Conditions on Consistency of Probabilistic Tree Adjoining Grammars*

Anoop Sarkar
Dept. of Computer and Information Science
University of Pennsylvania
200 South 33rd Street,
Philadelphia, PA 19104-6389 USA
anoop@linc.cis.upenn.edu

Abstract

Much of the power of probabilistic methods in modelling language comes from their ability to compare several derivations for the same string in the language. An important starting point for the study of such cross-derivational properties is the notion of *consistency*. The probability model defined by a probabilistic grammar is said to be *consistent* if the probabilities assigned to all the strings in the language sum to one. From the literature on probabilistic context-free grammars (CFGs), we know precisely the conditions which ensure that consistency is true for a given CFG. This paper derives the conditions under which a given probabilistic Tree Adjoining Grammar (TAG) can be shown to be consistent. It gives a simple algorithm for checking consistency and gives the formal justification for its correctness. The conditions derived here can be used to ensure that probability models that use TAGs can be checked for *deficiency* (i.e. whether any probability mass is assigned to strings that cannot be generated).

1 Introduction

Much of the power of probabilistic methods in modelling language comes from their ability to compare several derivations for the same string in the language. This cross-derivational power arises naturally from comparison of various derivational paths, each of which is a product of the probabilities associated with each step in each derivation. A common approach used to assign structure to language is to use a probabilistic grammar where each elementary rule or production is associated with a probability. Using such a grammar, a probability for each string in the language is computed. Assuming that the probability of each derivation of a sentence is well-defined, the probability of each string in the language is simply the sum of the probabilities of all derivations of the string. In general, for a probabilistic grammar G the language of G is denoted by $L(G)$. Then if a string v is in the language $L(G)$ the probabilistic grammar assigns v some non-zero probability.

There are several cross-derivational properties that can be studied for a given probabilistic grammar formalism. An important starting point for such studies is the notion of *consistency*. The probability model defined by a probabilistic grammar is said to be *consistent* if the probabilities assigned to all the strings in the language sum to 1. That is, if Pr defined by a probabilistic grammar, assigns a probability to each string $v \in \Sigma^*$, where $\Pr(v) = 0$ if $v \notin L(G)$, then

$$\sum_{v \in L(G)} \Pr(v) = 1 \qquad (1)$$

From the literature on probabilistic context-free grammars (CFGs) we know precisely the conditions which ensure that (1) is true for a given CFG. This paper derives the conditions under which a given probabilistic TAG can be shown to be consistent.

TAGs are important in the modelling of natural language since they can be easily lexicalized; moreover the trees associated with words can be used to encode argument and adjunct relations in various syntactic environments. This paper assumes some familiarity with the TAG formalism. (Joshi, 1988) and (Joshi and Schabes, 1992) are good introductions to the formalism and its linguistic relevance. TAGs have

* This research was partially supported by NSF grant SBR8920230 and ARO grant DAAH0404-94-G-0426. The author would like to thank Aravind Joshi, Jeff Reynar, Giorgio Satta, B. Srinivas, Fei Xia and the two anonymous reviewers for their valuable comments.

been shown to have relations with both phrase-structure grammars and dependency grammars (Rambow and Joshi, 1995) and can handle (non-projective) long distance dependencies.

Consistency of probabilistic TAGs has practical significance for the following reasons:

- The conditions derived here can be used to ensure that probability models that use TAGs can be checked for *deficiency*.
- Existing EM based estimation algorithms for probabilistic TAGs assume that the property of consistency holds (Schabes, 1992). EM based algorithms begin with an initial (usually random) value for each parameter. If the initial assignment causes the grammar to be inconsistent, then iterative re-estimation might converge to an inconsistent grammar[1].
- Techniques used in this paper can be used to determine consistency for other probability models based on TAGs (Carroll and Weir, 1997).

2 Notation

In this section we establish some notational conventions and definitions that we use in this paper. Those familiar with the TAG formalism only need to give a cursory glance through this section.

A probabilistic TAG is represented by $(N, \Sigma, \mathcal{I}, \mathcal{A}, S, \phi)$ where N, Σ are, respectively, non-terminal and terminal symbols. $\mathcal{I} \cup \mathcal{A}$ is a set of trees termed as *elementary trees*. We take V to be the set of all nodes in all the elementary trees. For each leaf $A \in V$, $label(A)$ is an element from $\Sigma \cup \{\epsilon\}$, and for each other node A, $label(A)$ is an element from N. S is an element from N which is a distinguished start symbol. The root node A of every initial tree which can start a derivation must have $label(A) = S$.

