Tree-Adjoining Grammars Are Not Closed Under Strong Lexicalization

Marco Kuhlmann*
Uppsala University

Giorgio Satta**
University of Padua

A lexicalized tree-adjoining grammar is a tree-adjoining grammar where each elementary tree contains some overt lexical item. Such grammars are being used to give lexical accounts of syntactic phenomena, where an elementary tree defines the domain of locality of the syntactic and semantic dependencies of its lexical items. It has been claimed in the literature that for every tree-adjoining grammar, one can construct a strongly equivalent lexicalized version. We show that such a procedure does not exist: Tree-adjoining grammars are not closed under strong lexicalization.

1. Introduction

Many contemporary linguistic theories give lexical accounts of syntactic phenomena, where complex syntactic structures are analyzed as the combinations of elementary structures taken from a finite lexicon. In the computational linguistics community, this trend has been called **lexicalization**, and has been extensively investigated since the 1990s. From a mathematical perspective, the main question that arises in the context of lexicalization is whether the restriction of a given class of grammars to lexicalized form has any impact on the generative or computational properties of the formalism.

As a simple example, consider the class of context-free grammars (CFGs). Recall that a CFG is in Greibach normal form if the right-hand side of every rule in the grammar starts with a terminal symbol, representing an overt lexical item. Although several procedures for casting a CFG in Greibach normal form exist, all of them substantially alter the structure of the parse trees of the source grammar. In technical terms, these procedures provide a **weak lexicalization** of the source grammar (because the string language is preserved) but not a **strong lexicalization** (because the sets of parse trees that the two grammars assign to the common string language are not the same). Strong lexicalization is highly relevant for natural language processing, however, where the parse tree assigned by a grammar represents the syntactic analysis of interest, and is used by other modules such as semantic interpretation or translation. In this article, we investigate the problem of strong lexicalization.

* Department of Linguistics and Philology, Box 635, 75126 Uppsala, Sweden.
E-mail: marco.kuhlmann@lingfil.uu.se.
** Department of Information Engineering, via Gradenigo 6/A, 35131 Padova, Italy.
E-mail: satta@dei.unipd.it.

Submission received: 16 July 2011; accepted for publication: 10 September 2011.

Two important results about strong lexicalization have been obtained by Schabes (1990). The first result is that CFGs are not closed under strong lexicalization. (The author actually shows a stronger result involving a formalism called tree substitution grammar, as will be discussed in detail in Section 3.) Informally, this means that we cannot cast a CFG G in a special form in which each rule has an overt lexical item in its right-hand side, under the restriction that the new grammar generates exactly the same set of parse trees as G. As a special case, this entails that no procedure can cast a CFG in Greibach normal form, under the additional condition that the generated parse trees are preserved.

The second result obtained by Schabes concerns the relation between CFGs and the class of tree-adjoining grammars (TAGs) (Joshi, Levy, and Takahashi 1975; Joshi and Schabes 1997). A TAG consists of a finite set of elementary trees, which are phrase structure trees of unbounded depth, and allows for the combination of these trees by means of two operations called substitution and adjunction (described in more detail in the next section). A lexicalized TAG is one where each elementary tree contains at least one overt lexical item called the anchor of the tree; the elementary tree is intended to encapsulate the syntactic and semantic dependencies of its anchor. Because CFG rules can be viewed as elementary trees of depth one, and because context-free rewriting can be simulated by the substitution operation defined for TAGs, we can view any CFG as a special TAG. Under this view, one can ask whether lexicalized TAGs can provide a strong lexicalization of CFGs. Schabes' second result is that this is indeed the case. This means that, given a CFG G, one can always construct a lexicalized TAG generating the same set of parse trees as G, and consequently the same string language.

Following from this result, there arose the possibility of establishing a third result, stating that TAGs are closed under strong lexicalization. Schabes (1990) states that this is the case, and provides an informal argument to justify the claim. The same claim still appears in two subsequent publications (Joshi and Schabes 1992, 1997), but no precise proof of it has appeared until now. We speculate that the claim could be due to the fact that adjunction is more powerful than substitution with respect to weak generative capacity. It turns out, however, that when it comes to strong generative capacity, adjunction also shares some of the restrictions of substitution. This observation leads to the main result of this article: TAGs are *not* closed under strong lexicalization. In other words, there are TAGs that lack a strongly equivalent lexicalized version.

