Lecture 4

Tree-Adjoining Grammars and Related Formalisms

Last modified 2016/05/30

During a conversation among some participants of the Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms (Tübingen, Germany, June 2008), Aravind Joshi mentioned that there is a passage in Huybregts and Riemsdijk's 1979–1980 interviews with Chomsky where Chomsky remarks that it would be nice if there were a grammar formalism that extends the power of context-free grammars, but does so in a limited way, without achieving the full power of context-sensitive grammars. The following remark (Chomsky 2004, pages 42–43) is the closest that I have been able to find:

> . . . Certainly context-free grammars represent some of the properties of languages. This seems to me what one would expect from applied mathematics, to see if you can find systems that capture some of the properties of the complex system that you are working with, and to ask whether those systems have any intrinsic mathematical interest, and whether they are worth studying in abstraction. And that has happened at exactly one level, the level of context-free grammar. At any other level it has not happened. The systems that capture other properties of language, for example transformational grammar, hold no interest for mathematics. But I do not think that that is a necessary truth. It could turn out that there would be richer or more appropriate mathematical ideas that would capture other, maybe deeper properties of language than context-free grammars do. In that case you have another branch of applied mathematics which might have linguistic consequences. That could be exciting.

It is not clear whether Chomsky views tree-adjoining grammars and other so-called *mildly context-sensitive* grammar formalisms as capturing "deeper properties" of

natural language, but it is certainly the case that they represent another level where grammar formalisms motivated by linguistics have some intrinsic mathematical interest.

Tree-adjoining grammars

Recall the cross-serial dependencies in Swiss German, which we looked at in Lecture 1:

(4.1) a. wil mer de maa em chind lönd hälffe schwüme

because we the man-ACC the child-DAT let help swim

b. wil mer em maa s chind hälfed laa schwüme

because we the man-DAT the child-ACC helped let swim

(The above example is from Huybregts 1984.) In this construction, sentences of the form

$$\text{NP NP}_1 \ldots \text{NP}_n \text{ V}_1 \ldots \text{V}_n \text{ V}, \tag{4.2}$$

where the V_i are verbs like "let", "help", etc., have the following properties:

- (4.2) is grammatical if the case of NP_i matches the type of V_i ($i = 1, \ldots, n$), and
- (4.2) is grammatical only if there is a permutation π of $\{1, \ldots, n\}$ such that the case of NP_i matches the type of $\text{V}_{\pi(i)}$.

It's not clear what exactly the necessary and sufficient conditions for grammaticality are, but the above properties are sufficient to establish that the set of grammatical sentences of Swiss German is not context-free (Huybregts 1984, Shieber 1985), as we saw in Lecture 1.

Partly motivated by facts like this, many grammar formalisms have been proposed that are more powerful than context-free grammars but yet share some desirable features of CFGs. The *tree-adjoining grammar* (TAG) (Joshi et al. 1975; see Joshi and Schabes 1997 and Abeillé and Rambow 2000 for overviews) is one of the first such formalisms. It is also our first example of a *tree grammar*: a TAG primarily generates a set of trees (*tree language*), and secondarily a set of strings (*string language*) through the yield function.

Roughly, a TAG consists of two finite sets of trees, the set $\mathscr{I}$ of *initial trees* and the set $\mathscr{A}$ of *auxiliary trees*. Initial and auxiliary trees are called *elementary trees*.

More complex trees are derived from elementary trees by repeated applications of the operations of *substitution* and *adjunction*:

- Substitution attaches a tree derived from an initial tree to a substitution node of a derived tree.
- Adjunction "inserts" an auxiliary tree into an internal node of a derived tree.

A substitution node is a leaf node labeled by a nonterminal marked with $\downarrow$. In an initial tree, each leaf node either is a substitution node or is labeled by a terminal (or ε). In an auxiliary tree whose root node is labeled by A, exactly one leaf node is labeled with A^* and is called the *foot node*; the other leaf nodes must either be substitution nodes or have terminal (or ε) labels. The operation of adjunction is illustrated in Figure 4.1.

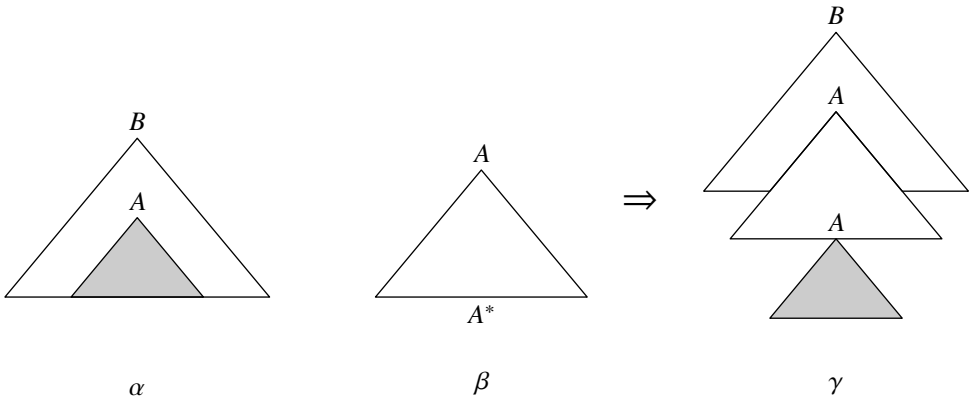
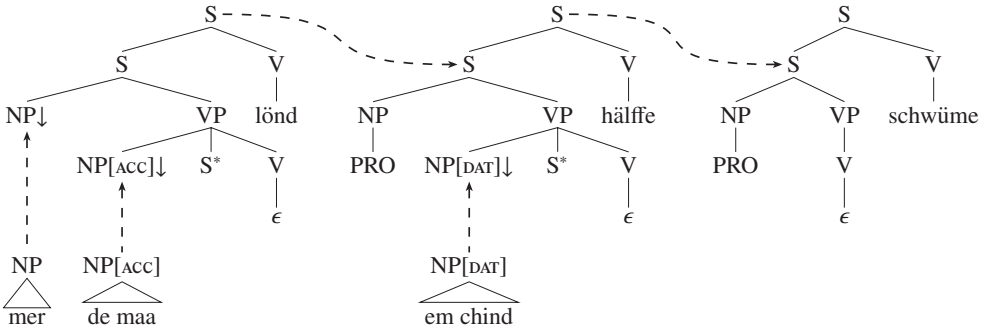


Figure 4.1: Adjunction of tree β to tree α, resulting in tree γ.

The following illustrates how the Swiss-German cross-serial construction may be treated in a tree-adjoining grammar:



A formal definition of TAGs usually omits the operation of substitution, since the expressive power and computational complexity of the formalism is the same

with or without substitution. It is also important to constrain which auxiliary tree can be adjoined at a given node, so each node labeled by a nonterminal will carry an *adjunction constraint*. Adjunction constraints come in two varieties:

- *Selective Adjunction* ($SA(\mathscr{C})$): only members of a set $\mathscr{C}$ of auxiliary trees may be adjoined at the given node. The adjunction is not mandatory.
- *Obligatory Adjunction* ($OA(\mathscr{C})$): a member of a set $\mathscr{C}$ of auxiliary trees must be adjoined at the given node.

A selective adjunction constraint of the form $SA(\varnothing)$ is called a *null adjunction* constraint and is written NA. If $\mathscr{C}$ is the set of auxiliary trees whose root node is labeled by a nonterminal B, a selective adjunction constraint $SA(\mathscr{C})$ on a node labeled by B is omitted, and an obligatory adjunction constraint $OA(\mathscr{C})$ on a node labeled by B is simply written OA.

Formally, we represent an adjunction constraint using a set $\mathscr{C}$ of positive integers $\leq m$, where m is the number of auxiliary trees in the grammar, assuming that the auxiliary trees are arranged in a list. (Informally, we take $\mathscr{C}$ to be a subset of the set of auxiliary trees.) A *tree-adjoining grammar* is a 4-tuple $(N, \Sigma, \mathscr{I}, \mathscr{A})$, where

1. N is a finite set of symbols, called nonterminals;
2. Σ is a finite set of symbols, called terminals, disjoint from N;
3. $\mathscr{I}$ is a finite set of trees, called *initial trees*, characterized as follows:
   - each leaf node is labeled by a terminal or ε;
   - each internal node is labeled by a nonterminal and an adjunction constraint;
4. $\mathscr{A}$ is a finite (ordered) set of trees, called *auxiliary trees*, characterized as follows:
   - each leaf node except one, called the *foot node*, is labeled by a terminal or ε;
   - each internal node and the foot node are labeled by a nonterminal and an adjunction constraint.

The foot node of an auxiliary tree is customarily distinguished by an asterisk, but in the absence of substitution, this carries no information. Note that the above definition does not impose the requirements that the root node of an initial tree be labeled by a distinguished nonterminal (start symbol) and that the root and

foot nodes of an auxiliary tree be labeled by the same nonterminal, since these requirements have little formal consequence.[1]

To define the operation of adjunction formally, we need a precise way of referring to nodes in a tree. We use a system of using elements of $\mathbb{N}_+^*$ (finite sequences of positive integers) as node addresses.[2] In writing a sequence of positive integers, we use the dot "." to separate consecutive integers, as in 136.187.45.174, which represents the sequence (136, 187, 45, 174). The dot is also used to denote concatenation of sequences; e.g., if $p = 1.4.13.2$ and $q = 25.1$, then $p.q = 1.4.13.2.25.1$. We write $p \leqslant q$ to mean that p is a prefix of q, i.e., that there is some r such that $p.r = q$.

