STOCHASTIC TREE-ADJOINING GRAMMARS*

Yves Schabes

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389

ABSTRACT

The notion of stochastic lexicalized tree-adjoining grammar (SLTAG) is defined and basic algorithms for SLTAG are designed. The parameters of a SLTAG correspond to the probability of combining two structures each one associated with a word. The characteristics of SLTAG are unique and novel since it is lexically sensitive (as N-gram models or Hidden Markov Models) and yet hierarchical (as stochastic context-free grammars). An algorithm for computing the probability of a sentence generated by a SLTAG is presented. Then, an iterative algorithm for estimating the parameters of a SLTAG given a training corpus is introduced.

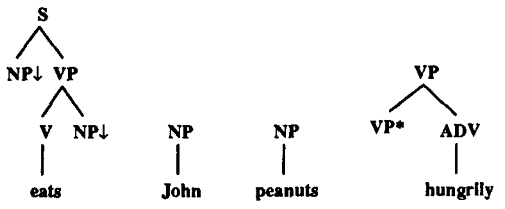
1. MOTIVATIONS

Although stochastic techniques applied to syntax modeling have recently regained popularity, current language models suffer from obvious inherent inadequacies. Early proposals such as Markov Models, N-gram models [1, 2, 3] and Hidden Markov Models were very quickly shown to be linguistically not appropriate for natural language (e.g. [4]) since they are unable to capture long distance dependencies or to describe hierarchically the syntax of natural languages. Stochastic context-free grammar [5] is a hierarchical model more appropriate for natural languages, however none of such proposals [6, 7] perform as well as the simpler Markov Models because of the difficulty of capturing lexical information. The parameters of a stochastic context-free grammar do not correspond directly to a distribution over words since distributional phenomena over words that are embodied by the application of more than one context-free rule cannot be captured under the context-freeness assumption. This leads to the difficulty of maintaining a standard hierarchical model while capturing lexical dependencies.

This fact prompted researchers in natural language processing to give up hierarchical language models in the favor of non-hierarchical statistical models over words (such as word N-grams models). Probably for lack of a better language model, it has also been argued that the phenomena that such devices cannot capture occur relatively infrequently. Such argumentation is linguistically not sound.

Lexicalized tree-adjoining grammars (LTAG)[1] combine hierarchical structures while being lexically sensitive and are therefore more appropriate for statistical analysis of language. In fact, LTAGs are the simplest hierarchical formalism which can serve as the basis for lexicalizing context-free grammar [10, 11].

LTAG is a tree-rewriting system that combines trees of large domain with *adjoining* and *substitution*. The trees found in a TAG take advantage of the available extended domain of locality by localizing syntactic dependencies (such as filler-gap, subject-verb, verb-object) and most semantic dependencies (such as predicate-argument relationship). For example, the following trees can be found in a LTAG lexicon:



Since the elementary trees of a LTAG are minimal syntactic and semantic units, distributional analysis of the combination of these elementary trees based on a training corpus will inform us about relevant statistical aspects of the language such as the classes of words appearing as arguments of a predicative element, the distribution of the adverbs licensed by a specific verb, or the adjectives licensed by a specific noun.

This kind of statistical analysis as independently suggested in [12] can be made with LTAGs because of their extended domain of locality but also because of their lexicalized property.

In this paper, this intuition is made formally precise by defining the notion of a stochastic lexicalized tree-adjoining grammar (SLTAG). We present an algorithm for computing the probability of a sentence generated by a SLTAG, and finally we introduce an iterative algorithm for estimating the parameters of a SLTAG given a training corpus of text. This algorithm can either be used for refining the parame-

*This work was partially supported by DARPA Grant N0014-90-31863, ARO Grant DAAL03-89-C-0031 and NSF Grant IRI90-16592. We thank Aravind Joshi for suggesting the use of TAGs for statistical analysis during a private discussion that followed a presentation by Fred Jelinek during the June 1990 meeting of the DARPA Speech and Natural Language Workshop. We are also grateful to Peter Braun, Fred Jelinek, Mark Liberman, Mitch Marcus, Robert Mercer, Fernando Pereira and Stuart Shieber for providing valuable comments.

[1]We assume familiarity throughout the paper with TAGs and its lexicalized variant. See, for instance, [8], [9], [10] or [11].

ters of a SLTAG or for inferring a tree-adjoining grammar from a training corpus.