In the same line of investigation, Schabes and Waters (1995) introduce a restricted variant of TAG called tree insertion grammars (TIGs). This formalism severely restricts the adjunction operation originally defined for TAGs, in such a way that the class of generated string languages, as well as the class of generated parse trees, are the same as those of CFGs. Schabes and Waters then conjecture that TIGs are closed under strong lexicalization. In this article we also disprove their conjecture.

2. Preliminaries

We assume familiarity with the TAG formalism; for a survey, we refer the reader to Joshi and Schabes (1997). We briefly introduce here the basic terminology and notation for TAG that we use in this article.

2.1 Basic Definitions

A TAG is a rewriting system that derives trees starting from a finite set of elementary trees. Elementary trees are trees of finite but arbitrary depth, with internal nodes labeled

with nonterminal symbols and frontier nodes labeled with terminal and nonterminal symbols. Each elementary tree is either an **initial tree** or else an **auxiliary tree**. Initial trees serve as the starting point for derivations, and may combine with other trees by means of an operation called **substitution**. Tree substitution replaces a node labeled with a nonterminal A in the frontier of some target tree with an initial tree whose root is labeled with A. The nodes that are the target of the substitution operation are identified by a down arrow ($\downarrow$). The substitution operation is illustrated in the left half of Figure 1.

Auxiliary trees are elementary trees in which a special node in the frontier has the same nonterminal label as the root node. This special node is called the foot node and is identified by an asterisk ($*$). Auxiliary trees may combine with other trees by means of an operation called **adjunction**. The adjunction operation entails splitting some target tree at an internal node with label A, and inserting an auxiliary tree whose root (and foot) node is labeled with A. The adjunction operation is illustrated in the right half of Figure 1.

A derivation in a TAG can be specified by a **derivation tree** d; this is a rooted tree whose nodes are labeled with (instances of) elementary trees, and whose edges are labeled with (addresses of) nodes at which substitution or adjunction takes place. More specifically, an edge $v \rightarrow_u v'$ in d represents the information that the elementary tree at v' is substituted at or adjoined into node u of the elementary tree at v. When we combine the elementary trees of our TAG as specified by d, we obtain a (unique) phrase structure tree called the **derived tree** associated with d, which we denote as $t(d)$.

We use the symbol γ as a variable ranging over elementary trees, α as a variable ranging over initial trees, and β as a variable ranging over auxiliary trees. We also use the symbols u and v as variables ranging over nodes of generic trees (elementary, derived, or derivation trees). For an elementary tree γ, a derivation tree d is said to have type γ if the root node of d is labeled with γ. A derivation tree d is called **sentential** if d is of some type γ, and the root node of γ is labeled with the start symbol of the grammar, denoted as S.

A node u in an elementary tree γ may be annotated with an **adjunction constraint**, which for purposes here is a label in the set $\{\mathrm{NA}, \mathrm{OA}\}$. The label NA denotes Null Adjunction, forbidding adjunction at u; the label OA denotes Obligatory Adjunction, forcing adjunction at u. A derivation tree d is called **saturated** if, at each node v of d there is an arc $v \rightarrow_u v'$, for some v', for every node u of the elementary tree at v that requires substitution or is annotated with an OA constraint.

For a TAG G, we denote by $T(G)$ the set of all the derived trees t such that $t = t(d)$ for some sentential and saturated derivation tree d obtained in G. Each such derived tree is

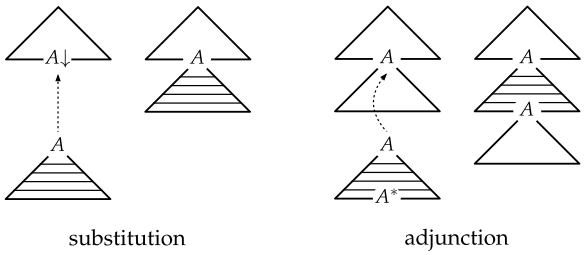


Figure 1
Combination operations in TAG.

(uniquely) associated with a string $y(t)$ called the **yield** of t, obtained by concatenating all terminal symbols labeling the frontier of t, from left to right. The string language generated by G is the set

$$L(G) = \{ y(t) \mid t \in T(G) \}$$

A TAG G is said to be **finitely ambiguous** if, for every string $w \in L(G)$, the subset of those trees in $T(G)$ that have w as their yield is finite.

An elementary tree γ of G is called **useless** if γ never occurs in a sentential and saturated derivation tree of G, that is, if no sentential and saturated derivation of G uses γ. A grammar G is called **reduced** if none of its elementary trees is useless. Throughout this article we shall assume that the grammars that we deal with are reduced.