$\mathcal{I}$ are termed *initial trees* and $\mathcal{A}$ are *auxiliary trees* which can rewrite a tree node $A \in V$. This rewrite step is called **adjunction**. ϕ is a function which assigns each adjunction with a probability and denotes the set of parameters in the model. In practice, TAGs also allow a leaf nodes A such that $label(A)$ is an element from N. Such nodes A are rewritten with initial trees from $\mathcal{I}$ using the rewrite step called **substitution**. Except in one special case, we will not need to treat substitution as being distinct from adjunction.

For $t \in \mathcal{I} \cup \mathcal{A}$, $\mathcal{A}(t)$ are the nodes in tree t that can be modified by adjunction. For $label(A) \in N$ we denote $Adj(label(A))$ as the set of trees that can adjoin at node $A \in V$. The adjunction of t into $N \in V$ is denoted by $N \mapsto t$. No adjunction at $N \in V$ is denoted by $N \mapsto nil$. We assume the following properties hold for every probabilistic TAG G that we consider:

1. G is *lexicalized*. There is at least one leaf node a that lexicalizes each elementary tree, i.e. $a \in \Sigma$.
2. G is *proper*. For each $N \in V$,
$$\phi(N \mapsto nil) + \sum_t \phi(N \mapsto t) = 1$$
3. Adjunction is prohibited on the foot node of every auxiliary tree. This condition is imposed to avoid unnecessary ambiguity and can be easily relaxed.
4. There is a distinguished non-lexicalized initial tree τ such that each initial tree rooted by a node A with $label(A) = S$ substitutes into τ to complete the derivation. This ensures that probabilities assigned to the input string at the start of the derivation are well-formed.

We use symbols $S, A, B, \ldots$ to range over V, symbols $a, b, c, \ldots$ to range over Σ. We use $t_1, t_2, \ldots$ to range over $I \cup A$ and ϵ to denote the empty string. We use X_i to range over all i nodes in the grammar.

3 Applying probability measures to Tree Adjoining Languages

To gain some intuition about probability assignments to languages, let us take for example, a language well known to be a tree adjoining language:

$$L(G) = \{a^n b^n c^n d^n | n \geq 1\}$$

[1] Note that for CFGs it has been shown in (Chaudhari et al., 1983; Sánchez and Benedí, 1997) that inside-outside reestimation can be used to avoid inconsistency. We will show later in the paper that the method used to show consistency in this paper precludes a straightforward extension of that result for TAGs.

It seems that we should be able to use a function ψ to assign any probability distribution to the strings in $L(G)$ and then expect that we can assign appropriate probabilites to the adjunctions in G such that the language generated by G has the same distribution as that given by ψ. However a function ψ that grows smaller by repeated multiplication as the inverse of an exponential function cannot be matched by any TAG because of the *constant growth* property of TAGs (see (Vijay-Shanker, 1987), p. 104). An example of such a function ψ is a simple Poisson distribution (2), which in fact was also used as the counterexample in (Booth and Thompson, 1973) for CFGs, since CFGs also have the constant growth property.

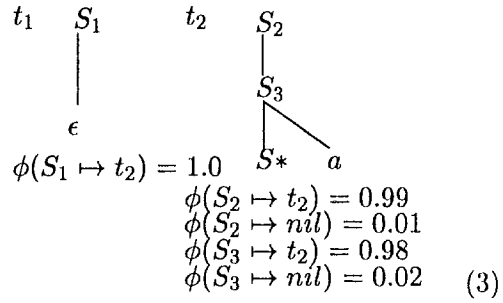
$$\psi(a^n b^n c^n d^n) = \frac{1}{e \cdot n!} \tag{2}$$

This shows that probabilistic TAGs, like CFGs, are constrained in the probabilistic languages that they can recognize or learn. As shown above, a probabilistic language can fail to have a generating probabilistic TAG.

The reverse is also true: some probabilistic TAGs, like some CFGs, fail to have a corresponding probabilistic language, i.e. they are not consistent. There are two reasons why a probabilistic TAG could be inconsistent: "dirty" grammars, and destructive or incorrect probability assignments.