Due to the lack of space, in this paper the algorithms are described succinctly without proofs of correctness and more attention is given to the concepts and techniques used for SLTAG.

2. SLTAG

Informally speaking, SLTAGs are defined by assigning a probability to the event that an elementary tree is combined (by adjunction or substitution) on a specific node of another elementary tree. These events of combination are the stochastic processes considered.

For sake of mathematical precision and elegance, we use a stochastic linear rewriting system, stochastic linear indexed grammars (SLIG), as a notation for SLTAGs. A linear indexed grammar is constructed following the method given in [13]. However, in addition, each rule is associated with a probability.

Linear Indexed grammar (LIG) [14, 15] is a rewriting system in which the non-terminal symbols are augmented with a stack. In addition to rewriting non-terminals, the rules of the grammar can have the effect of pushing or popping symbols on top of the stacks that are associated with each non-terminal symbol. A specific rule is triggered by the non-terminal on the left hand side of the rule and the top element of its associated stack. LIGs [15] restrict

The productions of a LIG are restricted to copy the stack corresponding to the non-terminal being rewritten to at most one stack associated with a non-terminal symbol on the right hand side of the production.[2]

In the following, $[\cdot\cdot\rho]$ refers to a possibly unbounded stack whose top element is ρ and whose remaining part is schematically written as '$\cdot\cdot$'. $[\$]$ represents a stack whose only element is the bottom of the stack. While it is possible to define SLIGs in general, we define them for the particular case where the rules are binary branching and where the left hand sides are always incomparable.

A *stochastic linear indexed grammar*, G, is denoted by $(V_N, V_T, V_I, S, Prod)$, where V_N is a finite set of non-terminal symbols; V_T is a finite set of terminal symbols; V_I is a finite set of stack symbols; $S \in V_N$ is the start symbol; $Prod$ is a finite set of productions of the form:

$$\begin{array}{lll} X_0[\$\rho_0] & \rightarrow & a \\ X_0[\cdot\cdot\rho_0] & \rightarrow & X_1[\cdot\cdot\rho_1] \;\; X_2[\$\rho_2] \\ X_0[\cdot\cdot\rho_0] & \rightarrow & X_1[\$\rho_1] \;\; X_2[\cdot\cdot\rho_2] \\ X_0[\$\rho_0] & \rightarrow & X_1[\$\rho_1] \;\; X_2[\$\rho_2] \end{array}$$

where $X_k \in V_N$, $a \in V_T$ and $\rho_0 \in V_I$, $\rho_1, \rho_2 \in V_I^\star$; P, a probability distribution which assigns a probability, $0 \leq P(X[\cdot\cdot x] \rightarrow \Delta) \leq 1$, to a rule, $X[\cdot\cdot x] \rightarrow \Delta \in Prod$ such that the sum of the probabilities of all the rules that can be applied to any non-terminal annotated with a stack is equal to one. More precisely if, $\forall X \in V_N, \forall \rho \in V_I$:

$$\sum_{\Delta} P(X[\cdot\cdot\rho] \rightarrow \Delta) = 1$$

$P(X[\cdot\cdot\rho] \rightarrow \Delta)$ should be interpreted as the probability that $X[\cdot\cdot\rho]$ is rewritten as Δ.

A derivation starts from S associated with the empty stack ($S[\$]$) and each level of the derivation must be validated by a production rule. The *language* of a SLIG is defined as follows: $L = \{w \in V_T^\star \mid S[\$] \overset{\star}{\Rightarrow} w\}$.

The *probability of a derivation* is defined as the product of the probabilities of all individual rules involved (counting repetition) in the derivation, the derivation being validated by a correct configuration of the stack at each level. The *probability of a sentence* is then computed as the sum of the probabilities of all derivations of the sentence.

Following the construction described in [13], given a LTAG, G_{tag}, we construct an equivalent[3] LIG, G_{slig}. In addition, a probability is assigned to each production of the LIG. For simplicity of explanation and without loss of generality we assume that each node in an elementary tree found in a tree-adjoining grammar is either a leaf node (i.e. either a foot node or a non-empty terminal node) or binary branching.[4] The construction of the equivalent SLIG follows.

The non-terminal symbols of G_{slig} are the two symbols 'top' (t) and 'bottom' (b), the set of terminal symbols is the same as the one of G_{tag}, the set of stack symbols is the set of nodes (not node labels) found in the elementary trees of G_{tag} augmented with the bottom of the stack ($\$$), and the start symbol is 'top' (t).