2.2 Lexicalization

In a tree, a node labeled with a terminal symbol is called a **lexical node**. A TAG is called **lexicalized** if each of its elementary trees has at least one lexical node. Observe that a lexicalized grammar cannot generate the empty string, denoted by ε, because every derived tree yields at least one lexical element. Similarly, a lexicalized grammar is always finitely ambiguous, because the length of the generated strings provides an upper bound on the size of the associated derived trees. Let $\mathcal{G}$ and $\mathcal{G}'$ be two subclasses of the class of all TAGs. We say that $\mathcal{G}'$ **strongly lexicalizes** $\mathcal{G}$, if, for every grammar $G \in \mathcal{G}$ that is finitely ambiguous and that satisfies $\varepsilon \notin L(G)$, there exists a lexicalized grammar $G' \in \mathcal{G}'$ such that $T(G') = T(G)$. We also say that $\mathcal{G}$ is **closed under strong lexicalization** if the class $\mathcal{G}$ strongly lexicalizes itself.

Using this terminology, we can now restate the two main results obtained by Schabes (1990) about strong lexicalization for subclasses of TAGs, already mentioned in the Introduction. The first result states that the class of CFGs is not closed under strong lexicalization. Here we view a CFG as a special case of a TAG using only substitution and elementary trees of depth one. Informally, this means that we cannot cast a CFG G in a special form in which each rule has an overt lexical item in its right-hand side, under the restriction that the new grammar generates exactly the same tree set as G. The second result is that the class of TAGs strongly lexicalizes the class of **tree substitution grammars** (TSGs). The latter class is defined as the class of all TAGs that use substitution as the only tree combination operation, and thus includes all context-free grammars. This means that, given a TSG or a CFG G, we can always construct a TAG that is lexicalized and that generates exactly the same tree set as G.

3. Tree Substitution Grammars Are Not Closed Under Strong Lexicalization

Before turning to our main result in Section 4, we find it useful to technically revisit the related result for TSGs.

Theorem 1
Tree substitution grammars are not closed under strong lexicalization.

To prove this result, Schabes (1990) uses a proof by contradiction: The author considers a specific TSG G_1, reported in Figure 2. It is not difficult to see that G_1 is finitely ambiguous and that $\varepsilon \notin L(G_1)$. The author then assumes that G_1 can be lexicalized by another TSG,

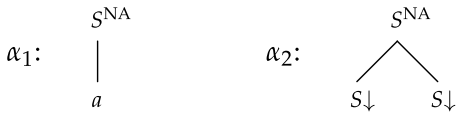


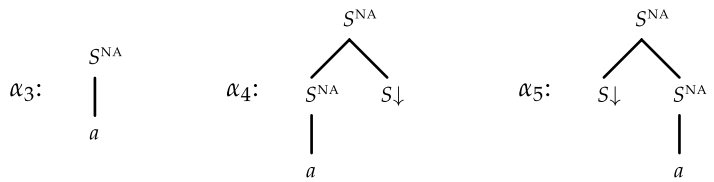
Figure 2
The counterexample tree substitution grammar G_1.

and derives a contradiction. We provide here an alternative, direct proof of Theorem 1. This alternative proof will be generalized in Section 4 to obtain the main result of this article.

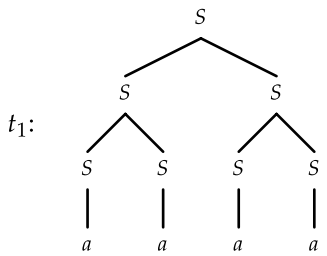
We use the following notation. For a derived tree t and a terminal symbol a, we write $Nodes(a, t)$ to denote the set of all nodes in t that are labeled with a. Furthermore, for a node u of t we write $depth(u, t)$ to denote the length of the unique path from the root node of t leading to u.

3.1 Intuition

In order to convey the basic idea behind Schabes's proof and our alternative version herein, we first consider a specific candidate grammar for the lexicalization of G_1. For example, one might think that the following TSG G_1' lexicalizes G_1:



This grammar is obtained from G_1 by taking the lexicalized tree α_1, as well as every elementary tree that can be obtained by substituting α_1 into the non-lexicalized tree α_2. The grammar G_1' only generates a subset of the trees generated by G_1, however. The following tree, for example, cannot be generated by G_1':



To see this, we reason as follows. Consider a lexical node v in an elementary tree γ of G_1', and let t be a tree obtained by substituting some elementary tree into γ. Because substitution takes place at the frontier of γ, $depth(v, t)$ must be the same as $depth(v, \gamma)$. More generally, the depth of a lexical node in an elementary tree γ is the same in all trees derived starting from γ. Because the maximal depth of a lexical node in an elementary

tree of G_1' is 2, we deduce that every tree generated by G_1' contains a lexical node with depth at most 2. In contrast, all lexical nodes in the tree t_1 have depth 3. Therefore the tree t_1 is not generated by G_1'.