"Dirty" grammars. Usually, when applied to language, TAGs are lexicalized and so probabilities assigned to trees are used only when the words anchoring the trees are used in a derivation. However, if the TAG allows non-lexicalized trees, or more precisely, auxiliary trees with no yield, then looping adjunctions which never generate a string are possible. However, this can be detected and corrected by a simple search over the grammar. Even in lexicalized grammars, there could be some auxiliary trees that are assigned some probability mass but which can never adjoin into another tree. Such auxiliary trees are termed *unreachable* and techniques similar to the ones used in detecting unreachable productions in CFGs can be used here to detect and eliminate such trees.

Destructive probability assignments. This problem is a more serious one, and is the main subject of this paper. Consider the probabilistic TAG shown in (3)[2].



(3)

Consider a derivation in this TAG as a generative process. It proceeds as follows: node S_1 in t_1 is rewritten as t_2 with probability 1.0. Node S_2 in t_2 is 99 times more likely than not to be rewritten as t_2 itself, and similarly node S_3 is 49 times more likely than not to be rewritten as t_2. This however, creates two more instances of S_2 and S_3 with same probabilities. This continues, creating multiple instances of t_2 at each level of the derivation process with each instance of t_2 creating two more instances of itself. The grammar itself is not malicious; the probability assignments are to blame. It is important to note that inconsistency is a problem even though for any given string there are only a finite number of derivations, all halting. Consider the probability mass function (*pmf*) over the set of all derivations for this grammar. An inconsistent grammar would have a *pmf* which assigns a large portion of probability mass to derivations that are non-terminating. This means there is a finite probability the generative process can enter a generation sequence which has a finite probability of non-termination.

4 Conditions for Consistency

A probabilistic TAG G is *consistent* if and only if:

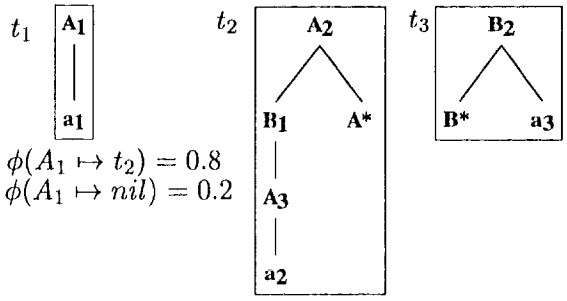
$$\sum_{v \in L(G)} \Pr(v) = 1 \tag{4}$$

where $\Pr(v)$ is the probability assigned to a string in the language. If a grammar G does not satisfy this condition, G is said to be inconsistent.

To explain the conditions under which a probabilistic TAG is consistent we will use the TAG

[2]The subscripts are used as a simple notation to uniquely refer to the nodes in each elementary tree. They are not part of the node label for purposes of adjunction.

in (5) as an example.



$$\begin{array}{ll} \phi(A_1 \mapsto t_2) = 0.8 & \\ \phi(A_1 \mapsto nil) = 0.2 & \\ \phi(A_2 \mapsto t_2) = 0.2 & \phi(B_2 \mapsto t_3) = 0.1 \\ \phi(A_2 \mapsto nil) = 0.8 & \phi(B_2 \mapsto nil) = 0.9 \\ \phi(B_1 \mapsto t_3) = 0.2 & \\ \phi(B_1 \mapsto nil) = 0.8 & \\ \phi(A_3 \mapsto t_2) = 0.4 & \\ \phi(A_3 \mapsto nil) = 0.6 & \end{array} \quad (5)$$

From this grammar, we compute a square matrix $\mathcal{M}$ which of size $|V|$, where V is the set of nodes in the grammar that can be rewritten by adjunction. Each $\mathcal{M}_{ij}$ contains the expected value of obtaining node X_j when node X_i is rewritten by adjunction at each level of a TAG derivation. We call $\mathcal{M}$ the stochastic *expectation matrix* associated with a probabilistic TAG.