For all root nodes η_0 of an initial tree whose root is labeled by S, the following starting rules are added:

$$t[\$] \xrightarrow{P} t[\$\eta_0] \qquad (1)$$

These rules state that a derivation must start from the top of the root node of some initial tree. P is the probability that a derivation starts from the initial tree associated with a lexical item and rooted by η_0.

Then, for all node η in an elementary tree, the following rules are generated.

- If $\eta_1\eta_2$ are the 2 children of a node η such that η_2 is on

[2] LIGs have been shown to be weakly equivalent to Tree-Adjoining Grammars [16].

[3] The constructed LIG generates the same language as the given tree-adjoining grammar.

[4] The algorithms explained in this paper can be generalized to lexicalized tree-adjoining grammars that need not be in Chomsky Normal Form using techniques similar the one found in [17].

the spine (i.e. subsumes the foot node), include:

$$b[\cdot\cdot\eta] \overset{p=1}{\rightarrow} t[\$\eta_1]t[\cdot\cdot\eta_2] \qquad (2)$$

Since (2) encodes an immediate domination link defined by the tree-adjoining grammar, its associated probability is one.

- Similarly, if $\eta_1\eta_2$ are the 2 children of a node η such that η_1 is on the spine (i.e. subsumes the foot node), include:

  $$b[\cdot\cdot\eta] \overset{p=1}{\rightarrow} t[\cdot\cdot\eta_1]t[\$\eta_2] \qquad (3)$$

  Since (3) encodes an immediate domination link defined by the tree-adjoining grammar, its associated probability is one.

- If $\eta_1\eta_2$ are the 2 children of a node η such that none of them is on the spine, include:

  $$b[\$\eta] \overset{p=1}{\rightarrow} t[\$\eta_1]t[\$\eta_2] \qquad (4)$$

  Since (4) also encodes an immediate domination link defined by the tree-adjoining grammar, its associated probability is one.

- If η is a node labeled by a non-terminal symbol and if it does not have an obligatory adjoining constraint, then we need to consider the case that adjunction might not take place. In this case, include:

  $$t[\cdot\cdot\eta] \overset{p}{\rightarrow} b[\cdot\cdot\eta] \qquad (5)$$

  The probability of rule (5) corresponds to the probability that no adjunction takes place at node η.

- If η is an node on which the auxiliary tree β can be adjoined, the adjunction of β can be predicted, therefore (assuming that η_r is the root node of β) include:

  $$t[\cdot\cdot\eta] \overset{p}{\rightarrow} t[\cdot\cdot\eta\eta_r] \qquad (6)$$

  The probability of rule (6) corresponds to the probability of adjoining the auxiliary tree whose root node is η_r, say β, on the node η belonging to some elementary tree, say α.[5]

- If η_f is the foot node of an auxiliary tree β that has been adjoined, then the derivation of the node below η_f must resume. In this case, include:

  $$b[\cdot\cdot\eta_f] \overset{p=1}{\rightarrow} b[\cdot\cdot] \qquad (7)$$

  The above stochastic production is included with probability one since the decision of adjunction has already been made in rules of the form (6).

- Finally, if η_1 is the root node of an initial tree that can be substituted on a node marked for substitution η, include:

  $$t[\$\eta] \overset{p}{\rightarrow} t[\$\eta_1] \qquad (8)$$

  Here, p is the probability that the initial tree rooted by η_1 is substituted at node η. It corresponds to the probability of substituting the lexicalized initial tree whose root node is η_1, say δ, at the node η of a lexicalized elementary tree, say α.[6]

The SLIG constructed as above is well defined if the following equalities hold for all nodes η:

$$P(t[\cdot\cdot\eta] \rightarrow b[\cdot\cdot\eta]) + \sum_{\eta_1} P(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta_1]) = 1 \qquad (9)$$

$$\sum_{\eta_1} P(t[\$\eta] \rightarrow t[\$\eta_1]) = 1 \qquad (10)$$

$$\sum_{\eta_0} P(t[\$] \rightarrow t[\$\eta_0]) = 1 \qquad (11)$$

A grammar satisfying (12) is called *consistent*.[7]

$$\sum_{w\in\Sigma^*} P(t[\$]\overset{*}{\Rightarrow}w) = 1 \qquad (12)$$

Beside the distributional phenomena that we mentioned earlier, SLTAG also captures the effect of adjoining constraints (selective, obligatory or null adjoining) which are required for tree-adjoining grammar.[8]

3. PROBABILITY OF A SENTENCE

We now define an bottom-up algorithm for SLTAG which computes the probability of an input string. The algorithm is an extension of the CKY-type parser for tree-adjoining grammar [18]. The extended algorithm parses all spans of the input string and also computes their probability in a bottom-up fashion.