3.2 Main Part

We now generalize this argument to arbitrary candidate grammars. For this, we are interested in the following class $\mathcal{G}_1$ of all (reduced) TSGs that derive a subset of the trees derived by G_1:

$$\mathcal{G}_1 \;=\; \{\, G \mid G \text{ is a TSG},\; T(G) \subseteq T(G_1) \,\}$$

For a grammar $G \in \mathcal{G}_1$, we define the **d-index** of G as the maximum in $\mathbb{N} \cup \{\infty\}$ of the minimal depths of a-labeled nodes in trees derived by G:

$$\textit{d-index}(G) \;=\; \max_{t \in T(G)} \; \min_{v \in \textit{Nodes}(a,t)} \textit{depth}(v,t)$$

Note that, for two grammars $G, G' \in \mathcal{G}_1$, $T(G) = T(G')$ implies that G and G' have the same d-index. This means that two grammars in $\mathcal{G}_1$ with different d-indices cannot generate the same tree language. Then Theorem 1 directly follows from the two statements in the next lemma.

Lemma 1
The grammar G_1 has infinite d-index. Every lexicalized grammar in $\mathcal{G}_1$ has finite d-index.

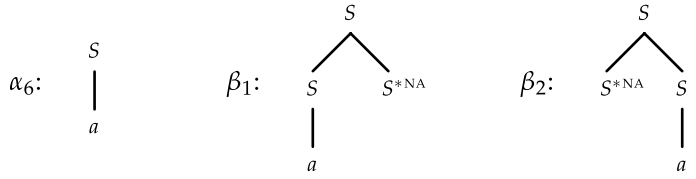
Proof
The first statement is easy to verify: Using longer and longer derivations, the minimal depth of an a-labeled node in the corresponding tree can be pushed beyond any bound.

To prove the second statement, let G be a lexicalized grammar in $\mathcal{G}_1$, and let $t \in T(G)$. The tree t is derived starting from some initial tree; call this tree α. Because G is lexicalized, at least one of the a-labeled nodes in $\textit{Nodes}(a,t)$ is contributed by α. Let v_a be any such node in t, and let u_a be the node of α that corresponds to v_a. Remember that the only tree combination operation allowed in a TSG derivation is substitution. Because substitution can only take place at the frontier of a derived tree, we must conclude that $\textit{depth}(v_a, t) = \textit{depth}(u_a, \alpha)$. There are only finitely many initial trees in G, therefore $\textit{depth}(u_a, \alpha)$ must be upper bounded by some constant depending only on G, and the same must hold for $\textit{depth}(v_a, t)$. Lastly, because t has been arbitrarily chosen in $T(G)$, we must conclude that $\textit{d-index}(G)$ is finite. ■

3.3 Lexicalization of Tree Substitution Grammars

What we have just seen is that lexicalized TSGs are unable to derive the tree structures generated by the grammar G_1 in Figure 2. This is essentially because tree substitution cannot stretch the depth of a lexical node in an elementary tree. In contrast, tree adjunction allows the insertion of additional structure at internal nodes of elementary trees,

and enables TAGs to provide a strong lexicalization of TSGs. For example, the following TAG G_1'' lexicalizes G_1.



Note that this grammar looks almost like G_1', except that adjunction now is allowed at internal nodes, and substitution nodes have become foot nodes. The following derivation tree witnesses that the tree t_1 can be derived in G_1''. We write 0 to denote the root node of an elementary tree, and 1 to denote its leftmost child.

$$\alpha_6 \rightarrow_0 \beta_1 \rightarrow_0 \beta_1 \rightarrow_1 \beta_1$$

Schabes (1990) provides a general procedure for constructing a lexicalized TAG for a given context-free grammar.

4. Tree-Adjoining Grammars Are Not Closed Under Strong Lexicalization

In this section we develop the proof of the main result of this article.

Theorem 2
Tree-adjoining grammars are not closed under strong lexicalization.

4.1 Proof Idea

The basic idea underlying the proof of Theorem 2 is essentially the same as the one used in the proof of Theorem 1 in Section 3. Some discussion of this issue is in order at this point. In the previous section, we have seen that adjunction, in contrast to substitution, allows the insertion of additional structure at internal nodes of elementary trees, and enables TAGs to provide a strong lexicalization of TSGs. One might now be tempted to believe that, because the depth-based argument that we used in the proof of Lemma 1 can no longer be applied to TAGs, they might be closed under strong lexicalization.