To get $\mathcal{M}$ for a grammar we first write a matrix $\mathbf{P}$ which has $|V|$ rows and $|I \cup A|$ columns. An element $\mathbf{P}_{ij}$ corresponds to the probability of adjoining tree t_j at node X_i, i.e. $\phi(X_i \mapsto t_j)$[3].

$$\mathbf{P} = \begin{array}{c} \\ A_1 \\ A_2 \\ B_1 \\ A_3 \\ B_2 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{bmatrix} 0 & 0.8 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.2 \\ 0 & 0.4 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \end{array}$$

We then write a matrix $\mathbf{N}$ which has $|I \cup A|$ rows and $|V|$ columns. An element $\mathbf{N}_{ij}$ is 1.0 if node X_j is a node in tree t_i.

$$\mathbf{N} = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \end{array} \begin{array}{c} \begin{array}{ccccc} A_1 & A_2 & B_1 & A_3 & B_2 \end{array} \\ \begin{bmatrix} 1.0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 1.0 & 1.0 & 0 \\ 0 & 0 & 0 & 0 & 1.0 \end{bmatrix} \end{array}$$

Then the stochastic expectation matrix $\mathcal{M}$ is simply the product of these two matrices.

[3]Note that $\mathbf{P}$ is not a row stochastic matrix. This is an important difference in the construction of $\mathcal{M}$ for TAGs when compared to CFGs. We will return to this point in §5.

$$\mathcal{M} = \mathbf{P} \cdot \mathbf{N} = \begin{array}{c} \\ A_1 \\ A_2 \\ B_1 \\ A_3 \\ B_2 \end{array} \begin{array}{c} \begin{array}{ccccc} A_1 & A_2 & B_1 & A_3 & B_2 \end{array} \\ \begin{bmatrix} 0 & 0.8 & 0.8 & 0.8 & 0 \\ 0 & 0.2 & 0.2 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 0.2 \\ 0 & 0.4 & 0.4 & 0.4 & 0 \\ 0 & 0 & 0 & 0 & 0.1 \end{bmatrix} \end{array}$$

By inspecting the values of $\mathcal{M}$ in terms of the grammar probabilities indicates that $\mathcal{M}_{ij}$ contains the values we wanted, i.e. expectation of obtaining node A_j when node A_i is rewritten by adjunction at each level of the TAG derivation process.

By construction we have ensured that the following theorem from (Booth and Thompson, 1973) applies to probabilistic TAGs. A formal justification for this claim is given in the next section by showing a reduction of the TAG derivation process to a multitype Galton-Watson branching process (Harris, 1963).

Theorem 4.1 *A probabilistic grammar is consistent if the* spectral radius $\rho(\mathcal{M}) < 1$, *where $\mathcal{M}$ is the stochastic expectation matrix computed from the grammar. (Booth and Thompson, 1973; Soule, 1974)*

This theorem provides a way to determine whether a grammar is consistent. All we need to do is compute the spectral radius of the square matrix $\mathcal{M}$ which is equal to the modulus of the largest eigenvalue of $\mathcal{M}$. If this value is less than one then the grammar is consistent[4]. Computing consistency can bypass the computation of the eigenvalues for $\mathcal{M}$ by using the following theorem by Geršgorin (see (Horn and Johnson, 1985; Wetherell, 1980)).

Theorem 4.2 *For any square matrix $\mathcal{M}$, $\rho(\mathcal{M}) < 1$ if and only if there is an $n \geq 1$ such that the sum of the absolute values of the elements of each row of $\mathcal{M}^n$ is less than one. Moreover, any $n' > n$ also has this property. (Geršgorin, see (Horn and Johnson, 1985; Wetherell, 1980))*

[4]The grammar may be consistent when the spectral radius is exactly one, but this case involves many special considerations and is not considered in this paper. In practice, these complicated tests are probably not worth the effort. See (Harris, 1963) for details on how this special case can be solved.

This makes for a very simple algorithm to check consistency of a grammar. We sum the values of the elements of each row of the stochastic expectation matrix $\mathcal{M}$ computed from the grammar. If *any* of the row sums are greater than one then we compute $\mathcal{M}^2$, repeat the test and compute $\mathcal{M}^{2^2}$ if the test fails, and so on until the test succeeds[5]. The algorithm does not halt if $\rho(\mathcal{M}) \geq 1$. In practice, such an algorithm works better in the average case since computation of eigenvalues is more expensive for very large matrices. An upper bound can be set on the number of iterations in this algorithm. Once the bound is passed, the exact eigenvalues can be computed.

