The weak generative capacity of linear tree-adjoining grammars

David Chiang[*]
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292, USA
chiang@isi.edu

1 Introduction

Linear tree-adjoining grammars (TAGs), by analogy with linear context-free grammars, are tree-adjoining grammars in which at most one symbol in each elementary tree can be rewritten (adjoined or substituted at). Uemura et al. (1999), calling these grammars *simple linear TAGs* (SL-TAGs), show that they generate a class of languages incommensurate with the context-free languages, and can be recognized in $\mathcal{O}(n^4)$ time.

Working within the application domain of modeling of RNA secondary structures, they find that SL-TAGs are too restrictive—they can model RNA pseudoknots but because they cannot generate all the context-free languages, they cannot model even some very simple RNA secondary structures. Therefore they propose a more powerful version of linear TAGs, *extended simple linear TAGs* (ESL-TAGs), which generate a class of languages that include the context-free languages and can be recognized in $\mathcal{O}(n^5)$ time.

Satta and Schuler (1998), working within the application domain of natural language syntax, define another restriction on TAG which is also recognizable in $\mathcal{O}(n^5)$ time. Despite being less powerful than full TAG, it is still able to generate languages like the copy language $\{ww\}$ and Dutch cross-serial dependencies (Joshi, 1985). Kato et al. (2004) conjecture that this restricted TAG is in fact equivalent to ESL-TAG.

In this paper we prove their conjecture, and also prove that adding substitution to ESL-TAG does not increase its weak generative capacity, whereas adding substitution to SL-TAG makes it weakly equivalent to ESL-TAG. Thus these four formalisms converge to the same weak-equivalence class, the intuition being that the "hardest" operation in TAG, namely, adjunction of a wrapping auxiliary tree in the middle of the spine of another wrapping auxiliary tree, is subjected to the linearity constraint, but most other operations are unrestricted.[1] Kato et al. (2004) show that these formalisms are more powerful than SL-TAG or general CFG or their union and conjecture, on the other hand, that they are less powerful than TAG. We prove this conjecture as well.

2 Definitions

We assume a standard definition of TAG, with or without substitution, in which adjunction is not allowed at foot nodes, and other nodes can have no-adjunction (NA) constraints, obligatory-adjunction (OA), or selective-adjunction constraints. We use the symbols η, η_1, η_2, etc. to range over nodes of elementary trees or derived trees, although sometimes we use the label of a node to refer to the node itself. The *spine* of an auxiliary tree is the path from its root node to its foot node, inclusive. The *subtree* of a node η is the set of all nodes dominated by η, including η itself. The *segment* of a tree from η_1 to η_2 (where η_1 dominates η_2) is the set of all nodes in the subtree of η_1 but not in the subtree of η_2. A segment can be *excised*, which means removing the nodes of the segment and making η_2 replace η_1 as the child of its parent.

We also assume a standard definition of TAG derivation trees. We use the symbols h, h_1, h_2, etc. to range over nodes of derivation trees. The *sub-*

[*] This research was primarily carried out while the author was at the University of Pennsylvania.

[1] Adjunction at root and foot nodes is another operation that by itself will not take a formalism beyond context-free power, a fact which is exploited in Rogers' regular-form TAG (Rogers, 1994). But allowing this in a linear TAG would circumvent the linearity constraint.

Proceedings of the 8th International Workshop on Tree Adjoining Grammar and Related Formalisms, pages 25–32,
Sydney, July 2006.

derivation of h is the subtree of h in the derivation tree. When we cut up derivations into subderivations or segments and recombine them, the edge labels (indicating addresses of adjunctions and substitutions) stay with the node above, not the node below.

Now we define various versions of linear TAG.

Definition 1. A *right (left) auxiliary tree* is one in which the leftmost (rightmost) frontier node is the foot node, and the spine contains only the root and foot nodes. A *wrapping auxiliary tree* is one which is neither a left or a right auxiliary tree.

Definition 2. We say that a node of an elementary tree is *active* if adjunction is allowed to occur at it, and that a node is *w-active* if adjunction of a wrapping auxiliary tree is allowed to occur at it.