There is a perspective under which adjunction quite closely resembles substitution, however. Let us first look at substitution as an operation on the *yield* of the derived tree. Under this view, substitution is essentially context-free rewriting: It replaces a nonterminal symbol in the yield of a derived tree with a new string consisting of terminals and nonterminals, representing the yield of the tree that is substituted. Under the same perspective, adjunction is more powerful than tree substitution, as is well known. But just as substitution can be seen as context-free rewriting on tree yields, adjunction can be seen as context-free rewriting on the *paths* of trees: It replaces a nonterminal symbol in some path of a derived tree with a string representing the **spine** of the tree that is adjoined—the unique path from the root node of the tree to the foot node.

This observation gives us the following idea for how to lift the proof of Theorem 1 to TAGs. We will specify a TAG G_2 such that the paths of the derived trees of G_2 encode in a string form the derived trees of the counterexample grammar G_1. This encoding is exemplified in Figure 3. Each internal node of a derived tree of G_1 is represented in

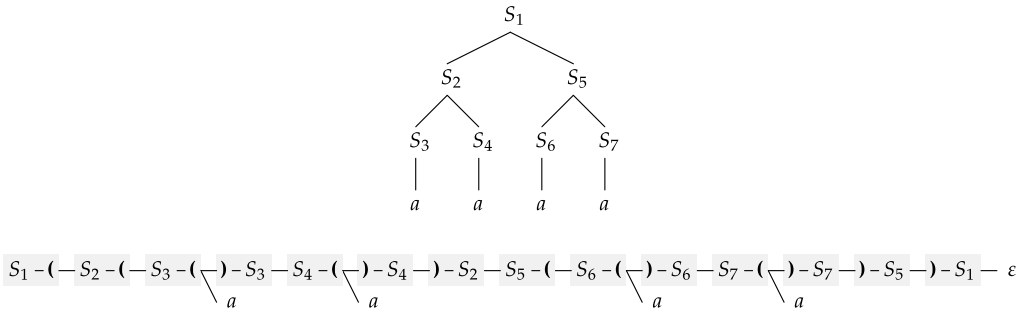


Figure 3
A derived tree of G_1, and the corresponding encoding, drawn from left to right. Every internal node of the original tree is represented by a pair of matching brackets $[S\,(,)\,S]$. The correspondence is indicated by the numerical subscripts.

the spine of the corresponding derived tree of G_2 as a pair of matching brackets. By our encoding, any TAG generating trees from $T(G_2)$ will have to exploit adjunction at nodes in the spine of its elementary trees, and will therefore be subject to essentially the same restrictions as the grammar G_1 which used substitution at nodes in the yield. This will allow us to lift our argument from Lemma 1. The only difference is that instead of working with the actual depth of a lexical node in a tree $t \in T(G_2)$, we will now need to work with the depth of the node in the encoded tree. As will be explained later, this measure can be recovered as the excess of left parentheses over right parentheses in the spine above the lexical node.

4.2 Preliminaries

As already mentioned, our proof of Theorem 2 follows the same structure as our proof of Theorem 1. As our counterexample grammar, we use the grammar G_2 given in Figure 4; this grammar generates the encodings of the derived trees of G_1 that we discussed previously. Note that the left parenthesis symbol '(' and the right parenthesis symbol ')' are nonterminal symbols. As with the grammar G_1 before, it is not difficult to see that G_2 is finitely ambiguous and that $\varepsilon \notin L(G_2)$.

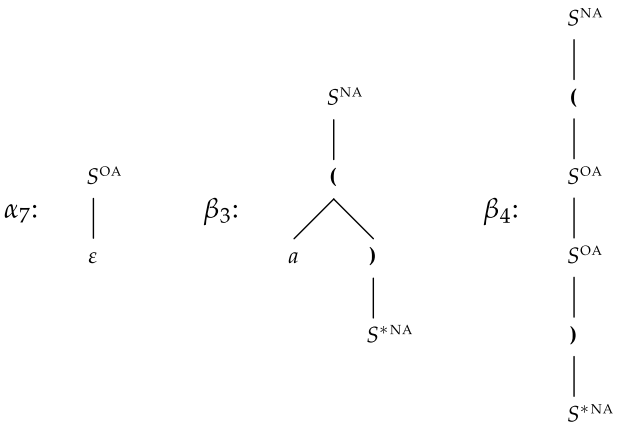


Figure 4
The counterexample TAG G_2.