For the grammar in (5) we computed the following stochastic expectation matrix:

$$\mathcal{M} = \begin{bmatrix} 0 & 0.8 & 0.8 & 0.8 & 0 \\ 0 & 0.2 & 0.2 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 0.2 \\ 0 & 0.4 & 0.4 & 0.4 & 0 \\ 0 & 0 & 0 & 0 & 0.1 \end{bmatrix}$$

The first row sum is 2.4. Since the sum of each row must be less than one, we compute the power matrix $\mathcal{M}^2$. However, the sum of one of the rows is still greater than 1. Continuing we compute $\mathcal{M}^{2^2}$.

$$\mathcal{M}^{2^2} = \begin{bmatrix} 0 & 0.1728 & 0.1728 & 0.1728 & 0.0688 \\ 0 & 0.0432 & 0.0432 & 0.0432 & 0.0172 \\ 0 & 0 & 0 & 0 & 0.0002 \\ 0 & 0.0864 & 0.0864 & 0.0864 & 0.0344 \\ 0 & 0 & 0 & 0 & 0.0001 \end{bmatrix}$$

This time all the row sums are less than one, hence $\rho(\mathcal{M}) < 1$. So we can say that the grammar defined in (5) is consistent. We can confirm this by computing the eigenvalues for $\mathcal{M}$ which are $0, 0, 0.6, 0$ and 0.1, all less than 1.

Now consider the grammar (3) we had considered in Section 3. The value of $\mathcal{M}$ for that grammar is computed to be:

$$\mathcal{M}_{(3)} = \begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \end{array} \begin{array}{c} \begin{array}{ccc} S_1 & S_2 & S_3 \end{array} \\ \begin{bmatrix} 0 & 1.0 & 1.0 \\ 0 & 0.99 & 0.99 \\ 0 & 0.98 & 0.98 \end{bmatrix} \end{array}$$

[5]We compute $\mathcal{M}^{2^2}$ and subsequently only successive powers of 2 because Theorem 4.2 holds for any $n' > n$. This permits us to use a single matrix at each step in the algorithm.

The eigenvalues for the expectation matrix $\mathcal{M}$ computed for the grammar (3) are 0, 1.97 and 0. The largest eigenvalue is greater than 1 and this confirms (3) to be an inconsistent grammar.

5 TAG Derivations and Branching Processes

To show that Theorem 4.1 in Section 4 holds for any probabilistic TAG, it is sufficient to show that the derivation process in TAGs is a Galton-Watson branching process.

A Galton-Watson branching process (Harris, 1963) is simply a model of processes that have objects that can produce additional objects of the same kind, i.e. recursive processes, with certain properties. There is an initial set of objects in the 0-th generation which produces with some probability a first generation which in turn with some probability generates a second, and so on. We will denote by vectors $Z_0, Z_1, Z_2, \ldots$ the 0-th, first, second, ... generations. There are two assumptions made about $Z_0, Z_1, Z_2, \ldots$:

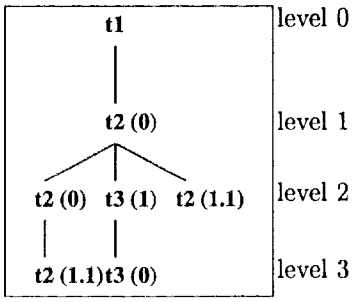
1. The size of the n-th generation does not influence the probability with which any of the objects in the $(n+1)$-th generation is produced. In other words, $Z_0, Z_1, Z_2, \ldots$ form a Markov chain.

2. The number of objects born to a parent object does not depend on how many other objects are present at the same level.

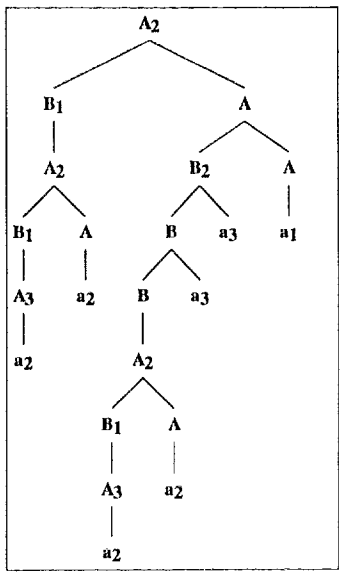
We can associate a generating function for each level Z_i. The value for the vector Z_n is the value assigned by the n-th iterate of this generating function. The expectation matrix $\mathcal{M}$ is defined using this generating function.

The theorem attributed to Galton and Watson specifies the conditions for the probability of extinction of a family starting from its 0-th generation, assuming the branching process represents a family tree (i.e, respecting the conditions outlined above). The theorem states that $\rho(\mathcal{M}) \leq 1$ when the probability of extinction is

1.0.