Definition 3. A *Satta-Schuler linear tree-adjoining grammar* (SSL-TAG) is a TAG with substitution in which:

1. In the spine of each wrapping auxiliary tree, there is at most one w-active node.

2. In the spine of each left or right auxiliary tree, there are no w-active nodes, nor are there any other adjoining constraints.

Definition 4. A *simple linear tree-adjoining grammar* (SL-TAG), with or without substitution, is a TAG, with or without substitution, respectively, in which every initial tree has exactly one active node, and every auxiliary tree has exactly one active node on its spine and no active nodes elsewhere.

Definition 5. An *extended simple linear tree-adjoining grammar* (ESL-TAG), with or without substitution, is a TAG, with or without substitution, respectively, in which every initial tree has exactly one active node, and every auxiliary tree has exactly one active node on its spine and at most one active node elsewhere.

3 Properties

We now review several old results and prove a few new results relating the weak generative capacity of these formalisms to one another and to (linear) CFG and TAG. These results are summarized in Figure 1.

3.1 Previous results

Proposition 1 (Uemura et al. 1999).

$$\textit{Linear CFL} \subsetneq \textit{SL-TAL}$$

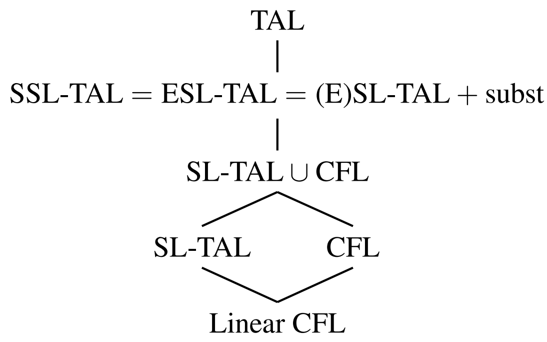


Figure 1: Summary of results: an edge indicates that the higher formalism has strictly greater weak generative capacity than the lower.

Proposition 2 (Uemura et al. 1999).

$$\textit{CFL} \subsetneq \textit{ESL-TAL}$$

Proposition 3 (Kato et al. 2004).

$$\textit{CFL} \cup \textit{SL-TAL} \subsetneq \textit{ESL-TAL}$$

Proposition 4 (Satta and Schuler 1998; Uemura et al. 1999). *SSL-TAG and ESL-TAG can be parsed in $\mathcal{O}(n^5)$ time.*

3.2 Weak equivalence

Proposition 5. *The following formalisms are weakly equivalent:*

(i) ESL-TAG

(ii) SL-TAG with substitution

(iii) ESL-TAG with substitution

(iv) SSL-TAG

Proof. We prove this by proving four inclusions.

$\mathcal{L}(\text{ESL-TAG}) \subseteq \mathcal{L}(\text{ESL-TAG} + \text{substitution})$: Trivial.

$\mathcal{L}(\text{ESL-TAG} + \text{substitution}) \subseteq \mathcal{L}(\text{SSL-TAG})$: Trivial.

$\mathcal{L}(\text{SSL-TAG}) \subseteq \mathcal{L}(\text{SL-TAG} + \text{substitution})$: We deal first with the left and right auxiliary trees, and then with off-spine adjunction.

First, we eliminate the left and right auxiliary trees. Since these only insert material to the left or right of a node, just as in tree-insertion grammars (TIGs), we may apply the conversion from TIGs to tree-substitution grammars (Schabes and Waters, 1995), used in the proof of the context-freeness of