Grammar G_2 derives trees that we call **right spinal**: Each node in such a tree has at most two children, and the left child of every node with two children is always a leaf node. The path from the root node of a right spinal tree t to the rightmost leaf of t is called **spine**. To save some space, in the following we write right spinal trees horizontally and from left to right, as already done in Figure 3. Thus the grammar G_2 can alternatively be written as follows:

For a node u in a right spinal tree derived by G_2, we define

$$c(u) = \begin{cases} +1 & \text{if } u \text{ is labeled with } \mathbf{(} \\ 0 & \text{if } u \text{ is labeled with } S \text{ or } a \\ -1 & \text{if } u \text{ is labeled with } \mathbf{)} \end{cases}$$

We exploit this function to compute the **excess** of left parentheses over right parentheses in a sequence of nodes, and write:

$$excess(\langle u_1, \ldots, u_n \rangle) = \sum_{i=1}^{n} c(u_i)$$

Let t be some right spinal tree in $T(G_2)$, and let v be some node in t. Assume that $\langle u_1, \ldots, u_n = v \rangle$ is the top–down sequence of all the nodes in the path from t's root u_1 to v. We write $excess(v, t)$ as a shorthand notation for $excess(\langle u_1, \ldots, u_n \rangle)$. If $\langle u_1, \ldots, u_n \rangle$ is the top–down sequence of all the nodes in the spine of t, we also write $excess(t)$ as a short hand notation for $excess(\langle u_1, \ldots, u_n \rangle)$.

It is easy to prove by induction that, for each tree $t \in T(G_2)$, the excess of the sequence of nodes in the spine of t is always zero. Thus, we omit the proof of the following statement.

Lemma 2
Every derived tree $t \in T(G_2)$ is a right spinal tree, and $excess(t) = 0$.

In order to get a better understanding of the construction used in the following proofs, it is useful at this point to come back to our discussion of the relation between that construction and the construction presented in Section 3. We observe that for each tree t_1 generated by G_1 there is a tree $t_2 \in T(G_2)$ such that the sequence of labels in t_2's spine encodes t_1, following the scheme exemplified in Figure 3. Using such encoding, we can establish a bijection between the a-labeled nodes in the frontier of t_1 and the a-labeled nodes in the frontier of t_2. Furthermore, if v_1 in t_1 and v_2 in t_2 are two nodes related by such a correspondence, then it is not difficult to see that $depth(v_1, t_1) = excess(v_2, t_2)$.

4.3 Intuition

Before we give the actual proof of Theorem 2, let us attempt to get some intuition about why our counterexample grammar G_2 cannot be strongly lexicalized by some

other TAG. One might think that the following TAG G_2' is a lexicalized version of G_2:

α_8 : S^{NA} – ($\underset{a}{\_}$) – S^{NA} — ε

β_5 : S^{NA} – ($\underset{a}{\_}$) – S^{*NA}

β_6 : S^{NA} – (— S^{NA} – ($\underset{a}{\_}$) – S^{NA} — S^{OA} —) – S^{*NA}

β_7 : S^{NA} – (— S^{OA} — S^{NA} – ($\underset{a}{\_}$) – S^{NA} —) – S^{*NA}

This grammar is obtained from G_2 by taking the lexicalized tree β_3 (repeated here as β_5), as well as all trees that can be obtained by adjoining β_3 into some non-lexicalized elementary tree. G_2' does not generate *all* trees generated by G_2, however. The following tree t_2 for example is not generated by G_2':

S –(— S –(— S –($\underset{a}{\_}$)– S — S –($\underset{a}{\_}$)– S —)– S — S –(— S –($\underset{a}{\_}$)– S — S –($\underset{a}{\_}$)– S —)– S —)– S — ε

Note that this tree is the encoded version of the counterexample tree t_1 from the previous section (cf. Figure 3).

To see that t_2 is not generated by G_2', we reason as follows. Consider a lexical node u in an elementary tree γ of G_2', and let t be a tree obtained by adjoining some elementary tree into γ. Although this adjunction increases the depth of u, it does not increase its excess, as it adds a balanced sequence of parentheses into the spine of γ. More generally, the excess of a lexical node in an elementary γ is constant in all trees derived starting from γ. From this we conclude that every tree generated by G_2' contains a lexical node with excess at most 2; this is the maximal excess of a lexical node in an elementary tree of G_2'. In contrast, all lexical nodes in the tree t_2 have excess 3. This shows that t_2 is not generated by G_2'.