Since the string on the frontier of an auxiliary is broken up into two substrings by the foot node, for the purpose of computing the probability of the sentence, we will consider the probability that a node derives two substrings of the input string. This entity will be called the *inside probability*. Its exact definition is given below.

We will refer to the subsequence of the input string $w = a_1 \cdots a_N$ from position i to j, w_i^j. It is defined as follows:

$$w_i^j \overset{def}{=} \begin{cases} a_{i+1}\cdots a_j & \text{, if } i < j \\ \varepsilon & \text{, if } i \geq j \end{cases}$$

Given a string $w = a_1 \cdots a_N$ and a SLTAG rewritten as in (1-8) the *inside probability*, $I^w(pos, \eta, i, j, k, l)$, is defined for all nodes η contained in an elementary tree α and for $pos \in \{t, b\}$, and for all indices $0 \leq i \leq j \leq k \leq l \leq N$ as follows:

[5] Since the grammar is lexicalized, both trees α and β are associated with lexical items, and the site node for adjunction η corresponds to some syntactic modification. Such rule encapsulates S modifiers (e.g. sentential adverbs as in "*apparently* John left"), VP modifiers (e.g. verb phrase adverbs as in "John left *abruptly*)", NP modifiers (e.g. relative clauses as in "The man *who left* was happy"), N modifiers (e.g. adjectives as in "*pretty* woman"), or even sentential complements (e.g. *John thinks that* Harry is sick).

[6] Among other cases, the probability of this rule corresponds to the probability of filling some argument position by a lexicalized tree. It will encapsulate the distribution for selectional restriction since the position of substitution is taken into account.

[7] We will not investigate the conditions under which (12) holds. We conjecture that some of the techniques used for checking the consistency of stochastic context-free grammars can be adapted to SLTAG.

[8] For example, for a given node η setting to zero the probability of all rules of the form (6) has the effect of blocking adjunction.

(i) If the node η does not subsume the foot node of α (if there is one), then j and k are unbound and:
$I^w(pos, \eta, i, -, -, l) \overset{def}{=} P(pos[\$\eta] \overset{*}{\Rightarrow} w_i^l)$

(ii) If the node η subsumes the foot node η_f of α, then:
$I^w(pos, \eta, i, j, k, l) \overset{def}{=} P(\ pos[\$\eta] \overset{*}{\Rightarrow} w_i^j b[\$\eta_f] w_k^l)$

In (ii), only the top element of the stack matters since as a consequence of the construction of the SLIG, we have that if $pos[\$\eta] \overset{*}{\Rightarrow} w_i^j b[\$\eta_f] w_k^l$ then for all string $\gamma \in V_I^*$ we also have $pos[\$\gamma\eta] \overset{*}{\Rightarrow} w_i^j b[\$\gamma\eta_f] w_k^l$.[9]

Initially, all inside probabilities are set to zero. Then, the computation goes bottom-up starting from the productions introducing lexical items: if η is a node such that $b[\$\eta] \rightarrow a$, then:

$$I^w(b, \eta, i, -, -, l) = \begin{cases} 1 & \text{if } l = i+1 \wedge a = w_i^{i+1} \\ 0 & otherwise. \end{cases} \quad (13)$$

Then, the inside probabilities of larger substrings are computed bottom-up relying on the recurrence equations stated in Appendix A. This computation takes in the worst case $O(|G|^2 N^6)$-time and $O(|G|N^4)$-space for a sentence of length N.

Once the inside probabilities computed, we obtain the *probability of the sentence* as follows:

$$P(w) \overset{def}{=} P(t[\$] \overset{*}{\Rightarrow} w) = I^w(t, \$, 0, -, -, |w|) \quad (14)$$

We now consider the problem of re-estimating a SLTAG.

4. RE-ESTIMATION OF SLTAG

Given a set of positive example sentences, $W = \{w_1 \cdots w_K\}$, assumed to have been generated by an unknown SLTAG, we would like to compute the probability of each rule of a given SLTAG in order to maximize the probability that the corpus were generated by this SLTAG. An algorithm solving this problem can be used in two different ways.