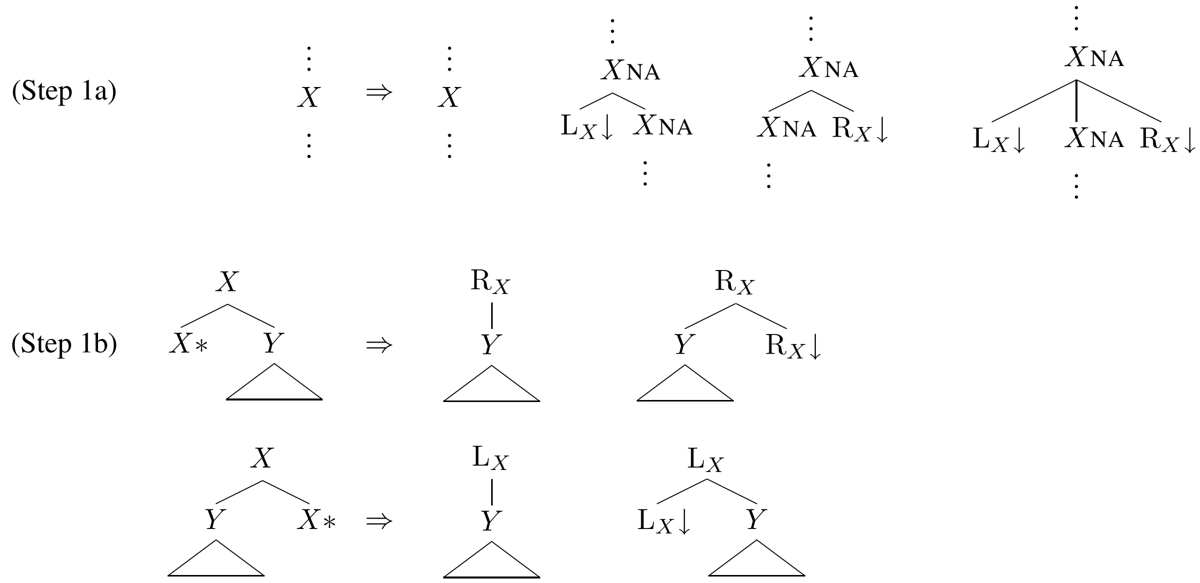


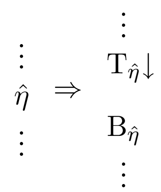
Figure 2: Elimination of left/right auxiliary trees.

TIG.[2] (Step 1a) For each active node X that is not the root of a left or right auxiliary tree, we create four copies of the containing elementary tree with X altered in the following ways: first, leave X unchanged; then, add a copy of X above it, making both nodes no-adjunction nodes, and add a new left sister substitution node labeled L_X or a new right sister substitution node labeled R_X, or both. See Figure 2. (Step 1b) For each β that was originally a left (right) auxiliary tree with root/foot label X, relabel the root node as L_X (R_X) and delete the foot node, and create two copies of the containing elementary tree, one unchanged, and one with a new left (right) sister substitution node. See Figure 2. When the modified β substitutes at one of the new children of an η, the substitution clearly results in the same string that would have resulted from adjoining the original β to η.

This construction might appear incorrect in two ways. First, the new grammar has trees with both an L_X and an R_X node corresponding to the same original node, which would correspond to adjunction of two auxiliary trees β_L and β_R at the same node X in the original grammar. But this new derivation generates a string that was generable in the original grammar, namely by adjoining β_L at X, then adjoining β_R at the root of β_L, which is allowed because the definition of SSL-TAG prohibits adjunction constraints at the root of β_L.

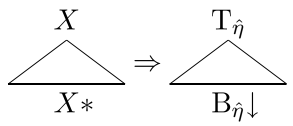
Thus the first apparent problem is really the solution to the second problem: in the original grammar, a left auxiliary tree β_L could adjoin at the root of a right auxiliary tree β_R, which in turn adjoined at a node η, whereas in the new grammar, β_R does not have an L_X substitution node to allow this possibility. But the same string can be generated by substituting both trees under η in the new grammar. In the case of a whole chain of adjunctions of left/right auxiliary trees at the root of left/right auxiliary trees, we can generate the same string by rearranging the chain into a chain of left auxiliary trees and a chain of right auxiliary trees (which is allowed because adjunction constraints are prohibited at all the roots), and substituting both at η.

(Step 2) Next, we eliminate the case of a wrapping auxiliary tree β that can adjoin at an off-spine node η. (Step 2a) For each active off-spine node η, we relabel η with a unique identifier $\hat{\eta}$ and split the containing elementary tree at η:

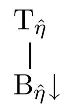


[2]This corresponds to Steps 1–4 of that proof (Schabes and Waters, 1995, p. 486). Since that proof uses a more relaxed definition of left and right auxiliary trees, it is probable that SSL-TAG could also be relaxed in the same way.

(Step 2b) After step 2a has been completed for all nodes η, we revisit each η, and for every wrapping β that could adjoin at η, create a copy of β with root relabeled to $\mathrm{T}_{\hat{\eta}}$ and foot relabeled to $\mathrm{B}_{\hat{\eta}}$.