4.4 Main Part

In what follows, we consider the class $\mathcal{G}_2$ of (reduced) TAGs that generate subsets of the trees derived by G_2:

$$\mathcal{G}_2 \;=\; \{\, G \mid G \text{ is a TAG},\ T(G) \subseteq T(G_2) \,\}$$

For a grammar $G \in \mathcal{G}_2$, we define the **e-index** of G as the maximum in $\mathbb{N} \cup \{\infty\}$ of the minimal excess of a-labeled nodes in trees derived by G:

$$\textit{e-index}(G) \;=\; \max_{t \in T(G)} \; \min_{v \in \textit{Nodes}(a,t)} \textit{excess}(v,t)$$

As we will see, the notion of e-index plays exactly the same role as the notion of d-index in Section 3.

There is one last obstacle that we need to overcome. For TSGs we noted (in the proof of Lemma 1) that the minimal depth of lexical nodes in a derived tree t is bounded by the minimal depth of lexical nodes in the elementary tree γ from which t was derived. For the TAGs in $\mathcal{G}_2$, the situation is not quite as simple, as an adjunction of an auxiliary tree β into an elementary tree γ might affect the excess of a lexical node of γ. It turns out, however, that this potential variation in the excess of a lexical node of γ is bounded by a grammar-specific constant. This observation is expressed in the following lemma. It is the correspondent of Lemma 4 in Knuth's paper on parenthesis languages (Knuth 1967), and is proved in essentially the same way. Recall that a derivation tree d is of type γ, γ some elementary tree, if d is derived starting from γ.

Lemma 3
Let $G \in \mathcal{G}_2$. For each elementary tree γ of G, there exists a number $e(\gamma)$ such that, for every saturated derivation tree d of type γ, $\mathit{excess}(t(d)) = e(\gamma)$.

Proof
Because γ is not useless, we can find at least one sentential and saturated derivation tree of G that contains an occurrence of γ. Let d be any such derivation tree, and let v be any node of d labeled with γ. Let d_1 be the subtree of d rooted at v. Observe that $t(d_1)$ must be a spinal tree. We then let $e(\gamma) = \mathit{excess}(t(d_1))$.

If d_1 is the only derivation tree of type γ available in G, then we are done. Otherwise, let $d_2 \neq d_1$ be some derivation tree of type γ occurring within some other sentential and saturated derivation tree of G. We can replace d_1 with d_2 in d at v to obtain a new sentential and saturated derivation tree $d' \neq d$. Every derived tree in $T(G)$ must be a right spinal tree: This follows from the assumption that $G \in \mathcal{G}_2$ and from Lemma 2. We can then write

$$\mathit{excess}(t(d')) \;=\; \mathit{excess}(t(d)) - \mathit{excess}(t(d_1)) + \mathit{excess}(t(d_2))$$

Because $\mathit{excess}(t(d)) = 0$ and $\mathit{excess}(t(d')) = 0$ (by Lemma 2), we conclude that

$$\mathit{excess}(t(d_2)) \;=\; \mathit{excess}(t(d_1)) \;=\; e(\gamma)$$

■

Using Lemma 3, we can now prove the following result.

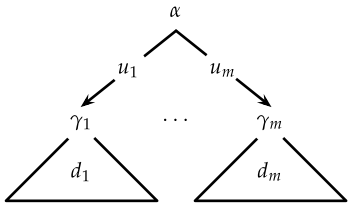
Lemma 4
The grammar G_2 has infinite e-index. Every lexicalized grammar in $\mathcal{G}_2$ has finite e-index.

Proof
As in the case of Lemma 1, the first statement is easy to verify and we omit its proof. To prove the second statement, let $G \in \mathcal{G}_2$. Let Γ be the set of all elementary trees of G, and let s be the maximal number of nodes in an elementary tree in Γ. We show that

$$\mathit{e\text{-}index}(G) \;\leq\; k\,, \qquad \text{where} \qquad k \;=\; s + s \cdot \max_{\gamma \in \Gamma} |e(\gamma)|$$

Note that k is a constant that only depends on G.

Let d be a sentential and saturated derivation tree of G. It has the following shape:



Here α is some initial tree, $m \geq 0$, each u_i is a node of α at which a tree combination operation takes place, each γ_i is an elementary tree, and each d_i is a derivation tree of type γ_i that is a subtree of d. According to this derivation tree, the derived tree $t(d)$ is obtained by substituting or adjoining the derived trees $t(d_i)$ at the respective nodes u_i of α.