(6)



(7)

The assumptions made about the generating process intuitively holds for probabilistic TAGs. (6), for example, depicts a derivation of the string $a_2a_2a_2a_2a_3a_3a_1$ by a sequence of adjunctions in the grammar given in (5)[6]. The parse tree derived from such a sequence is shown in Fig. 7. In the derivation tree (6), nodes in the trees at each level i are rewritten by adjunction to produce a level $i+1$. There is a final level 4 in (6) since we also consider the probability that a node is not rewritten further, i.e. $\Pr(A \mapsto nil)$ for each node A.

We give a precise statement of a TAG derivation process by defining a generating function for the levels in a derivation tree. Each level i in the TAG derivation tree then corresponds to Z_i in the Markov chain of branching processes. This is sufficient to justify the use of Theorem 4.1 in Section 4. The conditions on the probability of extinction then relates to the probability that TAG derivations for a probabilistic TAG will not recurse infinitely. Hence the probability of extinction is the same as the probability that a probabilistic TAG is consistent.

For each $X_j \in V$, where V is the set of nodes in the grammar where adjunction can occur, we define the k-argument *adjunction generating function* over variables $s_1, \ldots, s_k$ corresponding to the k nodes in V.

$$g_j(s_1, \ldots, s_k) = \sum_{t \in Adj(X_j) \cup \{nil\}} \phi(X_j \mapsto t) \cdot s_1^{r_1(t)} \cdots s_k^{r_k(t)}$$

where, $r_j(t) = 1$ iff node X_j is in tree t, $r_j(t) = 0$ otherwise.

For example, for the grammar in (5) we get the following adjunction generating functions taking the variable s_1, s_2, s_3, s_4, s_5 to represent the nodes A_1, A_2, B_1, A_3, B_2 respectively.

$$\begin{aligned}
&g_1(s_1, \ldots, s_5) = \\
&\quad \phi(A_1 \mapsto t_2) \cdot s_2 \cdot s_3 \cdot s_4 + \phi(A_1 \mapsto nil) \\
&g_2(s_1, \ldots, s_5) = \\
&\quad \phi(A_2 \mapsto t_2) \cdot s_2 \cdot s_3 \cdot s_4 + \phi(A_2 \mapsto nil) \\
&g_3(s_1, \ldots, s_5) = \\
&\quad \phi(B_1 \mapsto t_3) \cdot s_5 + \phi(B_1 \mapsto nil) \\
&g_4(s_1, \ldots, s_5) = \\
&\quad \phi(A_3 \mapsto t_2) \cdot s_2 \cdot s_3 \cdot s_4 + \phi(A_3 \mapsto nil) \\
&g_5(s_1, \ldots, s_5) = \\
&\quad \phi(B_2 \mapsto t_3) \cdot s_5 + \phi(B_2 \mapsto nil)
\end{aligned}$$

The n-th level generating function $G_n(s_1, \ldots, s_k)$ is defined recursively as follows.

$$\begin{aligned}
G_0(s_1, \ldots, s_k) &= s_1 \\
G_1(s_1, \ldots, s_k) &= g_1(s_1, \ldots, s_k) \\
G_n(s_1, \ldots, s_k) &= G_{n-1}[g_1(s_1, \ldots, s_k), \ldots, \\
&\qquad g_k(s_1, \ldots, s_k)]
\end{aligned}$$

For the grammar in (5) we get the following level generating functions.

$$G_0(s_1, \ldots, s_5) = s_1$$

[6]The numbers in parentheses next to the tree names are node addresses where each tree has adjoined into its parent. Recall the definition of node addresses in Section 2.

$$
\begin{aligned}
G_1(s_1,\ldots,s_5) &= g_1(s_1,\ldots,s_5) \\
&= \phi(A_1 \mapsto t_2)\cdot s_2 \cdot s_3 \cdot s_4 + \phi(A_1 \mapsto nil) \\
&= 0.8 \cdot s_2 \cdot s_3 \cdot s_4 + 0.2 \\
G_2(s_1,\ldots,s_5) &= \\
&\quad \phi(A_2 \mapsto t_2)[g_2(s_1,\ldots,s_5)][g_3(s_1,\ldots,s_5)] \\
&\quad [g_4(s_1,\ldots,s_5)] + \phi(A_2 \mapsto nil) \\
&= 0.08 s_2^2 s_3^2 s_4^2 s_5 + 0.03 s_2^2 s_3^2 s_4^2 + 0.04 s_2 s_3 s_4 s_5 + \\
&\quad 0.18 s_2 s_3 s_4 + 0.04 s_5 + 0.196
\end{aligned}
$$

...