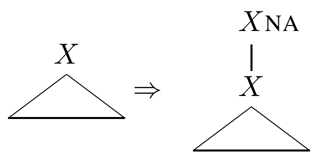


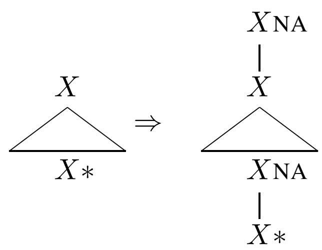
Then the original β is discarded. Substituting one of these copies of β at a $\mathrm{T}_{\hat{\eta}}$ node and then substituting a $\mathrm{B}_{\hat{\eta}}$ tree at the former foot node has the same effect as adjoining β at η. Finally, unless η had an obligatory-adjunction constraint, simulate the lack of adjunction at η by adding the initial tree



$\mathcal{L}$(SL-TAG + substitution) $\subseteq$ $\mathcal{L}$(ESL-TAG): This construction is related to Lang's normal form which ensures binary-branching derivation trees (Lang, 1994), but guarantees that one adjunction site is on the spine and one is off the spine.

(Step 0a) Ensure that the elementary trees are binary-branching. (Step 0b) Add a new root and foot node to every elementary tree:

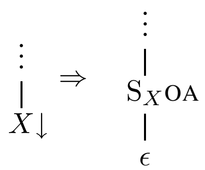




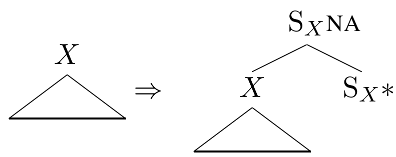
(Step 1) We transform the grammar so that no auxiliary tree has more than one substitution node. For any auxiliary tree with spine longer than four nodes, we apply the following transformation: target either the active node or its parent, and call it Y. Let Z_1 be the child that dominates the foot node; let V_1 be a fresh nonterminal symbol and insert V_1 nodes above Y and below Z_1, and excise the segment between the two V nodes, leaving behind an active obligatory-adjunction node. If Y has another child, call it Z_2; let V_2 be a fresh nonterminal symbol and insert a V_2 node above Z_2, and break off the subtree rooted in V_2, leaving behind a substitution node. See Figure 3. This transformation reduces the spine of the auxiliary tree by one node, and creates two new trees that satisfy the desired form. We repeat this until the entire grammar is in the desired form.

(Step 2) Next, we transform the grammar so that no initial tree has more than one substitution node, while maintaining the form acquired in step 1. For any initial tree with height greater than three nodes, we apply the same transformation as in step 1, except that Y is the child of the root node, Z_1 is its left child, and Z_2 is its other child if it exists and is not already a substitution node. See Figure 3. This transformation replaces an initial tree with at most two shorter initial trees, and one auxiliary tree in the desired form. Again we repeat this until the entire grammar is in the desired form.

(Step 3) Finally, we convert each substitution node into an adjunction node (Schabes, 1990). For each substitution node η, let X be the label of η. Relabel η to S_X with obligatory adjunction and place an empty terminal beneath η.



For each initial tree with root label X, convert it into an auxiliary tree by adding a new root node labeled S_X whose children are the old root node and a new foot node.



□

3.3 Relation to tree-adjoining languages

Our second result, also conjectured by Kato et al., is that the weak equivalence class established above is a proper subset of TAL.

Proposition 6. *The language*

$$L = \{\mathrm{a}_1^r \mathrm{b}_1^p \mathrm{b}_2^p \mathrm{c}_1^q \mathrm{c}_2^q \mathrm{a}_2^r \mathrm{a}_3^r \mathrm{c}_3^q \mathrm{c}_4^q \mathrm{b}_3^p \mathrm{b}_4^p \mathrm{a}_4^r\}$$

is in TAL but not ESL-TAL.