A *tree domain* is a non-empty subset D of $\mathbb{N}_+^*$ satisfying the following properties:

- $p.1 \in D$ implies $p \in D$, and
- $p.i \in D$ implies $p.j \in D$ for all $j \in \mathbb{N}_+$ such that $j < i$.

Let Γ be an alphabet. A *labeled ordered tree* (or simply a *tree*) (over Γ) is a function α from a tree domain D_α to Γ.[3] The elements of D_α are the *nodes* of α. The *root* of α is ε (the empty string), which is an element of every tree domain, and if $p.i \in D_\alpha$, p is the unique *parent* of $p.i$, and every $p.j \in D_\alpha$ is a *child* of p. A node $p \in D_\alpha$ such that $p.1 \notin D_\alpha$ is a *leaf*.

Let α and β be trees. Let p be a node of α and q be a leaf of β. Then adjoin(α, p, β, q) is defined to be the tree γ such that

$$D_\gamma = \{\, r \in D_\alpha \mid p \not\leqslant r \,\} \cup \{\, p.s \mid s \in D_\beta \,\} \cup \{\, p.q.r \mid p.r \in D_\alpha \,\},$$

$$\gamma(t) = \begin{cases} \alpha(t) & \text{if } t \in D_\alpha \text{ and } p \not\leqslant t, \\ \beta(s) & \text{if } t = p.s \text{ and } s \in D_\beta, \\ \alpha(p.r) & \text{if } t = p.q.r,\ r \neq \varepsilon, \text{ and } p.r \in D_\alpha. \end{cases}$$

Note that adjoin$(\alpha, p, \beta, p) = \beta(\varepsilon)$ and adjoin$(\alpha, p, \beta, q)(p.q) = \beta(q)$, so the label $\alpha(p)$ is lost.

Suppose that α is a derived tree of a TAG $G = (N, \Sigma, \mathscr{I}, \mathscr{A})$, and consider adjoining the k-th auxiliary tree β at node p of α. This adjunction is legitimate if for some $B \in N$ and some set $\mathscr{C}$ of positive integers, the label at node p of α is $(B, SA(\mathscr{C}))$ or $(B, OA(\mathscr{C}))$ and $k \in \mathscr{C}$. In this case, we write $\alpha[p \leftarrow \beta]$ for adjoin(α, p, β, q), where q is the foot node of β.

[1] Rogers (2003) called TAGs without the second requirement *non-strict* TAGs.

[2] These addresses are sometimes called *Gorn addresses* (Gorn 1967). Knuth (1997) calls this system *Dewey decimal notation*.

[3] We have used Greek letters $\alpha, \beta, \gamma, \ldots$ as variables for strings of terminal and nonterminal symbols. In the context of TAGs, it is customary to denote elementary and derived trees with these letters.

A TAG $G = (N, \Sigma, \mathscr{I}, \mathscr{A})$ derives a tree γ if γ can be derived from an initial tree by repeatedly adjoining auxiliary trees. Formally, $\alpha \Rightarrow_G \alpha'$ if there is an auxiliary tree $\beta \in \mathscr{A}$ such that for some node p of α, adjunction of β at node p of α is legitimate and α' is the result of this adjunction, or in symbols, $\alpha' = \alpha[p \leftarrow \beta]$. We write $\Rightarrow^*_G$ for the reflexive transitive closure of $\Rightarrow_G$. A tree γ is a *derived tree* of G if for some initial tree $\alpha \in \mathscr{I}$, we have $\alpha \Rightarrow^*_G \gamma$, and it is a *derived adjunction tree* if for some auxiliary tree $\alpha \in A$, we have $\alpha \Rightarrow^*_G \gamma$.

The *tree language* generated by G is

$$L(G) = \{\, \gamma \mid \gamma \text{ is a derived tree of } G \text{ that has no node labeled with an OA constraint} \,\}.$$

The *string language* generated by G is defined as follows:

$$\mathbf{y}L(G) = \{\, \mathbf{y}(\gamma) \mid \gamma \in L(G) \,\},$$

where $\mathbf{y}(\gamma)$ is the *yield* of the tree γ, i.e., the concatenation of the labels of the leaf nodes of γ, from left to right.[4] A language L is a *tree-adjoining language* (TAL) if $L = \mathbf{y}L(G)$ for some TAG G.

When we introduced CFGs, we gave two different ways of characterizing the generated strings, namely, derivations and parse trees. Analogously, TAGs have an alternative way of characterizing the derived trees, namely *derivation trees*. Derivation trees of TAGs are like parse trees of CFGs—they correspond to equivalence classes of derivations.

To define the set of derivation trees of a TAG $G = (N, \Sigma, \mathscr{I}, \mathscr{A})$, we define a *multi-start CFG* $\mathrm{cf}(G) = (N', \Sigma', P', \mathscr{S}')$ corresponding to G, which is like a CFG except that it has a finite set $\mathscr{S}' \subseteq N' \cup \Sigma'$ of start symbols. A *derivation tree* of G is simply a parse tree of $\mathrm{cf}(G)$ (whose root label need not be in $\mathscr{S}'$). A *complete derivation tree* of G is a parse tree of $\mathrm{cf}(G)$ whose root label is in $\mathscr{S}'$. If $\alpha \in \mathscr{I} \cup \mathscr{A}$, we write $OA(\alpha)$ and $NA(\alpha)$ to denote the set of *OA* nodes of α and the set of *NA* nodes of α, respectively.

- N' consists of nonterminals of the form

  $$[\alpha p_1 \ldots p_n],$$

  where $\alpha \in \mathscr{I} \cup \mathscr{A}$, $n \geq 1$, and $p_1, \ldots, p_n$ are nodes of α with nonterminal labels (listed in preorder) such that

  - $OA(\alpha) \subseteq \{p_1, \ldots, p_n\}$, and
  - $NA(\alpha) \cap \{p_1, \ldots, p_n\} = \varnothing$.

- $\Sigma' = \{\, [\alpha] \in \mathscr{I} \cup \mathscr{A} \mid OA(\alpha) = \varnothing \,\}$.

[4] The label ε is treated as the empty string, as in the case of parse trees of context-free grammars.

- P' consists of productions of the form

$$[\alpha p_1 \dots p_n] \to [\beta_1 q_{1,1} \dots q_{1,m_1}] \dots [\beta_n q_{n,1} \dots q_{n,m_n}],$$

  where $[\alpha p_1 \dots p_n] \in N'$ and for $k = 1, \dots, n$, it holds that $[\beta_k q_{k,1} \dots q_{k,m_k}] \in N' \cup \Sigma'$ ($m_k \geq 0$) and the adjunction of β_k at node p_k of α is legitimate.
- $\mathscr{S}' = \{\, [\alpha p_1 \dots p_n] \in N' \mid \alpha \in \mathscr{S} \,\} \cup \{\, [\alpha] \in \Sigma' \mid \alpha \in \mathscr{S} \,\}$.

Define a function dtree that maps derivation trees to derived (adjunction) trees inductively as follows:

$$\text{dtree}([\alpha]) = \alpha, \quad \text{for } [\alpha] \in \Sigma',$$

$$\text{dtree}\left(\begin{array}{c} [\alpha p_1 \dots p_n] \\ \tau_1 \quad \cdots \quad \tau_n \end{array}\right) = \alpha[p_1 \leftarrow \text{dtree}(\tau_1), \dots, p_n \leftarrow \text{dtree}(\tau_n)],$$

where the right-hand side of the second equation means iterated adjunction[5]

$$\alpha[p_n \leftarrow \text{dtree}(\tau_n)] \dots [p_1 \leftarrow \text{dtree}(\tau_1)].$$

Then we can show

$$L(G) = \{\, \text{dtree}(\tau) \mid \tau \text{ is a complete derivation tree of } G \,\}.$$

The class of tree-adjoining languages properly includes the class of context-free languages. The following languages are TALs:

- COPY = $\{\, ww \mid w \in \{\mathsf{a}, \mathsf{b}\}^* \,\}$.
- COUNT-3 = $\{\, \mathsf{a}^n\mathsf{b}^n\mathsf{c}^n \mid n \geq 0 \,\}$ and COUNT-4 = $\{\, \mathsf{a}^n\mathsf{b}^n\mathsf{c}^n\mathsf{d}^n \mid n \geq 0 \,\}$.

The language COUNT-4 is generated by a TAG $G = (\{S, A\}, \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}, \{\alpha\}, \{\beta\})$, where:

$$\alpha = \begin{array}{c} S_{NA} \\ | \\ A \\ | \\ \varepsilon \end{array} \qquad \beta = \begin{array}{ccc} & A_{NA} & \\ \mathsf{a} & & \mathsf{d} \\ & \mathsf{b} \quad A^*_{NA} \quad \mathsf{c} & \end{array}$$

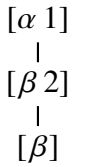
The associated multi-start context-free grammar is $\text{cf}(G) = (N', \Sigma', P', \mathscr{S}')$, where

$$N' = \{[\alpha\ 1], [\beta\ 2]\},$$

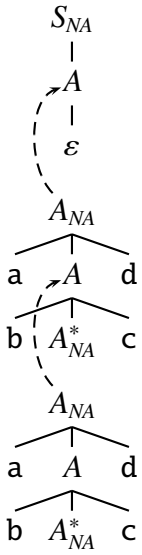
[5]Recall that the nodes $p_1, \dots, p_n$ are listed in preorder. This ensures that the target of adjunction of $\text{dtree}(\tau_i)$ keeps the original address in $\alpha[p_n \leftarrow \text{dtree}(\tau_n)] \dots [p_{i+1} \leftarrow \text{dtree}(\tau_{i+1})]$.

$$\Sigma' = \{[\alpha], [\beta]\},$$
$$P' = \{[\alpha\ 1] \to [\beta\ 2],\ [\alpha\ 1] \to [\beta],\ [\beta\ 2] \to [\beta\ 2],\ [\beta\ 2] \to [\beta]\},$$
$$\mathscr{S}' = \{[\alpha\ 1], [\alpha]\}.$$

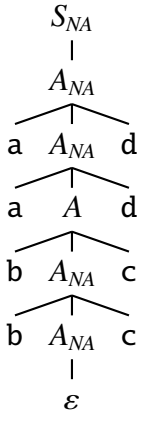
The following tree τ is a complete derivation tree of G:



A more graphical way of depicting this derivation tree would be:



The corresponding derived tree dtree(τ) is



Exercise 4.1. Write a TAG that generates COPY.