The first use is as a re-estimation algorithm. In this approach, the input SLTAG derives structures that are reasonable according to some criteria (such as a linguistic theory and some a priori knowledge of the corpus) and the intended use of the algorithm is to refine the probability of each rule.

The second use is as a learning algorithm. At the first iteration, a SLTAG which generates all possible structures over a given set of nodes and terminal symbols is used. Initially the probability of each rule is randomly assigned and then the algorithm will re-estimate these probabilities.

Informally speaking, given a first estimate of the parameters of a SLTAG, the algorithm re-estimates these parameters on the basis of the parses of each sentence in a training corpus obtained by a CKY-type parser. The algorithm derives a new estimate such that the probability that the corpus were generated by the grammar is increased. By analogy to the inside-outside algorithm for stochastic context-free grammars [19, 7], we believe that the following quantity decreases after each iteration:[10]

$$H_e(W) = -\frac{\sum_{w \in W} log_2(P(w))}{\sum_{w \in W} |w|} \quad (15)$$

In order to derive a new estimate, the algorithm needs to compute for all sentences in W the inside probabilities and the ***outside probabilities.*** Given a string $w = a_1 \cdots a_N$, the ***outside probability***, $O^w(pos, \eta, i, j, k, l)$, is defined for all nodes η contained in an elementary tree α and for $pos \in \{t, b\}$, and for all indices $0 \leq i \leq j \leq k \leq l \leq N$ as follows:

(i) If the node η does not subsume the foot node of α (if there is one), then j and k are unbound and:
$O^w(pos, \eta, i, -, -, l) \overset{def}{=}$
$P(\exists \gamma \in V_I^* \ s.t. \ t[\$] \overset{*}{\Rightarrow} w_0^i \ pos[\$\gamma\eta] \ w_l^N)$

(ii) If the node η does subsume the foot node η_f of α then:
$O^w(pos, \eta, i, j, k, l) \overset{def}{=}$
$P(\exists \gamma \in V_I^* \ s.t.$
$t[\$] \overset{*}{\Rightarrow} w_0^i \ pos[\$\gamma\eta] \ w_l^N$ and $b[\$\gamma\eta_f] \overset{*}{\Rightarrow} w_j^k)$

Once the inside probabilities computed, the outside probabilities can be computed top-down by considering smaller spans of the input string starting with $O^w(t, \$, 0, -, -, N) = 1$ (by definition). This is done by computing the recurrence equations stated in Appendix B.

Due to the lack of space, we only illustrate the re-estimation of the rules corresponding to adjunction, rules of the form: $t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta\prime]$. The other re-estimation formulae can be derived in a similar manner.

In the following, we assume that η subsumes the foot node η_f within a same elementary tree, and also that $\eta\prime$ subsumes the foot node $\eta_{f\prime}$ (within a same elementary tree).

[9]This can be seen by observing that for any node on the path from the root node to the foot node of an auxiliary tree, the stack remains unchanged.

[10]H_e is an estimate of the entropy H of the unknown language being estimated and it converges to the entropy of the language as the size of the corpus grows.

Let:

$$N_w(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta'], i, r, j, k, s, l) = \frac{P(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta'])}{P(w)} \times \left(\begin{array}{l} I^w(t, \eta', i, r, s, l) \\ \times \ I^w(b, \eta, r, j, k, s) \\ \times \ O^w(t, \eta, i, j, k, l) \end{array} \right)$$

and:

$$D_w(t, \eta, i, j, k, l) = \frac{I^w(t, \eta, i, j, k, l) \times O^w(t, \eta, i, j, k, l)}{P(w)}$$

It can be shown that the rule $t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta']$ is optimally reestimated at each iteration as follows:

$$\hat{P}(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta']) = \frac{\sum_{w \in W} \sum_{0 \leq i \leq r \leq j \leq k \leq s \leq l \leq |w|} N_w(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta'], i, r, j, k, s, l)}{\sum_{w \in W} \sum_{0 \leq i \leq j \leq k \leq l \leq |w|} D_w(t, \eta, i, j, k, l)}$$

The denominator of the above reestimation formula estimates the probability that a derivation will involve at least one expansion of $t[\cdot\cdot\eta]$. The numerator estimates the probability that a derivation will involve the rule $t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta']$.