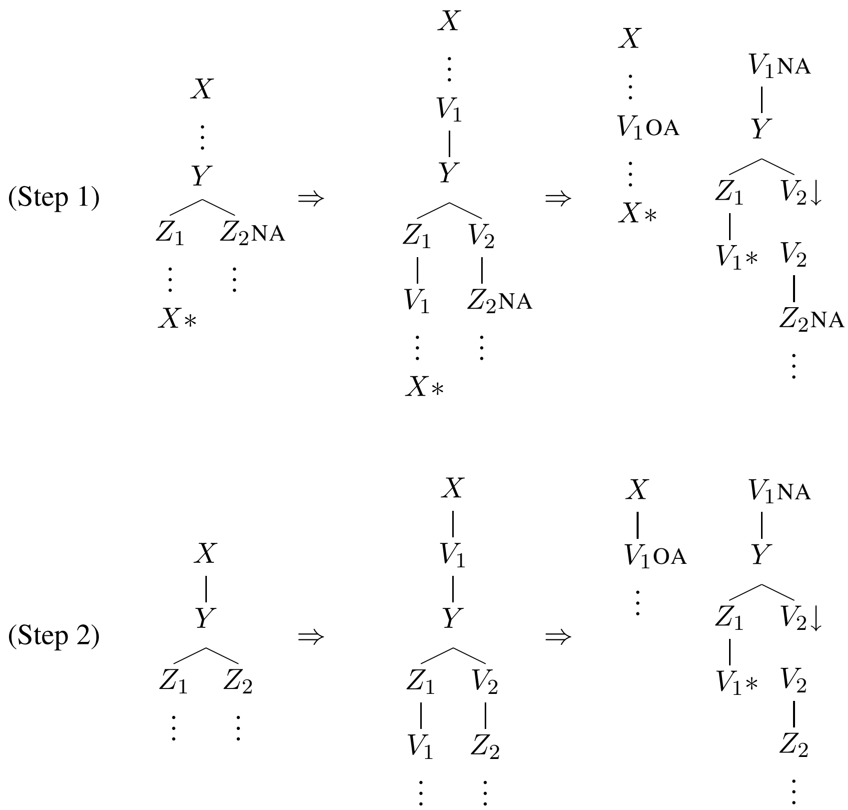
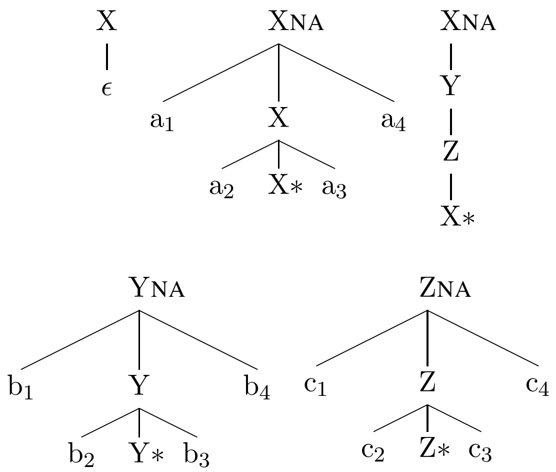


Figure 3: Separation of substitution nodes. Some adjunction constraints are omitted to avoid clutter.

Proof ($L \in$ TAL). The language is generated by the following TAG:



□

Before proceeding to the other half of the proof, we define a few useful notions. A *marked string* (as in Ogden's Lemma) over an alphabet Σ is a string over $\Sigma \times \{0, 1\}$, where a symbol $\langle \sigma, 1 \rangle$ is *marked* and a symbol $\langle \sigma, 0 \rangle$ is not. Marked strings over Σ can be projected into Σ^* in the obvious way and we will talk about marked strings and their projections interchangeably.

A *decomposed string* over Σ is a sequence of strings over Σ, which can be projected into Σ^* by concatenating their members in order, and again we will talk about decomposed strings and their projections interchangeably. In particular, we will often simply write a decomposed string $\langle w_1, \ldots, w_n \rangle$ as $w_1 \cdots w_n$. Moreover, we may use the symbol w_i to refer to the occurrence of the ith member of the decomposition in w; for example, if w is a marked string, we may say that a symbol in w_i is marked, or if w is generated by a TAG derivation, we may say that w_i is generated by some set of nodes in the derivation tree.

The second half of the proof requires a double-decker pumping lemma.