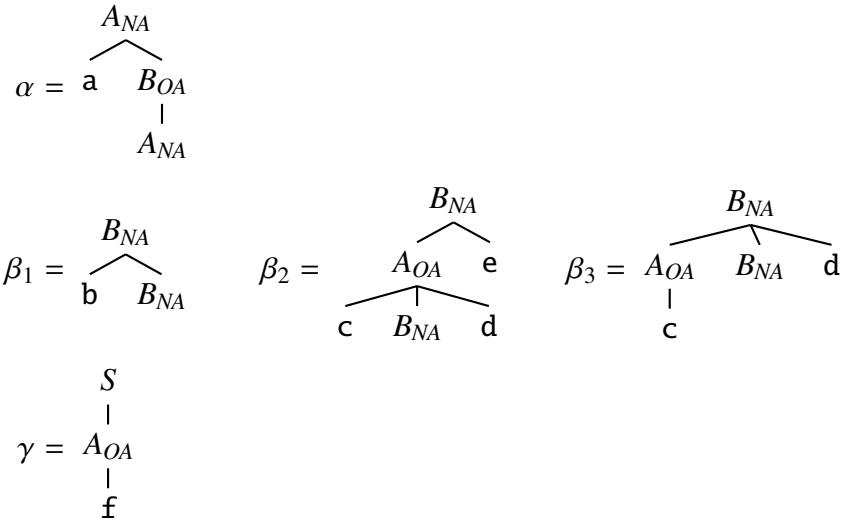
Exercise 4.2. Write a TAG that generates $\{ ww^R \mid w \in D_1^* \}$, where D_1^* is the *Dyck language* over $\{a, \bar{a}\}$, defined by the following context-free grammar:

$$S \to TS \mid \varepsilon$$
$$T \to aS\bar{a}$$

A weak pumping lemma for tree-adjoining languages

The derivation trees of a TAG constitute a *local set*, which is a special kind of *regular tree language*. There is a pumping lemma for the regular tree languages, which says (roughly) that every tree in a regular tree language whose height exceeds a certain bound (depending on the language) has a part that can be pumped any number of times with the resulting tree still belonging to the language. Using this fact we can obtain a weak form of pumping lemma for the string languages generated by tree-adjoining grammars.

Consider a TAG $G = (N, \Sigma, \mathscr{I}, \mathscr{A})$, where $N = \{S, A, B\}$, $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}\}$, $\mathscr{I} = \{\gamma\}$, $\mathscr{A} = \{\alpha, \beta_1, \beta_2, \beta_3\}$.



The associated multi-start context-free grammar is $\mathrm{cf}(G) = (N', \Sigma', P', \mathscr{S}')$, where

$$N' = \{[\alpha\, 2], [\beta_2\, 1], [\beta_3\, 1], [\gamma\, 1]\},$$
$$\Sigma' = \{[\beta_1]\},$$
$$\mathscr{S}' = \{[\gamma\, 2]\}$$

and P' consists of the following productions:

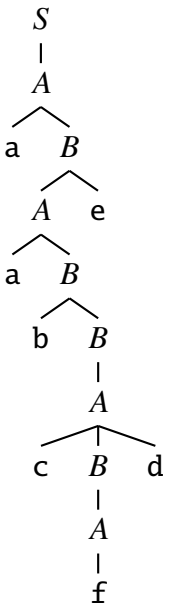
$$
\begin{aligned}
[\gamma\ 1] &\rightarrow [\alpha\ 2] \\
[\alpha\ 2] &\rightarrow [\beta_1] \\
[\alpha\ 2] &\rightarrow [\beta_2\ 1] \\
[\alpha\ 2] &\rightarrow [\beta_3\ 1] \\
[\beta_2\ 1] &\rightarrow [\alpha\ 2] \\
[\beta_3\ 1] &\rightarrow [\alpha\ 2]
\end{aligned}
$$

Consider the following derivation tree:

$$
\tau = \begin{array}{c} [\gamma\ 1] \\ | \\ [\alpha\ 2] \\ | \\ [\beta_2\ 1] \\ | \\ [\alpha\ 2] \\ | \\ [\beta_1] \end{array}
$$

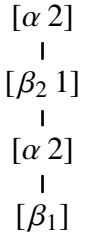
We have

$$\text{dtree}(\tau) = \gamma[1 \leftarrow \alpha[2 \leftarrow \beta_2[1 \leftarrow \alpha[2 \leftarrow \beta_1]]]] =$$

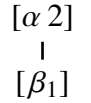


$\mathbf{y}(\text{dtree}(\tau))$ = `aabcfde`.

The subtree

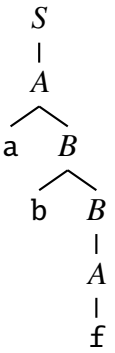


of τ can be replaced with



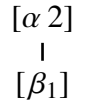
resulting in a shorter derivation tree:

$$\text{dtree}(\tau_0) = \gamma[1 \leftarrow \alpha[2 \leftarrow \beta_1]] =$$

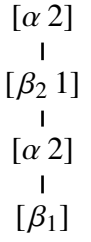


$$\mathbf{y}(\text{dtree}(\tau_0)) = \texttt{abf}.$$

Conversely, the subtree



of τ can be replaced with



resulting in a longer derivation tree

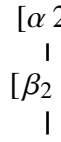
$$\tau_2 = \begin{array}{c} [\gamma\ 1] \\ | \\ [\alpha\ 2] \\ | \\ [\beta_2\ 1] \\ | \\ [\alpha\ 2] \\ | \\ [\beta_2\ 1] \\ | \\ [\alpha\ 2] \\ | \\ [\beta_1] \end{array}$$

$$\text{dtree}(\tau_2) = \gamma[1 \leftarrow \alpha[2 \leftarrow \beta_2[1 \leftarrow \alpha[2 \leftarrow \beta_2[1 \leftarrow \alpha[2 \leftarrow \beta_1]]]]]]$$

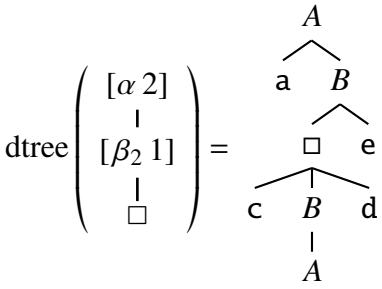
$$= \begin{array}{c} S \\ | \\ A \\ \texttt{a}\quad B \\ \quad A\quad \texttt{e} \\ \texttt{a}\quad B \\ \quad A\quad \texttt{e} \\ \texttt{a}\quad B \\ \quad \texttt{b}\quad B \\ \qquad\ \ | \\ \qquad\ A \\ \quad \texttt{c}\quad B\quad \texttt{d} \\ \qquad\ \ | \\ \qquad\ A \\ \quad \texttt{c}\quad B\quad \texttt{d} \\ \qquad\ \ | \\ \qquad\ A \\ \qquad\ \ | \\ \qquad\ \texttt{f} \end{array}$$

$\mathbf{y}(\text{dtree}(\tau_2)) = \texttt{aaabccfddee}$.

We see that the part



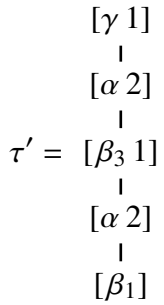
of τ can be repeated $n \geq 0$ times, resulting in a derivation tree τ_n. If we let $\Box$ denote a dummy auxiliary tree with just one node (labeled by $\Box$), then

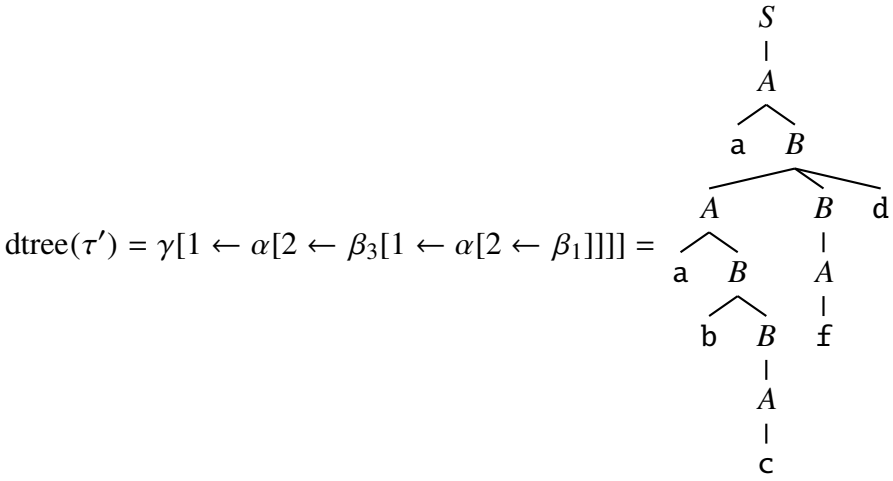


So repeating this part n times contributes substrings $\mathtt{a}^n, \mathtt{c}^n, \mathtt{d}^n, \mathtt{e}^n$. We see

$$\mathbf{y}(\text{dtree}(\tau_n)) = \mathtt{a}^n\mathtt{abc}^n\mathtt{fd}^n\mathtt{e}^n.$$

Consider another derivation tree





$\mathbf{y}(\text{dtree}(\tau')) = \texttt{aabcfd}$.