Because G is lexicalized, at least one a-labeled node on the frontier of $t(d)$ is contributed by α. Let v_a be any such node, and let u_a be the node of α that corresponds to v_a. The quantity $excess(v_a, t(d))$, representing the excess of the path in $t(d)$ from its root to the node v_a, can be computed as follows. Let $\langle u'_1, \ldots, u'_n = u_a \rangle$ be the top–down sequence of nodes in the path from the root node of α to u_a. For each i with $1 \leq i \leq n$ we define

$$\hat{c}(u'_i) \;=\; \begin{cases} excess(t(d_j)) & \text{if } u'_i = u_j \text{ for some } 1 \leq j \leq m \\ c(u'_i) & \text{otherwise} \end{cases}$$

Because $G \in \mathcal{G}_2$ and because $t(d)$ is a right spinal tree (Lemma 2), we can write

$$excess(v_a, t(d)) \;=\; \sum_{i=1}^{n} \hat{c}(u'_i)$$

By Lemma 3, we have $excess(t(d_j)) = e(\gamma_j)$, for each j with $1 \leq j \leq m$. We can then write

$$excess(v_a, t(d)) \;\leq\; n + \sum_{i=1}^{m} |e(\gamma_i)| \;\leq\; s + s \cdot \max_{\gamma \in \Gamma} |e(\gamma)| \;=\; k$$

Thus, every derived tree t in $T(G)$ contains at least one node v_a in its frontier such that $excess(v_a, t) \leq k$. Therefore, $e\text{-}index(G) \leq k$. ■

Two grammars in $\mathcal{G}_2$ that have a different e-index cannot generate the same tree language, thus we have concluded the proof of Theorem 2.

5. Tree Insertion Grammars Are Not Closed Under Strong Lexicalization

As mentioned earlier Schabes and Waters (1995) introduce a restricted variant of TAG called TIG. The essential restriction in that formalism is the absence of **wrapping trees**, which are trees derived starting from auxiliary trees with overt lexical material on both sides of the foot node. Schabes and Waters (1995, Section 5.1.4) conjecture that the class of all TIGs is closed under strong lexicalization.

It is easy to see that the counterexample grammar G_2 that we gave in Figure 4 does not derive wrapping trees; this means that G_2 actually is a TIG. Using the proof of Section 4, we then obtain the following result.

Theorem 3
Tree insertion grammars are not closed under strong lexicalization.

In fact, we have even proved the stronger result that the class of TAGs does not lexicalize the class of TIGs.

6. Conclusion

We have shown that, in contrast to what has been claimed in the literature, TAGs are not closed under strong lexicalization: The restriction to lexicalized TAGs involves a loss in strong generative capacity.

In this article we have only considered TAGs with Null Adjunction and Obligatory Adjunction constraints. A third kind of adjunction constraint that has been used in the literature is Selective Adjunction, where a set of trees is provided that may be adjoined at some node. It is not difficult to see that the proofs of Lemma 3, Lemma 4, and Theorem 3 still hold if Selective Adjunction constraints are used.

Our result triggers a number of follow-up questions. First, are TAGs closed under *weak* lexicalization, defined in Section 1? We know that, in the case of CFGs, this question can be answered affirmatively, because Greibach normal form is a special case of lexicalized form, and for every CFG there is a weakly equivalent grammar in Greibach normal form. But to our knowledge, no comparable result exists for TAG. Second, if TAGs cannot strongly lexicalize themselves, what would a grammar formalism look like that is capable of providing strong lexicalization for TAGs?

Acknowledgments
We are grateful to Aravind Joshi for discussion on previous versions of this article and for helping us in shaping the text in the Introduction of the current version. We also acknowledge three anonymous reviewers for their helpful comments.

References

Joshi, Aravind K., Leon S. Levy, and Masako Takahashi. 1975. Tree Adjunct Grammars. *Journal of Computer and System Sciences*, 10(2):136–163.

Joshi, Aravind K. and Yves Schabes. 1992. Tree-adjoining grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, *Tree Automata and Languages*. North-Holland, Amsterdam, pages 409–431.

Joshi, Aravind K. and Yves Schabes. 1997. Tree-adjoining grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3. Springer, Berlin, pages 69–123.

Knuth, Donald E. 1967. A characterization of parenthesis languages. *Information and Control*, 11(3):269–289.

Schabes, Yves. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, University of Pennsylvania, Philadelphia.

Schabes, Yves and Richard C. Waters. 1995. Tree insertion grammar: A cubic-time parsable formalism that lexicalizes context-free grammars without changing the trees produced. *Computational Linguistics*, 21(4):479–513.