Condition 1 (cf. Vijay-Shanker (1987), Theorem 4.7). Given a language L and a decomposed string $x_1 z x_2 \in L$ with some symbols in z marked, there exists a decomposition of z into $u_1 v_1 w_1 v_2 u_2 v_3 w_2 v_4 u_3$ such that one of the v_i contains a mark, and L contains, for all $k \geq 1$,

$$x_1(u_1 v_1^k w_1 v_2^k u_2 v_3^k w_2 v_4^k u_3) x_2$$

**Condition 2 (cf. Uemura et al. (1999), Lemma

1). Given a language L and a decomposed string $x_1 z_1 z_2 x_2 z_3 z_4 x_3 \in L$ with some symbols in one of the z_i marked, there exist decompositions of the z_i into $u_i v_i w_i$ such that one of the v_i contains a mark, and L contains, for all $k \geq 1$,

$$x_1(u_1 v_1^k w_1)(u_2 v_2^k w_2)x_2(u_3 v_3^k w_3)(u_4 v_4^k w_4)x_3$$

Lemma 7. *If L is an ESL-TAL, then there exists a constant n such that for any $z \in L$ with n symbols marked, Condition 1 holds of $\epsilon \cdot z \cdot \epsilon$. Moreover, it holds such that the w_1 and w_2 it provides can be further decomposed into $z_1 z_2$ and $z_3 z_4$, respectively, such that for any marking of n symbols of any of the z_j, either Condition 1 holds of $z = x_1 z_j x_2$ (where x_1 and x_2 are the surrounding context of z_j) or Condition 2 holds of $z = x_1 z_1 z_2 x_2 z_3 z_4 x_3$ (where x_1, x_2, and x_3 are the surrounding context of $z_1 z_2$ and $z_3 z_4$).*

Proof. Since L is an ESL-TAL, it is generated by some ESL-TAG G. Let k be the number of elementary trees in G and t be the maximum number of terminal symbols in any elementary tree of G. Then set $n = 2^{k+1}t$.

The first invocation of Condition 1 is the TAG version of Ogden's lemma (Hopcroft and Ullman, 1979). To show that it holds, we need to find a path P in the derivation tree of z that has a cycle that generates at least one marked symbol. Define a *branch point* to be a node h in the derivation tree such that the marked nodes generated by the subderivation of h are not all generated by the subderivation of a single child of h. We seek a P that has at least $k + 1$ branch points. Start by adding the root of the derivation tree to P. Thereafter let h be the last node in P. If h is a leaf, stop; otherwise, add to P the child of h whose subderivation generates the most marked symbols. Note that if a branch point in P generates m marked symbols, the next branch point generates at least $\frac{m-t}{2}$. Our choice of n then guarantees that P has at least $k+1$ branch points, at least two of which must correspond to the same auxiliary tree. Call these nodes h_1 and h_2.

These two nodes divide the derivation up into three phases: first, the derivation segment from the root to h_1, which we call α (because it can be thought of as the derived initial tree it generates); then the segment from h_1 to h_2, which we call β_1 (because it can be thought of as the derived auxiliary tree it generates); then subderivation of h_2, which we call β_2. Note that we can form new valid derivations of G by repeating β_2: that is, in terms of derivation trees, stacking α on top of one or more copies of β_1, on top of β_2—or in terms of derived trees, repeatedly adjoining β_1 into α and then adjoining β_2.

If β_2 adjoins into the spine of β_1, then let $\langle u_1, u_2, u_3 \rangle$ be the parts of z generated by α, $\langle v_1, v_2, v_3, v_4 \rangle$ the parts generated by β_1, and $\langle w_1, w_2 \rangle$ the parts generated by β_2 (see Figure 4a). Then these new derivations generate the strings $u_1 v_1^k w_1 v_2^k u_2 v_3^k w_2 v_4^k u_3$.

But if β_2 adjoins at a node to the left of the spine of β_1, then let $\langle u_1, v_{42}, u_3 \rangle$ be the parts of the z generated by α, $\langle v_1, u_2, v_{41}, v_{43} \rangle$ the parts generated by β_1, and $\langle w_1, w_2 \rangle$ the parts generated by β_2 (see Figure 4b). Then let $v_2 = v_3 = \epsilon$ and $v_4 = v_{41} v_{42} v_{43}$; the new derivations will generate the strings $u_1 v_1^k w_1 v_2^k u_2 v_3^k w_2 v_4^k u_3$. The case where β_2 adjoins to the right of the spine.