The subtree

$$\begin{array}{c}[\alpha\ 2]\\ \vert\\ [\beta_3\ 1]\\ \vert\\ [\alpha\ 2]\\ \vert\\ [\beta_1]\end{array}$$

of τ' can be replaced with

$$\begin{array}{c}[\alpha\ 2]\\ \vert\\ [\beta_1]\end{array}$$

resulting in a shorter derivation tree:

$$\tau_0' = \begin{array}{c}[\gamma\ 1]\\ \vert\\ [\alpha\ 2]\\ \vert\\ [\beta_1]\end{array} \quad (= \tau_0)$$

$$\text{dtree}(\tau_0') = \begin{array}{ccc} & S & \\ & \vert & \\ & A & \\ \swarrow & & \searrow \\ \texttt{a} & & B \\ & \swarrow\ \searrow & \\ & \texttt{b} \quad B & \\ & & \vert \\ & & A \\ & & \vert \\ & & \texttt{f} \end{array}$$

$\mathbf{y}(\text{dtree}(\tau_0')) = \texttt{abf}$.

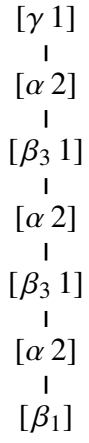
Conversely, the subtree

$$\begin{array}{c}[\alpha\ 2]\\ \vert\\ [\beta_1]\end{array}$$

of τ can be replaced with

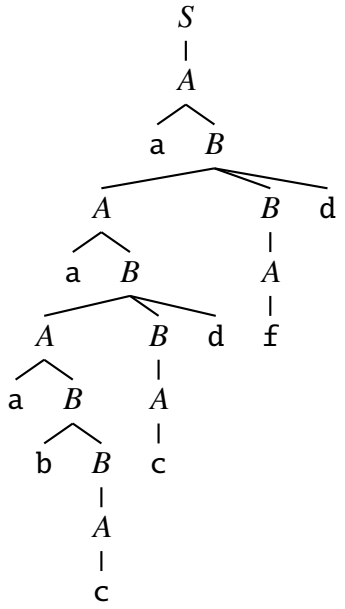
resulting in a longer derivation tree

$\tau_2' =$



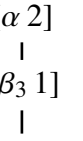
$$\text{dtree}(\tau_2') = \gamma[1 \leftarrow \alpha[2 \leftarrow \beta_3[1 \leftarrow \alpha[2 \leftarrow \beta_3[1 \leftarrow \alpha[2 \leftarrow \beta_1]]]]]]$$

$=$

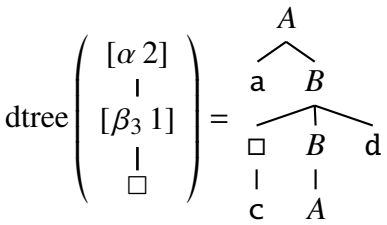


$\mathbf{y}(\text{dtree}(\tau_2'))$ = aaabccdfd.

The part



of τ' can be repeated $n+1$ times, resulting in a derivation tree τ'_{n+1}. Note



Note that unlike in the previous case, the node labeled by □ is not on the spine (i.e., the path from the root node to the foot node) of the derived adjunction tree. It is not difficult to see that repeating this part $n+1$ times contributes substrings $\mathtt{a}^{n+1}$, $\mathtt{c}$, $(\mathtt{cd})^n$, $\mathtt{d}$, and

$$\mathbf{y}(\mathrm{dtree}(\tau'_{n+1})) = \mathtt{a}^{n+1}\mathtt{abc}(\mathtt{cd})^n\mathtt{fd}.$$

Note that one of the pumped strings, $\mathtt{cd}$, is not a substring of the original string $z = \mathbf{y}(\mathrm{dtree}(\tau')) = \mathtt{aabcfd}$. Indeed, it is not possible to find $u_1, v_1, w_1, v_2, u_2, v_3, w_3, v_4, u_3$ such that $z = u_1v_1w_1v_2u_2v_3w_2v_4u_3$, $|v_1v_2v_3v_4| > 0$, and

$$u_1v_1^nw_1v_2^nu_2v_3^nw_2v_4^nu_3 \in \mathbf{y}L(G) \quad \text{for all } n \geq 0.$$

The consideration so far leads to the following weak version of the pumping lemma for tree-adjoining languages (Vijayashanker 1987, Kallmeyer 2010).

Theorem 4.1 (Weak Pumping Lemma for TALs)**.** *Let L be a TAL. Then there is a natural number p such that for every string $z \in L$ with $|z| \geq p$, there are strings $u_1, u_2, u_3, v_1, v_2, v_3, v_4, w_1, w_2$ satisfying the following conditions:*

1. $|v_1v_2v_3v_4| \geq 1$,
2. $|v_1v_2v_3v_4| + |w_1w_2| \leq p$, *and*
3. *one of the following holds:*
   (a) $z = u_1v_1w_1v_2u_2v_3w_2v_4u_3$ *and* $u_1v_1^nw_1v_2^nu_2v_3^nw_2v_4^nu_3 \in L$ *for all* $n \geq 0$*;*
   (b) $z = u_1v_1w_1v_2w_2v_3u_2v_4u_3$ *and* $u_1v_1^{n+1}w_1v_2w_2(v_3v_2v_4)^nv_3u_2v_4u_3 \in L$ *for all* $n \geq 0$*; or*

(c) $z = u_1v_1u_2v_2w_1v_3w_2v_4u_3$ *and* $u_1v_1u_2v_2(v_1v_3v_2)^nw_1v_3w_2v_4^{n+1}u_3 \in L$ *for all* $n \geq 0$.

Vijayashanker (1987) casually remarks that the cases (b) and (c) reduce to the case (a), but as Kallmeyer (2010) notes, he does not explain why that is so. The above example shows that much more work is needed to prove the strong form of the pumping lemma for TALs through analysis of derivation trees:

Theorem 4.2 (Pumping Lemma for TALs)**.** *Let L be a TAL. Then there is a natural number p such that for every string $z \in L$ with $|z| \geq p$, there are strings $u_1, v_1, w_1, v_2, u_2, v_3, w_2, v_4, u_3$ satisfying the following conditions:*

1. $z = u_1v_1w_1v_2u_2v_3w_2v_4u_3$,
2. $|v_1v_2v_3v_4| \geq 1$,
3. $|v_1w_1v_2v_3w_2v_4| \leq p$, *and*
4. $u_1v_1^nw_1v_2^nu_2v_3^nw_2v_4^nu_3 \in L$ *for all* $n \geq 0$.

We shall approach this problem through an analysis of TAG *derived trees* in terms of *linear indexed grammars*.

Linear indexed grammars

Linear indexed grammars are a restrictive variant of *indexed grammars* (Aho 1968) and were introduced—but no so named—by Gazdar (1988).[6] The class of languages generated by linear indexed grammars coincides with the class of TALs (Vijay-Shanker and Weir 1994). The following is an indexed grammar generating COPY:

$$\begin{aligned} S[\circ\circ] &\rightarrow \mathtt{a}\ S[\mathtt{a}\circ\circ] \\ S[\circ\circ] &\rightarrow \mathtt{b}\ S[\mathtt{b}\circ\circ] \\ S[\circ\circ] &\rightarrow T[\circ\circ] \\ T[\mathtt{a}\circ\circ] &\rightarrow T[\circ\circ]\ \mathtt{a} \\ T[\mathtt{b}\circ\circ] &\rightarrow T[\circ\circ]\ \mathtt{b} \\ T[] &\rightarrow \varepsilon \end{aligned}$$

In indexed grammar derivations, each nonterminal is followed by a string of indices, which acts as a pushdown stack.[7] If B is a nonterminal and χ is a string

[6]The term *linear indexed grammar* is apparently due to Vijayashanker (1987).

[7]The top of the stack is on the left. This convention from Aho's (1968) original paper on indexed grammars has been reversed by some people, e.g., Vijay-Shanker and Weir (1994).

of indices, we write $B[\chi]$ for $B\chi$. Formally, a linear indexed grammar is a 5-tuple $G = (N, \Sigma, I, P, S)$, where

1. N and Σ are finite sets of nonterminals and terminals, respectively,
2. I is a finite set of *indices*,
3. $S \in N$, and
4. P is a finite set of productions, each having one of the following forms:

$$\begin{aligned} A[\circ\circ] &\to \alpha B[\circ\circ]\beta, \\ A[\circ\circ] &\to \alpha B[f\circ\circ]\beta, \\ A[f\circ\circ] &\to \alpha B[\circ\circ]\beta, \\ A[] &\to w, \end{aligned}$$

   where $A, B \in N$, $f \in I$, $\alpha, \beta \in (N[] \cup \Sigma)^*$, $w \in \Sigma^*$.

