computational Linguistics*, San Francisco. Morgan Kaufmann.

Anton Nijholt. 1980. *Context-free Grammars: Covers, Normal Forms, and Parsing*. Springer Verlag, Berlin.

Fernando C.N. Pereira and Stuart M. Shieber. 1987. *Prolog and Natural Language Analysis*. Number 10 in CSLI Lecture Notes Series. Chicago University Press, Chicago.

Philip Resnik. 1992. Left-corner parsing and psychological plausibility. In *The Proceedings of the fifteenth International Conference on Computational Linguistics, COLING-92*, volume 1, pages 191–197.

Brian Roark and Mark Johnson. 1999. Efficient probabilistic top-down and left-corner parsing. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 421–428.

Stanley J. Rosenkrantz and Philip M. Lewis II. 1970. Deterministic left corner parser. In *IEEE Conference Record of the 11th Annual Symposium on Switching and Automata*, pages 139–152.

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