The probability of no adjunction on the node η, $P(t[\cdot\cdot\eta] \rightarrow b[\cdot\cdot\eta]$ is reestimated using the equality (9).

The algorithm reiterates until $H_e(W)$ is unchanged (within some epsilon) between two iterations. Each iteration of the algorithm requires at most $O(|G|^2 N^6)$-time for each sentence of length N.

5. CONCLUSION

A novel statistical language model and fundamental algorithms for this model have been presented.

SLTAGs provide a stochastic model both hierarchical and sensitive to lexical information. They combine the advantages of purely lexical models such as N-gram distributions or Hidden Markov Models and the one of hierarchical modes as stochastic context-free grammars without their inherent limitations. The parameters of a SLTAG correspond to the probability of combining two structures each one associated with a word and therefore capture linguistically relevant distributions over words.

An algorithm for computing the probability of a sentence generated by a SLTAG was presented as well as an iterative algorithm for estimating the parameters of a SLTAG given a training corpus of raw text. Similarly to its context-free counterpart, the reestimation algorithm can be extended to handle partially parsed corpora [20]. The worst case complexity of the algorithm with respect to the length of the input string ($O(N^6)$) makes it impractical with a large corpus on a single processor computer for grammars requiring the worst case complexity. However, this complexity reduces to $O(N^3)$ or to $O(N^2)$ for interesting subsets of SLTAGs. If time permits, experiments in this direction will be reported at the time of the meeting.

Furthermore, the techniques explained in this paper apply to other grammatical formalisms such as combinatory categorial grammars and modified head grammars since they have been proven to be equivalent to tree-adjoining grammars and linear indexed grammars [21].

In collaboration with Aravind Joshi, Fernando Pereira and Stuart Shieber, we are currently investigating additional algorithms and applications for SLTAG, methods for lexical clustering and automatic construction of a SLTAG from a large training corpus.

REFERENCES

1. Pratt, F. *Secret and urgent, the story of codes and ciphers.* Blue Ribbon Books, 1942.
2. Shannon, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.
3. Shannon, C. E. Prediction and entropy of printed english. *The Bell System Technical Journal*, 30:50–64, 1951.
4. Chomsky, N. *Syntactic Structures*, chapter 2-3, pages 13–18. Mouton, 1964.
5. Booth, T. Probabilistic representation of formal languages. In *Tenth Annual IEEE Symposium on Switching and Automata Theory*, October 1969.
6. Lari, K. and Young, S. J. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.
7. Jelinek, F., Lafferty, J. D., and Mercer, R. L. Basic methods of probabilistic context free grammars. Technical Report RC 16374 (72684), IBM, Yorktown Heights, New York 10598, 1990.
8. Joshi, A. K. An Introduction to Tree Adjoining Grammars. In Manaster-Ramer, A., editor, *Mathematics of Language.* John Benjamins, Amsterdam, 1987.
9. Schabes, Y., Abeillé, A., and Joshi, A. K. Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *Proceedings of the* 12th *International Conference on Computational Linguistics (COLING'88)*, Budapest, Hungary, August 1988.
10. Schabes, Y. *Mathematical and Computational Aspects of Lexicalized Grammars.* PhD thesis, University of Pennsylvania, Philadelphia, PA, August 1990. Available as technical report (MS-CIS-90-48, LINC LAB179) from the Department of Computer Science.
11. Joshi, A. K. and Schabes, Y. Tree-adjoining grammars and lexicalized grammars. In Nivat, M. and Podelski, A., editors, *Definability and Recognizability of Sets of Trees.* Elsevier, 1991. Forthcoming.
12. Resnik, P. Lexicalized tree-adjoining grammar for distributional analysis. In *Penn Review of Linguistics*, Spring 1991.
13. Vijay-Shanker, K. and Weir, D. J. Parsing constrained grammar formalisms, 1991. In preparation.
14. Aho, A. V. Indexed grammars — An extension to context free grammars. *J. ACM*, 15:647–671, 1968.
15. Gazdar, G. Applicability of indexed grammars to natural languages. Technical Report CSLI-85-34, Center for Study of Language and Information, 1985.