The expression $\circ\circ$ serves as a variable for strings of indices.

For $\gamma, \delta \in (NI^* \cup \Sigma)^*$, we write $\gamma \Rightarrow_G \delta$ if there are $A \in N$, $\chi \in I^*$, $\gamma_1, \gamma_2 \in (NI^* \cup \Sigma)^*$ such that $\gamma = \gamma_1 A[\chi]\gamma_2$ and one of the following holds:

- $A[\circ\circ] \to \alpha B[\circ\circ]\beta$ is a production of G and $\delta = \gamma_1\alpha B[\chi]\beta\gamma_2$;
- $A[\circ\circ] \to \alpha B[f\circ\circ]\beta$ is a production of G and $\delta = \gamma_1\alpha B[f\chi]\beta\gamma_2$;
- $A[f\circ\circ] \to \alpha B[\circ\circ]\beta$ is a production of G, $\chi = f\chi'$, and $\delta = \gamma_1\alpha B[\chi']\beta\gamma_2$;
- $A[] \to w$ is a production of G, $\chi = \varepsilon$, and $\delta = \gamma_1 w\gamma_2$.

Here's an example of a derivation of the above grammar for COPY:

$$\begin{aligned} S[] \Rightarrow \mathtt{a}S[\mathtt{a}] \Rightarrow \mathtt{aa}S[\mathtt{aa}] \Rightarrow \mathtt{aab}S[\mathtt{baa}] \Rightarrow \mathtt{aab}T[\mathtt{baa}] \Rightarrow \mathtt{aab}T[\mathtt{aa}]\mathtt{b} \\ \Rightarrow \mathtt{aab}T[\mathtt{a}]\mathtt{ab} \Rightarrow \mathtt{aab}T[]\mathtt{aab} \Rightarrow \mathtt{aabaab}. \end{aligned}$$

The language generated by a LIG $G = (N, \Sigma, I, P, S)$ is defined as follows:

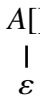
$$L(G) = \{\, w \in \Sigma^* \mid S[] \Rightarrow_G^* w \,\}.$$

If $L = L(G)$ for some LIG G, L is said to be a *linear indexed language* (LIL).

We can define *derivation trees* as well:

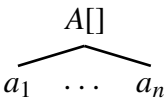
1. A tree consisting of a single node labeled by $a \in \Sigma \cup \{\varepsilon\}$ is a derivation tree from a.

2. If $A[] \to \varepsilon$ is a production, then



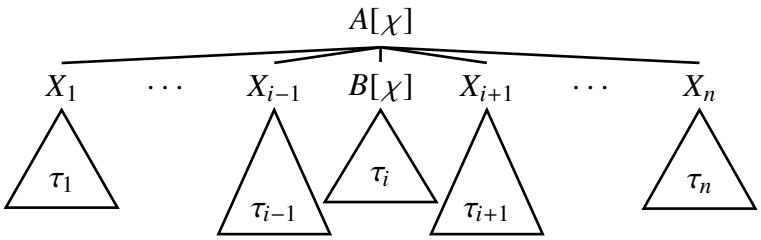
is a derivation tree from $A[]$.

3. If $A[] \to a_1 \ldots a_n$ $(n \geq 1, a_i \in \Sigma)$ is a production, then



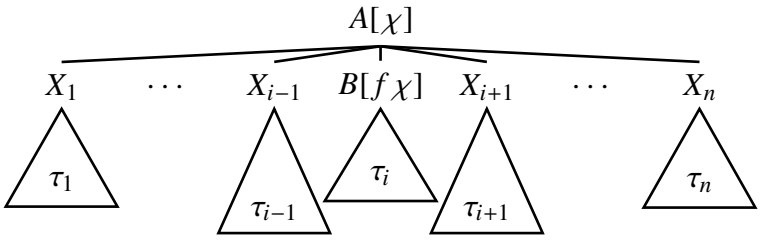
is a derivation tree from $A[]$.

4. If $A[\circ\circ] \to X_1 \ldots X_{i-1}\, B[\circ\circ]\, X_{i+1} \ldots X_n$ is a production $(X_i \in N[] \cup \Sigma)$ and $\tau_1, \ldots, \tau_n$ are derivation trees from $X_1, \ldots, X_{i-1}, B[\chi], X_{i+1}, \ldots, X_n$, respectively, then



is a derivation tree from $A[\chi]$.

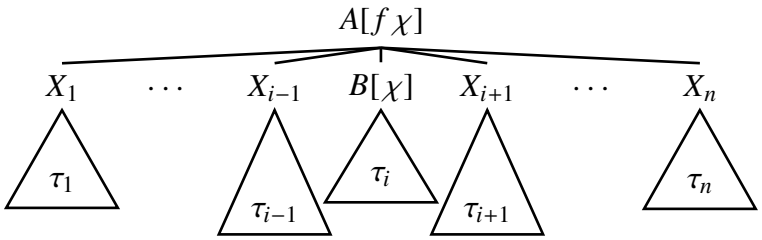
5. If $A[\circ\circ] \to X_1 \ldots X_{i-1}\, B[f\circ\circ]\, X_{i+1} \ldots X_n$ is a production $(X_i \in N[] \cup \Sigma)$ and $\tau_1, \ldots, \tau_n$ are derivation trees from $X_1, \ldots, X_{i-1}, B[f\,\chi], X_{i+1}, \ldots, X_n$, respectively, then



is a derivation tree from $A[\chi]$.

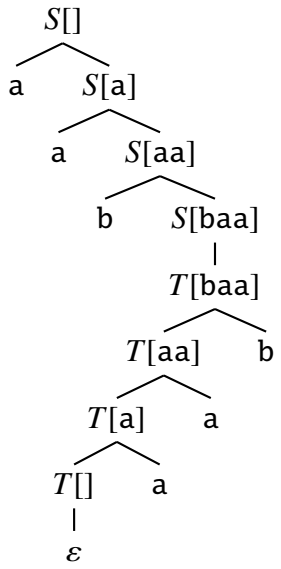
6. If $A[f\circ\circ] \to X_1 \ldots X_{i-1}\, B[\circ\circ]\, X_{i+1} \ldots X_n$ is a production $(X_i \in N[] \cup \Sigma)$ and $\tau_1, \ldots, \tau_n$ are derivation trees from $X_1, \ldots, X_{i-1}, B[\chi], X_{i+1}, \ldots, X_n$,

respectively, then



is a derivation tree from $A[f\,\chi]$.

The following is an example of a derivation tree of the above grammar for COPY:



We have

$A[\chi] \Rightarrow_G^* w$ if and only if there is a derivation tree from $A[\chi]$ whose yield is w.

A *complete* derivation tree is a derivation tree from $S[]$. As a consequence,

$$L(G) = \{\, \mathbf{y}(\tau) \mid \tau \text{ is a complete derivation tree of } G \,\}.$$

Consider the following linear indexed grammar G:

$$S[\circ\circ] \to S[\mathsf{a}\circ\circ]\,\mathsf{a}$$
$$S[\circ\circ] \to S[\mathsf{b}\circ\circ]\,\mathsf{b}$$
$$S[\circ\circ] \to T[\circ\circ]$$

$$T[\mathsf{a}\circ\circ] \to T[\circ\circ]\,\mathsf{a}$$
$$T[\mathsf{b}\circ\circ] \to T[\circ\circ]\,\mathsf{b}$$
$$T[] \to \varepsilon$$

This grammar generates $\{\, ww^R \mid w \in \{\mathsf{a},\mathsf{b}\}^* \,\}$, where w^R denotes the reversal of w. This language is a non-regular context-free language, but the derivation trees of G are all purely left-branching. Gazdar (1988) suggests that this property of linear indexed grammars may be potentially useful to analyze sentences like the following:

(4.3) Jude is [less]$_b$ obviously [as]$_a$ nice [as]$_a$ Kim [than]$_b$ Chris is.

(4.4) You are [as]$_b$ much tall[er]$_a$ [than]$_a$ me [as]$_b$ I expected.

(4.5) This fence is [so much]$_c$ [too much]$_b$ high[er]$_a$ [than]$_a$ that one [for]$_b$ me to even consider climbing it [that]$_c$ it's simply incomprehensible to me that Mary would try to get me to do it.

The first two of these sentences are originally due to Klein (1981) and the third to Bowers (1975). These constructions exhibit nested dependency, but it seems reasonable to assume that their tree structures are predominantly left-branching.

Exercise 4.3. Write a LIG that generates $\{\, ww^R \mid w \in D_1^* \,\}$ (cf. Exercise 4.2). Can you make the derivation trees purely left-branching?

From tree-adjoining grammars to linear indexed grammars

Given a TAG $G = (N, \Sigma, \mathscr{I}, \mathscr{A})$, we construct a LIG $G' = (N', \Sigma, I', P', S')$ such that $\mathbf{y}L(G) = L(G')$. Assume that the foot node of every auxiliary tree has an NA constraint. (If not, add a new NA node under the foot node, making it the new foot node.) The *spine* of an auxiliary tree is the path from its root to its foot node.