Now we focus attention on β_2. Let S be the longest path of the derivation of β_2 containing the root of the derivation and auxiliary trees adjoined at spine nodes. This S is unique because each spine can only have one active node. Let h_3 be the last node in S, which divides the derivation of β_2 into two phases: the segment from the root to h_3, which we call β_{21}, and the subderivation of h_3, which we call β_{22}. This gives a decomposition $\langle w_1, w_2 \rangle = \langle z_1 z_{21} z_{22}, z_{31} z_{32} z_4 \rangle$, where β_{22} generates z_{21} and z_{32} (see Figure 5). Note that the derivation nodes in S are the only ones that can generate symbols in z_1, z_{22}, z_{31}, and z_4 at once; the other derivation nodes only generate symbols in a single z_i. We let $z_2 = z_{21} z_{22}$ and $z_3 = z_{31} z_{32}$ and hand off the decomposition $\langle w_1, w_2 \rangle = \langle z_1 z_2, z_3 z_4 \rangle$ to our adversary, who may choose a z_j and mark n symbols in it.

Then we recapitulate the reasoning above to get a path P' starting from the root of the derivation of β_2 and containing at least $k + 1$ branch points, two of which correspond to the same auxiliary tree. Call these nodes h_4 and h_5 and the segment between them β_3, and let $\langle v_1, v_2, v_3, v_4 \rangle$ now stand for the parts of $\langle w_1, w_2 \rangle$ generated by β_3. Once again, we are going to repeat β_3 to generate new derivations, pumping copies of the v_i into $\langle w_1, w_2 \rangle$. But the location of the v_i depends on h_5: if h_5 is in S, then the v_i will appear inside each of the z_i, satisfying Condition 2. Otherwise, they will all appear inside z_j. □

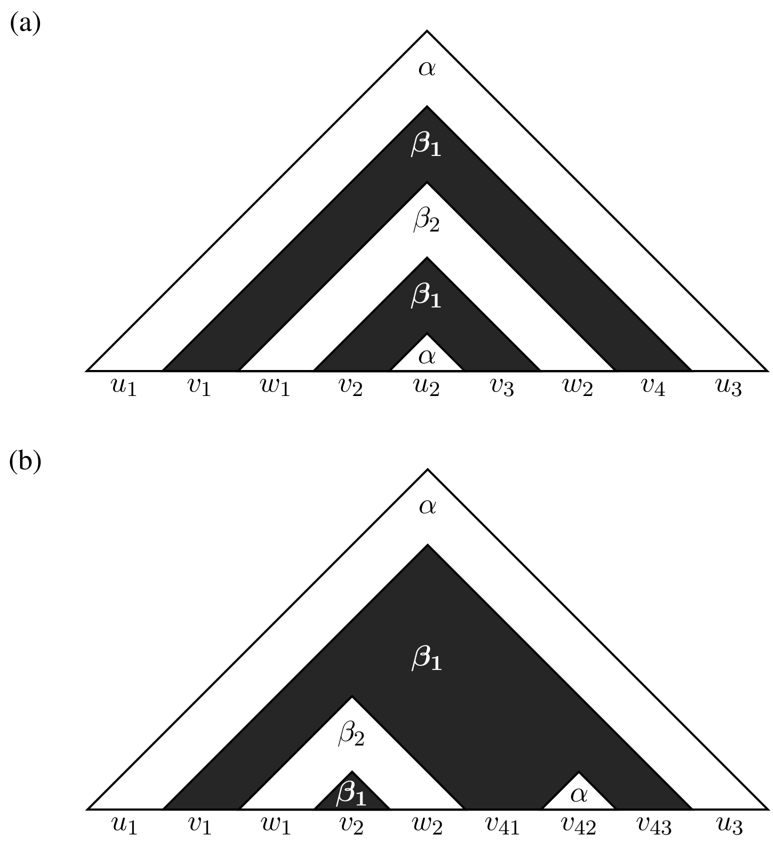


Figure 4: Anatomy of derived tree in proof of Lemma 7.