16. Vijay-Shanker, K. *A Study of Tree Adjoining Grammars.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1987.
17. Schabes, Y. An inside-outside algorithm for estimating the parameters of a hidden stochastic context-free grammar based on Earley's algorithm. Manuscript, 1991.
18. Vijay-Shanker, K. and Joshi, A. K. Some computational properties of Tree Adjoining Grammars. In 23rd *Meeting of the Association for Computational Linguistics*, pages 82–93, Chicago, Illinois, July 1985.
19. Baker, J. Trainable grammars for speech recognition. In Wolf, J. J. and Klatt, D. H., editors, *Speech communication papers presentaed at the 97th Meeting of the Acoustical Society of America*, MIT, Cambridge, MA, June 1979.
20. Pereira, F. and Schabes, Y. Inside-outisde reestimation from partially bracketed corpora. 1992. Also in these proceedings.
21. Joshi, A. K., Vijay-Shanker, K., and Weir, D. The convergence of mildly context-sensitive grammatical formalisms. In Sells, P., Shieber, S., and Wasow, T., editors, *Foundational Issues in Natural Language Processing.* MIT Press, Cambridge MA, 1991.

A. INSIDE PROBABILITIES

In the following, the inside and outside probabilities are relative to the input string w. $\mathcal{F}$ stands for the the set of foot nodes, $\mathcal{S}$ for the set of nodes on which substitution can occur, $\mathcal{R}$ for the set of root nodes of initial trees, and $\mathcal{A}$ for the set of non-terminal nodes of auxiliary trees. The inside probability can be computed bottom-up with the following recurrence equations. For all node η found in an elementary tree, it can be shown that:

1. If $b[\$\eta] \rightarrow a$, $I(b,\eta,i,-,-,l) = 1$ if $l = i+1$ and if $a = w_i^{i+1}$, 0 otherwise.
2. If $\eta_f \in \mathcal{F}$, $I(b,\eta_f,i,j,k,l) = 1$ if $i = j$ and if $k = l$, 0 otherwise.
3. If $b[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta_1]t[\$\eta_2]$: $I(b,\eta,i,j,k,l) =$
$$\sum_{m=k}^{l-1} I(t,\eta_1,i,j,k,m) \times I(t,\eta_2,m,-,-,l)$$
4. If $b[\cdot\cdot\eta] \rightarrow t[\$\eta_1]t[\cdot\cdot\eta_2]$, $I(b,\eta,i,j,k,l) =$
$$\sum_{m=i+1}^{j} I(t,\eta_1,i,-,-,m) \times I(t,\eta_2,m,j,k,l)$$
5. If $b[\$\eta] \rightarrow t[\$\eta_1]t[\$\eta_2]$, $I(b,\eta,i,-,-,l) =$
$$\sum_{m=i+1}^{l-1} I(t,\eta_1,i,-,-,m) \times I(t,\eta_2,m,-,-,l)$$
6. For all node η on which adjunction can be performed: $I(t,\eta,i,j,k,l) =$
$$I(b,\eta,i,j,k,l) \times P(t[\cdot\cdot\eta] \rightarrow b[\cdot\cdot\eta]) + \sum_{r=i}^{j}\sum_{s=k}^{l}\sum_{\eta_1} \left(\begin{array}{ll} & I(t,\eta_1,i,r,s,l) \\ \times & I(b,\eta,r,j,k,s) \\ \times & P(t[\cdot\cdot\eta] \rightarrow t[\cdot\cdot\eta\eta_1]) \end{array}\right)$$
7. For all node $\eta \in \mathcal{S}$: $I(t,\eta,i,-,-,l) =$
$$\sum_{\eta_1} I(t,\eta_1,i,-,-,l) \times P(t[\$\eta] \rightarrow t[\$\eta_1])$$
8. $I(t,\$,i,-,-,l) = \sum_{\eta} I(t,\eta,i,-,-,l) \times P(t[\$] \rightarrow t[\$\eta])$

B. OUTSIDE PROBABILITIES

The outside probabilities can be computed top-down recursively over smaller spans of the input string once the inside probabilities have been computed. First, by definition we have: $O(t,\$,0,-,-,N) = 1$. The following recurrence equations hold for all node η found in an elementary tree.