Examining this example, we can express $G_i(s_1,\ldots,s_k)$ as a sum $D_i(s_1,\ldots,s_k) + C_i$, where C_i is a constant and $D_i(\cdot)$ is a polynomial with no constant terms. A probabilistic TAG will be consistent if these recursive equations terminate, i.e. iff

$$lim_{i\to\infty} D_i(s_1,\ldots,s_k) \to 0$$

We can rewrite the level generation functions in terms of the stochastic expectation matrix $\mathcal{M}$, where each element $m_{i,j}$ of $\mathcal{M}$ is computed as follows (cf. (Booth and Thompson, 1973)).

$$m_{i,j} = \left.\frac{\partial g_i(s_1,\ldots,s_k)}{\partial s_j}\right|_{s_1,\ldots,s_k=1} \tag{8}$$

The limit condition above translates to the condition that the spectral radius of $\mathcal{M}$ must be less than 1 for the grammar to be consistent.

This shows that Theorem 4.1 used in Section 4 to give an algorithm to detect inconsistency in a probabilistic holds for any given TAG, hence demonstrating the correctness of the algorithm.

Note that the formulation of the adjunction generating function means that the values for $\phi(X \mapsto nil)$ for all $X \in V$ do not appear in the expectation matrix. This is a crucial difference between the test for consistency in TAGs as compared to CFGs. For CFGs, the expectation matrix for a grammar G can be interpreted as the contribution of each non-terminal to the derivations for a sample set of strings drawn from $L(G)$. Using this it was shown in (Chaudhari et al., 1983) and (Sánchez and Benedí, 1997) that a single step of the inside-outside algorithm implies consistency for a probabilistic CFG. However, in the TAG case, the inclusion of values for $\phi(X \mapsto nil)$ (which is essential if we are to interpret the expectation matrix in terms of derivations over a sample set of strings) means that we cannot use the method used in (8) to compute the expectation matrix and furthermore the limit condition will not be convergent.

6 Conclusion

We have shown in this paper the conditions under which a given probabilistic TAG can be shown to be consistent. We gave a simple algorithm for checking consistency and gave the formal justification for its correctness. The result is practically significant for its applications in checking for *deficiency* in probabilistic TAGs.

References

T. L. Booth and R. A. Thompson. 1973. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, C-22(5):442–450, May.

J. Carroll and D. Weir. 1997. Encoding frequency information in lexicalized grammars. In *Proc. 5th Int'l Workshop on Parsing Technologies IWPT-97*, Cambridge, Mass.

R. Chaudhari, S. Pham, and O. N. Garcia. 1983. Solution of an open problem on probabilistic grammars. *IEEE Transactions on Computers*, C-32(8):748–750, August.

T. E. Harris. 1963. *The Theory of Branching Processes.* Springer-Verlag, Berlin.

R. A. Horn and C. R. Johnson. 1985. *Matrix Analysis.* Cambridge University Press, Cambridge.

A. K. Joshi and Y. Schabes. 1992. Tree-adjoining grammar and lexicalized grammars. In M. Nivat and A. Podelski, editors, *Tree automata and languages*, pages 409–431. Elsevier Science.

A. K. Joshi. 1988. An introduction to tree adjoining grammars. In A. Manaster-Ramer, editor, *Mathematics of Language.* John Benjamins, Amsterdam.

O. Rambow and A. Joshi. 1995. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In Leo Wanner, editor, *Current Issues in Meaning-Text Theory.* Pinter, London.

J.-A. Sánchez and J.-M. Benedí. 1997. Consistency of stochastic context-free grammars from probabilistic estimation based on growth transformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):1052–1055, September.

Y. Schabes. 1992. Stochastic lexicalized tree-adjoining grammars. In *Proc. of COLING '92*, volume 2, pages 426–432, Nantes, France.

S. Soule. 1974. Entropies of probabilistic grammars. *Inf. Control*, 25:55–74.

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

C. S. Wetherell. 1980. Probabilistic languages: A review and some open questions. *Computing Surveys*, 12(4):361–379.