1. $N' = \{\, (\alpha, q) \mid \alpha \in \mathscr{I} \cup \mathscr{A}$ and q is a node of $\alpha \,\} \cup \{S'\}$;
2. $I' = \{\, (\alpha, q) \mid \alpha \in \mathscr{I} \cup \mathscr{A}$ and q is an internal node of α labeled by $OA(\mathscr{C})$ or $SA(\mathscr{C})$ for some $\mathscr{C} \neq \varnothing \,\}$;
3. P' is constructed as follows:
   (a) for every $\alpha \in \mathscr{I}$, P' contains the production
   $$S'[\circ\circ] \to (\alpha, \varepsilon)[\circ\circ].$$

(b) for every $\alpha \in \mathscr{I} \cup \mathscr{A}$, if p is an internal node of α that does not have an OA constraint, then P' contains the production

$$(\alpha, p)[\circ\circ] \rightarrow$$
$$(\alpha, p.1)[] \ldots (\alpha, p.(k-1))[]\ (\alpha, p.k)[\circ\circ]\ (\alpha, p.(k+1))[] \ldots (\alpha, p.n)[],$$

where n is the number of children of p in α, and if p is on the spine of α, then $p.k$ is on the spine of α, and otherwise $k = 1$;

(c) for every $\alpha \in \mathscr{I} \cup \mathscr{A}$, if p is a leaf node of α labeled by some $a \in \Sigma \cup \{\varepsilon\}$, then P' contains the production

$$(\alpha, p)[] \rightarrow a.$$

(d) for every $\alpha \in \mathscr{I} \cup \mathscr{A}$ and $\beta \in \mathscr{A}$, if p is an internal node of α labeled by $OA(\mathscr{C})$ or $SA(\mathscr{C})$ for some $\mathscr{C}$ such that $\beta \in \mathscr{C}$, then P' contains the productions

$$(\alpha, p)[\circ\circ] \rightarrow (\beta, \varepsilon)[(\alpha, p)\circ\circ],$$
$$(\beta, q)[(\alpha, p)\circ\circ] \rightarrow$$
$$(\alpha, p.1)[] \ldots (\alpha, p.(k-1))[]\ (\alpha, p.k)[\circ\circ]\ (\alpha, p.(k+1))[] \ldots (\alpha, p.n)[],$$

where q is the foot node of β, n is the number of children of p in α, and if p is on the spine of α, $p.k$ is on the spine of α and otherwise $k = 1$.

If G' is the LIG constructed from a TAG G by the above procedure, the derivation trees of G' have almost the same shape as the derived trees of G, and it can be proved that $\mathbf{y}L(G) = L(G')$.

Theorem 4.3. *Every TAL is a LIL.*

Proof. Let G be a TAG such that the foot node of every auxiliary tree has an NA constraint, and let G' be the LIG constructed from G by the above method. Let G'' be the LIG which is just like G' except that G'' contains (b) productions for all internal nodes of α, including those with an OA constraint. By a *derivation tree fragment* of G'', we mean a tree just like a derivation tree of G'' except that it may have a leaf node with a label of the form $(\beta, q)[]$, where β is an auxiliary tree of G and q is its foot node (it may have at most one such leaf and all the other leaves must be labeled by terminals). Each elementary tree α of G corresponds to a derivation tree fragment τ_α of G'' constructed with only (b) and (c) productions, in an obvious way.

Let τ be a derivation tree fragment of G''. We call a node r in τ a *push* node if r is labeled with $(\alpha, p)[\chi]$ and the only child of r is labeled with $(\beta, \varepsilon)[(\alpha, p)\chi]$. We modify τ by deleting some nodes and relabeling others, as follows.

- Any node r in τ labeled by $S'[]$ is deleted.
- If a node r in τ is labeled by $(\alpha, p)[\chi]$ and the first component of $\alpha(p)$ is a nonterminal, then r is relabeled with $\alpha(p)$, except when r is a push node, in which case r is deleted.
- If a node r in τ is labeled by $(\alpha, p)[]$ and $\alpha(p)$ is a terminal, then r is deleted.
- Any node in τ labeled by $a \in \Sigma \cup \{\varepsilon\}$ is unchanged.

We denote the resulting tree by $\phi(\tau)$.

Claim 1. If τ is a derivation tree fragment of G'' whose root node is labeled with $(\delta, \varepsilon)[]$, then $\delta \Rightarrow_G^* \phi(\tau)$.

We prove Claim 1 by induction on the number of push nodes in τ. If τ has no push node, it is easy to see that $\tau = \tau_\delta$ and $\phi(\tau) = \delta$. Otherwise, let r be one of the lowest push nodes of τ. Let the labels of r and its only child be $(\alpha, p)[\chi]$ and $(\beta, \varepsilon)[(\alpha, p)\chi]$, respectively. There must be a descendant s of r labeled by $(\beta, q)[(\alpha, p)\chi]$ with a child labeled by $(\alpha, p.k)[\chi]$, where q is the foot node of β and $p.k$ is a child of p in α. The nodes on the path from r to s all have labels of the form $(\beta, t)[(\alpha, p)\chi]$. (Note that none of them are push nodes.) Let τ' be the result of removing from τ all descendants of r except those that are descendants of s, making the children of s new children of r. Then τ' is a derivation tree fragment of G'', and the removed nodes will constitute a derivation tree fragment υ if we change the labels of the nodes that were on the path from r to s from $(\beta, t)[(\alpha, p)\chi]$ to $(\beta, t)[]$. By induction hypothesis, $\delta \Rightarrow_G^* \phi(\tau')$. It is easy to see that $\upsilon = \tau_\beta$, $\phi(\upsilon) = \beta$, and $\phi(\tau) = \phi(\tau')[u \leftarrow \beta]$, where u is the node of $\phi(\tau')$ that r gets mapped to. (Since push nodes above r are removed, u is a scattered substring of r, i.e., u is obtained from r by skipping some integers.) This proves Claim 1.

Claim 2. If δ is an elementary tree of G and $\delta \Rightarrow_G^* \gamma$, then there is a derivation tree fragment τ of G'' whose root node is labeled with $(\delta, \varepsilon)[]$ such that $\gamma = \phi(\tau)$.

If $\gamma = \delta$, then we can take $\tau = \tau_\delta$. Suppose $\delta \Rightarrow_G^* \delta'$ and $\gamma = \delta'[u \leftarrow \beta]$ for some $\beta \in \mathscr{A}$. By induction hypothesis, there is a derivation tree fragment τ' of G'' such that $\phi(\tau') = \delta'$. Let r be the node of τ' that is mapped to u by ϕ. Since u does not have an NA constraint, it originates as an internal node (α, p) of some elementary tree of G. In particular, p is not the foot node of α. Since r is not removed by ϕ, it is not a push node, so the label of r must be $(\alpha, p)[\chi]$ for some $\chi \in I^*$ and the labels of the children of r must be $(\alpha, p.1)[], \ldots, (\alpha, p.(k-1))[], (\alpha, p.k)[\chi], (\alpha, p.(k+1))[], \ldots, (\alpha, p.n)$, where n is the number of children of p in α and $p.k$ is on the spine of α if p is. Let υ be the tree that results from τ_β by changing all labels $(\beta, t)[]$ such that t is on the spine of β to $(\beta, t)[(\alpha, p)\chi]$. Let τ

be the result of 'inserting' υ immediately below r in τ'. (This insertion operation is different from adjunction in that all nodes of τ' are present in the resulting tree.) Clearly, τ is a derivation tree fragment of G'', and $\phi(\tau) = \gamma$. This proves Claim 2.

Now we can show

(†) $$L(G) = \{\, \phi(\tau) \mid \tau \text{ is a complete derivation tree of } G' \,\}.$$

It immediately follows from this that $\mathbf{y}L(G) = L(G')$.

To prove (†), suppose first that τ is a complete derivation tree of G'. Then its unique immediate subtree is a derivation tree of G'' whose root node is labeled with (δ, ε) for some $\delta \in \mathscr{I}$. By Claim 1, $\delta \Rightarrow^*_G \phi(\tau)$. Now suppose (α, p) labels a node r of τ, where the node p of α has an OA constraint. Then by the restriction on (b) productions of G', r must be a push node, so r is removed by ϕ. This means that $\phi(\tau)$ has no node with an OA constraint, and hence $\phi(\tau) \in L(G)$.

Conversely, suppose $\gamma \in L(G)$. Then there is a $\delta \in \mathscr{I}$ such that $\delta \Rightarrow^*_G \gamma$. By Claim 2, there is a derivation tree υ of G'' whose root node is labeled with $(\delta, \varepsilon)[]$ such that $\gamma = \phi(\upsilon)$. Since γ contains no OA node, it is clear that υ is a derivation tree of G'. Let τ be a complete derivation tree of G' with υ as its unique immediate subtree. We have $\phi(\tau) = \phi(\upsilon) = \gamma$.

This shows (†), completing the proof of the theorem. □

Exercise 4.4. Take the example TAG that was used above to illustrate the weak pumping lemma for tree-adjoining languages and convert it to a LIG.

The pumping lemma for linear indexed languages

Consider a derivation tree τ for a LIG G. If q is an internal node of τ sanctioned by a production of one of the following types:

$$A[\circ\circ] \to \alpha B[\circ\circ]\beta,$$
$$A[\circ\circ] \to \alpha B[f\circ\circ]\beta,$$
$$A[f\circ\circ] \to \alpha B[\circ\circ]\beta,$$

where $|\alpha| = k$ (note $\alpha \in (N \cup \Sigma)^*$), then the $(k+1)$st child of q is called its *head child*. Suppose that a path $q_1, \ldots, q_n$ in τ satisfies the following properties:

- q_1 is not the head child of any node,
- for $i = 1, \ldots, n-1$, q_{i+1} is the head child of q_i, and
- q_n does not have a head child.