1. If $\eta \in \mathcal{R}$, $O(t,\eta,0,-,-,N) = P(t[\$] \rightarrow t[\$\eta])$.
And for all $(i,j) \neq (0,N)$, $O(t,\eta,i,-,-,j) = O(t,\eta_0,i,-,-,j) \times P(t[\$\eta_0] \rightarrow t[\$\eta])$
2. If η is an interior node which subsumes the foot node of the elementary tree it belongs to, $O(t,\eta,i,j,k,l) =$
$$\sum_{q=l+1}^{N} \left(\begin{array}{ll} & O(b,\eta_0,i,j,k,q) \\ \times & I(t,\eta_2,l,-,-,q) \\ \times & P(b[\cdot\cdot\eta_0] \rightarrow t[\cdot\cdot\eta]t[\$\eta_2]) \end{array}\right) + \sum_{p=0}^{i-1} \left(\begin{array}{ll} & O(b,\eta_0,p,j,k,l) \\ \times & I(t,\eta_1,p,-,-,i) \\ \times & P(b[\cdot\cdot\eta_0] \rightarrow t[\$\eta_1]t[\cdot\cdot\eta]) \end{array}\right)$$
3. If η is an interior node which does not subsume the foot node of the elementary tree it belongs to, we have: $O(t,\eta,i,-,-,l) =$
$$\sum_{q=l+1}^{N} \left(\begin{array}{ll} & O(b,\eta_0,i,-,-,q) \\ \times & I(t,\eta_2,l,-,-,q) \\ \times & P(b[\$\eta_0] \rightarrow t[\$\eta]t[\$\eta_2]) \end{array}\right) + \sum_{p=0}^{i-1} \left(\begin{array}{ll} & O(b,\eta_0,p,-,-,l) \\ \times & I(t,\eta_1,p,-,-,i) \\ \times & P(b[\$\eta_0] \rightarrow t[\$\eta_1]t[\$\eta]) \end{array}\right)$$
$$+ \sum_{j=l}^{N}\sum_{k=j+1}^{N}\sum_{q=k}^{N} \left(\begin{array}{ll} & O(b,\eta_0,i,j,k,q) \\ \times & I(t,\eta_2,l,j,k,q) \\ \times & P(b[\cdot\cdot\eta_0] \rightarrow t[\$\eta]t[\cdot\cdot\eta_2]) \end{array}\right) + \sum_{p=0}^{i-1}\sum_{j=p}^{i}\sum_{k=j}^{i} \left(\begin{array}{ll} & O(b,\eta_0,p,j,k,l) \\ \times & I(t,\eta_1,p,j,k,i) \\ \times & P(b[\cdot\cdot\eta_0] \rightarrow t[\cdot\cdot\eta_1]t[\$\eta]) \end{array}\right)$$
4. If $\eta \in \mathcal{A}$, then: $O(t,\eta,i,j,k,l) =$
$$\sum_{\eta_0}\sum_{p=j}^{k-1}\sum_{q=p+1}^{k} \left(\begin{array}{ll} & O(t,\eta_0,i,p,q,l) \\ \times & I(t,\eta_0,j,p,q,k) \\ \times & P(t[\cdot\cdot\eta_0] \rightarrow t[\cdot\cdot\eta_0\eta]) \end{array}\right) + \sum_{\eta_0} \left(\begin{array}{ll} & O(t,\eta_0,i,-,-,l) \\ \times & I(t,\eta_0,j,-,-,k) \\ \times & P(t[\$\eta_0] \rightarrow t[\$\eta_0\eta]) \end{array}\right)$$
5. If η is a node which subsumes the foot node of the elementary tree it belongs to, we have: $O(b,\eta,i,j,k,l) =$
$$O(t,\eta,i,j,k,l) \times P(t[\cdot\cdot\eta] \rightarrow b[\cdot\cdot\eta]) + \sum_{\eta_0}\sum_{p=0}^{i}\sum_{q=l}^{N} \left(\begin{array}{ll} & O(t,\eta,p,j,k,q) \\ \times & I(t,\eta_0,p,i,l,q) \\ \times & P(t[\cdot\cdot\eta_0] \rightarrow t[\cdot\cdot\eta_0\eta]) \end{array}\right)$$
6. And finally, if η is a node which does not subsume the foot node of the elementary tree it belongs to: $O(b,\eta,i,-,-,l) =$
$$O(t,\eta,i,-,-,l) \times P(t[\$\eta] \rightarrow b[\$\eta]) + \sum_{\eta_0}\sum_{p=0}^{i-1}\sum_{q=l}^{N} \left(\begin{array}{ll} & O(t,\eta,p,-,-,q) \\ \times & I(t,\eta_0,p,i,l,q) \\ \times & P(t[\$\eta_0] \rightarrow t[\$\eta_0\eta]) \end{array}\right)$$