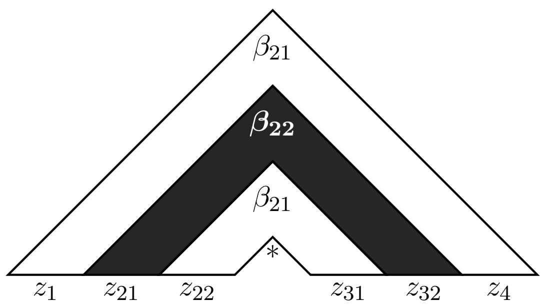


Figure 5: Anatomy of β_2 in proof of Lemma 7.

Finally we complete the proof of Proposition 6.

Proof of Proposition 6 ($L \notin$ ESL-TAL). Suppose L is an ESL-TAL. Let z be the string obtained by setting $p = q = r = n$, and mark the a_1s. Then Lemma 7 must hold. The first invocation of Condition 1 must give a w_1 of the form $a_1^* b_1^n b_2^n c_1^n c_2^n a_2^*$ and a w_2 of the form $a_3^* c_3^n c_4^n b_3^n b_4^n a_4^*$. Lemma 7 must further decompose w_1 into $z_1 z_2$. Obviously, either z_1 contains all the b_js or z_2 contains all the c_js. Supposing the former, we can obtain a contradiction by marking the b_1s: Condition 2 is impossible because it would give unequal numbers of b_1s and b_2s; Condition 1 is impossible because it would give unequal numbers of b_1s and b_3s. On the other hand, if z_2 contains all the c_js, we mark the c_1s, and both Conditions are again rendered impossible. □

4 Conclusion

The weak equivalence of the previously proposed ESL-TAG and SSL-TAG, along with the fact that SL-TAG with substitution and ESL-TAG with substitution belong to the same class, suggests that they represent a useful compromise between CFGs and TAGs. In the two-dimensional language hierarchy of Rambow and Satta (1999), where the two dimensions are *rank* (how many substructures does a rule combine) and *fanout* (how many discontinuous spans of the input does a substructure cover), CFGs comprise the fanout-1 grammars and TAGs are a subset of the the fanout-2 grammars; both have arbitrary rank, whereas linear CFGs and linear TAGs are rank-1. The grammars discussed here are mixed: a rule can combine one fanout-2 substructure and an arbitrary number of fanout-1 substructures. A related example would be a version of synchronous CFG that allows only one pair of linked nonterminals and any number of unlinked nonterminals, which could be bitext-parsed in $\mathcal{O}(n^5)$ time, whereas inversion transduction grammar (Wu, 1997) takes $\mathcal{O}(n^6)$. It may be of interest to make a more general exploration of other formalisms that are mixed in this sense.

Acknowledgements

Thanks to Hiroyuki Seki for discussions that led to this paper, and to Anoop Sarkar, Giorgio Satta, and William Schuler. This research was partially supported by NSF grant ITR EIA-02-05456. *S. D. G.*

References

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

Aravind K. Joshi. 1985. Tree adjoining grammars: How much context-sensitivity is necessary for assigning structural descriptions? In David Dowty, Lauri Karttunen, and Arnold Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, Cambridge.

Yuki Kato, Hiroyuki Seki, and Tadao Kasami. 2004. Subclasses of tree adjoining grammar for RNA secondary structure. In *Proc. Seventh International Workshop on TAG and Related Formalisms (TAG+)*, pages 48–55.

Bernard Lang. 1994. Recognition can be harder than parsing. *Computational Intelligence*, 10(4):484–494. Special Issue on Tree Adjoining Grammars.

Owen Rambow and Giorgio Satta. 1999. Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, 223:87–120.

James Rogers. 1994. Capturing CFLs with tree adjoining grammars. In *Proc. 32nd Annual Meeting of the ACL*, pages 155–162.

Giorgio Satta and William Schuler. 1998. Restrictions on tree adjoining languages. In *Proc. COLING-ACL*, pages 1176–1182.

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

Yves Schabes. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, University of Pennsylvania. Available as technical report MS-CIS-90-48.

Yasuo Uemura, Aki Hasegawa, Satoshi Kobayashi, and Takashi Yokomori. 1999. Tree adjoining grammars for RNA structure prediction. *Theoretical Computer Science*, 210:277–303.

K Vijayashanker. 1987. *A study of tree adjoining grammars*. Ph.D. thesis, University of Pennsylvania. Available as technical report MS-CIS-88-03.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23:377–404.