Then we call $q_1, \ldots, q_n$ a *spine*. Every spine starts with a node labeled by $A[]$ for some $A \in N$ and ends in a node labeled by $B[]$ for some $B \in N$ whose children are all leaves. Every internal node of a derivation tree is on a unique spine.[8]

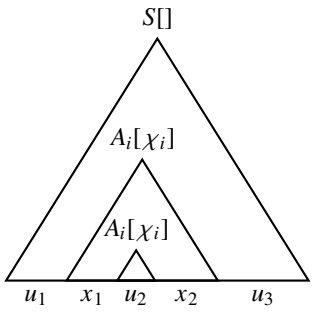
Theorem 4.4 (Pumping Lemma for LILs)**.** *Let L be a linear indexed language. Then there is a natural number p such that for every string $z \in L$ with $|z| \geq p$, there exist strings $u_1, v_1, w_1, v_2, u_2, v_3, w_2, v_4, u_3$ satisfying the following conditions:*

1. $z = u_1 v_1 w_1 v_2 u_2 v_3 w_2 v_4 u_3$,
2. $|v_1 v_2 v_3 v_4| \geq 1$, *and*
3. $u_1 v_1^n w_1 v_2^n u_2 v_3^n w_2 v_4^n u_3 \in L$ *for all* $n \geq 0$.

Proof. Let $G = (N, \Sigma, I, P, S)$ be a LIG generating L. Let $s = \sum_{n=0}^{|N|^2} |I|^n$, the number of strings of indices whose length does not exceed $|N|^2$. Let l be the maximal length of the right-hand side of productions in P, and let $p = l^{|N| \cdot s} + 1$.

Suppose $z \in L(G)$ and $|z| \geq p$. Let τ be one of the smallest complete derivation trees such that $\mathbf{y}(\tau) = z$. Since $|z| \geq p$, there must be a path from the root node of τ leading to a leaf that contains at least $h = |N| \cdot s + 1$ internal nodes. Let $q_1, \ldots, q_h$ be the lowest h internal nodes on this path, and for $i = 1, \ldots, h$, let $A_i[\chi_i]$ be the label of q_i.

Case 1. For all $i = 1, \ldots, h$, $|\chi_i| \leq |N|^2$. Then there must be i, j such that $i < j \leq h$ and $A_i[\chi_i] = A_j[\chi_j]$. The derivation tree τ looks like the following:



The three regions of this derivation represent derivations

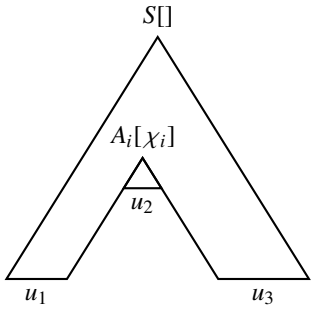
$$\begin{aligned} S[] &\Rightarrow_G^* u_1 A_i[\chi_i] u_3, \\ A_i[\chi_i] &\Rightarrow_G^* x_1 A_i[\chi_i] x_2, \\ A_i[\chi_i] &\Rightarrow_G^* u_2, \end{aligned}$$

[8]This is not strictly true, because the same internal node may be sanctioned by more than one production. We assume that we are given an assignment of a unique production to each internal node.

where $z = u_1x_1u_2x_2u_3$. Combining these three derivations, it is clear that we have

$$\begin{aligned} S[] &\Rightarrow_G^* u_1 A_i[\chi_i] u_3 \\ &\Rightarrow_G^* u_1 x_1^n A_i[\chi_i] x_2^n u_3 \\ &\Rightarrow_G^* u_1 x_1^n u_2 x_2^n u_3 \in L \end{aligned}$$

for every $n \geq 0$. If $x_1 = x_2 = \varepsilon$, then $z = u_1u_2u_3$ has the following smaller derivation tree:



This contradicts the assumption that τ is oen of the smallest derivation trees for z. Therefore, $|x_1x_2| \geq 1$. We obtain the desired conditions by putting $v_1 = x_1$, $w_1 = v_2 = v_3 = w_2 = \varepsilon$, and $v_4 = x_2$.

Case 2. For some m, $|\chi_m| > |N|^2$. Let $r_1, \ldots, r_{\hat{n}}$ be the spine that q_m is on, and let $r_{\hat{m}} = q_m$. Let the label of r_i be $B_i[\zeta_i]$. Define

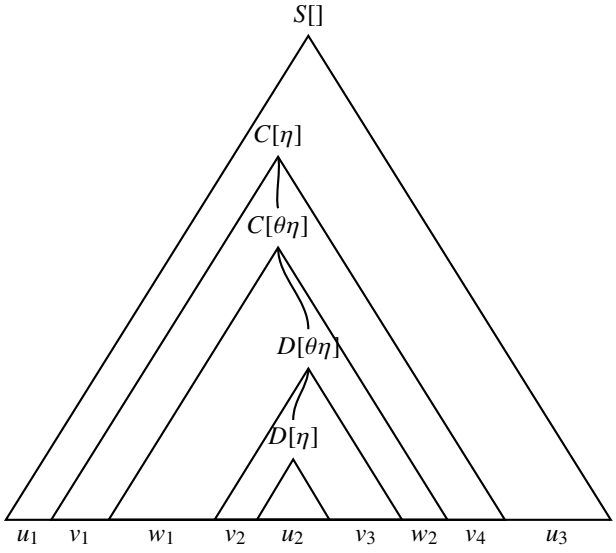
$$Q = \{\, (i,j) \mid 1 \leq i < \hat{m} < j \leq \hat{n},\ \zeta_i = \zeta_j, \text{ and for all } k \text{ such that } i < k < j,\ |\zeta_k| > |\zeta_i| \,\}.$$

It is easy to see that for each $k < |\chi_m|$, there is exactly one pair (i,j) in Q such that $|\zeta_i| = k$, so Q has exactly $|\chi_m|$ elements. Since $|\chi_m| > |N|^2$, there are pairs $(i,j), (i',j') \in Q$ such that $i < i' < j' < j$ and

$$B_i = B_{i'} = C, \qquad B_j = B_{j'} = D.$$

Then $\zeta_i = \zeta_j = \eta$ and $\zeta_{i'} = \zeta_{j'} = \theta\eta$ for some $\eta, \theta \in I^*$. The derivation tree τ

looks like the following:



The five regions of this derivation tree represent derivations

$$\begin{aligned} S[] &\Rightarrow_G^* u_1 C[\eta] u_3, \\ C[\eta] &\Rightarrow_G^* v_1 C[\theta\eta] v_4, \\ C[\theta\eta] &\Rightarrow_G^* w_1 D[\theta\eta] w_2, \\ D[\theta\eta] &\Rightarrow_G^* v_2 D[\eta] v_3, \\ D[\eta] &\Rightarrow_G^* u_2, \end{aligned}$$

where $z = u_1 v_1 w_1 v_2 u_2 v_3 w_2 v_4 u_3$. By the definition of Q, we must have

$$\begin{aligned} C[\xi] &\Rightarrow_G^* v_1 C[\theta\xi] v_4, \\ C[\xi] &\Rightarrow_G^* w_1 D[\xi] w_2, \\ D[\theta\xi] &\Rightarrow_G^* v_2 D[\xi] v_3 \end{aligned}$$

for every $\xi \in I^*$, and this implies for every $n \geq 0$,

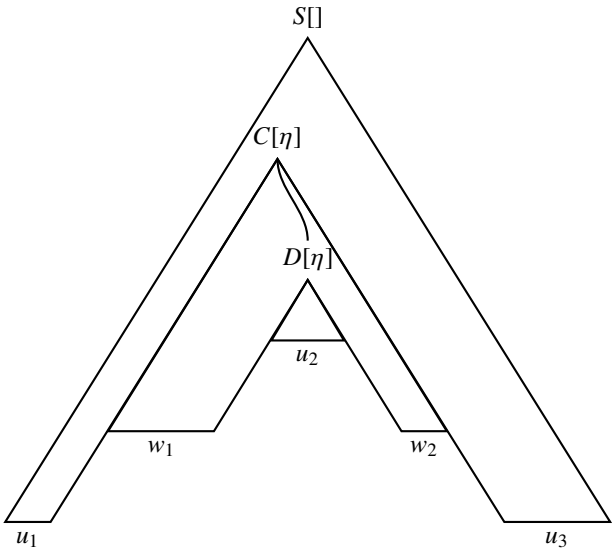
$$\begin{aligned} C[\theta^n\eta] &\Rightarrow_G^* v_1 C[\theta^{n+1}\eta] v_4, \\ C[\theta^{n+1}\eta] &\Rightarrow_G^* w_1 D[\theta^{n+1}\eta] w_2, \\ D[\theta^{n+1}\eta] &\Rightarrow_G^* v_2 D[\theta^n\eta] v_3. \end{aligned}$$

Therefore, for every $n \geq 0$,

$$S[] \Rightarrow_G^* u_1 C[\eta] u_3$$

$$
\begin{aligned}
&\Rightarrow^*_G u_1 v_1^n C[\theta^n \eta] v_4^n u_3 \\
&\Rightarrow^*_G u_1 v_1^n w_1 D[\theta^n \eta] w_2 v_4^n u_3 \\
&\Rightarrow^*_G u_1 v_1^n w_1 v_2^n D[\eta] v_3^n w_2 v_4^n u_3 \\
&\Rightarrow^*_G u_1 v_1^n w_1 v_2^n u_2 v_3^n w_2 v_4^n u_3 \in L.
\end{aligned}
$$

If $v_1 = v_2 = v_3 = v_4 = \varepsilon$, then $z = u_1 w_1 u_2 w_2 u_3$ has the following smaller derivation tree:



This contradicts the assumption that τ is one of the smallest derivation trees for z. Therefore, $|v_1 v_2 v_3 v_4| \geq 1$.

This completes the proof. □

Since every TAL is a LIL, Theorem 4.4 applies to every TAL.

Exercise 4.5. Show that the following languages are not TALs (or LILs):

- COUNT-5 = $\{ \mathsf{a}^n \mathsf{b}^n \mathsf{c}^n \mathsf{d}^n \mathsf{e}^n \mid n \geq 0 \}$.
- COPY-2 = $\{ www \mid w \in \{\mathsf{a}, \mathsf{b}\}^* \}$.

Palis and Shende (1995) proved an analogue of Ogden's (1968) lemma for each level of the *control language hierarchy* introduced by Weir (1988, 1992). The first level of this hierarchy is the class of CFLs, and the second level coincides with the class of TALs (LILs). Palis and Shende's (1995) proof is essentially a generalization of the proof for LILs we have given above.[9]

[9]Essentially the same proof also appears in Shamir 2013.

From Linear Indexed Grammars to Tree-Adjoining Grammars

It is known that the converse of Theorem 4.3 also holds: every LIL is a TAL. In fact, we can prove a stronger statement: for every LIG G, there is a TAG G' such that $L(G')$ consists precisely of the *stripped complete derivation trees* of G, where a stripped complete derivation tree is the result of removing the stack portion of the node labels from a complete derivation tree.

Let $G = (N, \Sigma, I, P, S)$ be a LIG. For the sake of simplicity, let us suppose that each production in P has one of the following forms:

$$\begin{aligned} A[\circ\circ] &\to B_1[] \ldots B_{i-1}[]\, B_i[\circ\circ]\, B_{i+1}[] \ldots B_n[] \\ A[\circ\circ] &\to B_1[] \ldots B_{i-1}[]\, B_i[f\circ\circ]\, B_{i+1}[] \ldots B_n[] \\ A[f\circ\circ] &\to B_1[] \ldots B_{i-1}[]\, B_i[\circ\circ]\, B_{i+1}[] \ldots B_n[] \\ A[] &\to w \end{aligned}$$

In other words, terminals and nonterminals never appear mixed in the right-hand side of any production. (It is not difficult to lift this restriction.)

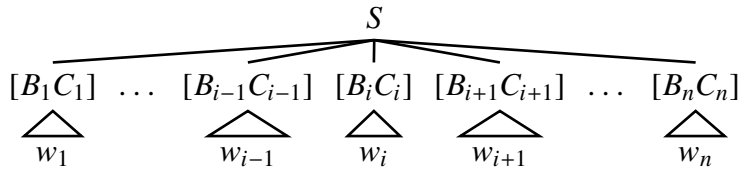
We construct a tree-adjoining grammar $G' = (N', \Sigma, \mathscr{I}, \mathscr{A})$ corresponding to G. Define

$$N' = N \cup \{\, [AB] \mid A, B \in N \,\}.$$

We assume that the symbols in N are always used with the NA constraint, whereas a symbol of the form $[AB]$ is always used with the constraint $OA(\mathscr{C}_{A,B})$, where $\mathscr{C}_{A,B}$ refers to the set of auxiliary trees whose root is labeled by A and whose foot node is labeled either by B or by a nonterminal of the form $[DB]$.

The set $\mathscr{I}$ of initial trees of G' is defined as follows:

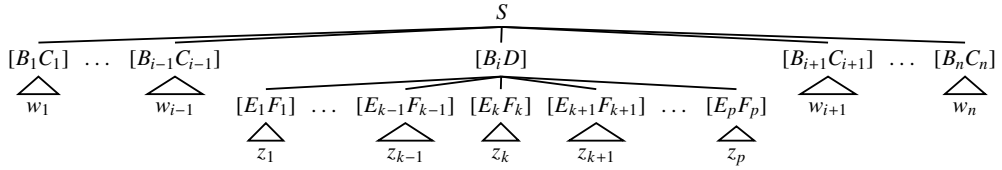
- If
$$\begin{aligned} S[\circ\circ] &\to B_1[] \ldots B_{i-1}[]\, B_i[\circ\circ]\, B_{i+1}[] \ldots B_n[] \\ C_j[] &\to w_j \quad (j = 1, \ldots, n) \end{aligned}$$
are productions in P, then the following tree is in $\mathscr{I}$:



- If

$$\begin{aligned} S[\circ\circ] &\to B_1[] \ldots B_{i-1}[]\, B_i[f\circ\circ]\, B_{i+1}[] \ldots B_n[] \\ C_j[] &\to w_j \quad (j = 1, \ldots, i-1, i+1, \ldots, n) \\ D[f\circ\circ] &\to E_1[] \ldots E_{k-1}[]\, E_k[\circ\circ]\, E_{k+1}[] \ldots E_p[] \\ F_l[] &\to z_l \quad (l = 1, \ldots, p) \end{aligned}$$

  are productions in P, then the following tree is in $\mathscr{I}$:



- If

$$S[] \to w$$

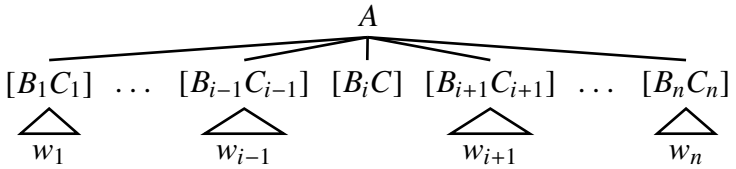
  is a production in P, then the following tree is in $\mathscr{I}$:

The set $\mathscr{A}$ of auxiliary trees of G' is defined as follows:

- If

$$\begin{aligned} A[\circ\circ] &\to B_1[] \ldots B_{i-1}[]\, B_i[\circ\circ]\, B_{i+1}[] \ldots B_n[] \\ C_j[] &\to w_j \quad (j = 1, \ldots, i-1, i+1, \ldots, n) \end{aligned}$$

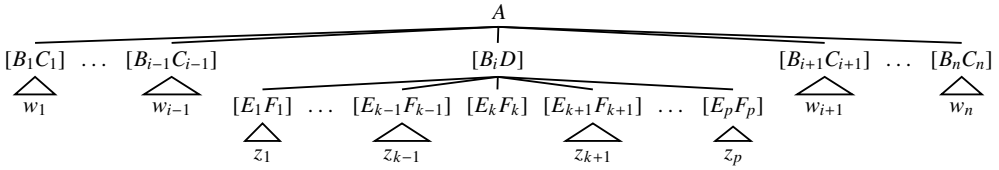
  are productions in P and C is a nonterminal in N, then the following tree is in $\mathscr{A}$:



- If

$$\begin{aligned} A[\circ\circ] &\to B_1[] \ldots B_{i-1}[]\, B_i[f\circ\circ]\, B_{i+1}[] \ldots B_n[] \\ C_j[] &\to w_j \quad (j = 1, \ldots, i-1, i+1, \ldots, n) \\ D[f\circ\circ] &\to E_1[] \ldots E_{k-1}[]\, E_k[\circ\circ]\, E_{k+1}[] \ldots E_p[] \\ F_l[] &\to z_l \quad (l = 1, \ldots, k-1, k+1, \ldots, p) \end{aligned}$$

are productions in P, then the following tree is in $\mathscr{A}$:



- For each $A \in N$, the following tree is in $\mathscr{A}$:

  A

  (This is the tree consisting of a single node labeled by A and can be adjoined into a node labeled by $[AA]$.)

As before, by a *derivation tree fragment* of G, we mean a tree that is just like a derivation tree of G except that it has a unique leaf with a label of the form $B[]$, where B is a nonterminal. If τ is a derivation tree or a derivation tree fragment, we write $\overline{\tau}$ for the result of removing all occurrences of indices in τ. The *main spine* of a derivation tree (context) refers to the spine that contains the root. We can prove the following:

(a) If τ is a complete derivation tree of G, then $\overline{\tau} \in L(G')$.

(b) If τ is a derivation tree fragment of G whose root is labeled by $A[]$ and which has a leaf labeled by $B[]$ on the main spine, then $\upsilon \Rightarrow^*_{G'} \overline{\tau}$ for some $\upsilon \in \mathscr{C}_{A,B}$.

Conversely, we can show

(c) If $\tau' \in L(G')$, then there is a complete derivation tree τ in G such that $\tau' = \overline{\tau}$.

(d) If $\upsilon \Rightarrow^*_{G'} \tau'$ for some $\upsilon \in \mathscr{C}_{A,B}$ and τ' contains no label of the form $[CD]$, then there is a derivation tree fragment τ of G such that the root of τ is labeled by $A[]$, τ has a leaf labeled by $B[]$ on the main spine, and $\overline{\tau} = \tau'$.

Theorem 4.5. *For every LIG G, there is a TAG G' such that $L(G') = \{\, \overline{\tau} \mid \tau$ is a complete derivation tree of $G \,\}$.*

Corollary 4.6. *Every LIL is a TAL.*

Problems

4.1. Prove that the two definitions of $L(G)$ for TAGs are equivalent:

$$L(G) = \{\, \gamma \mid \exists \alpha \in \mathscr{I}(\alpha \Rightarrow_G^* \gamma \wedge \gamma \text{ has no OA node}) \,\}$$
$$L(G) = \{\, \mathrm{dtree}(\tau) \mid \tau \text{ is a complete derivation tree of } G \,\}$$

4.2. Prove that the class of LILs is closed under intersection with regular languages.

4.3. Call a production $A \to \alpha$ of a LIG *left-linear* if $\alpha \in (N[(\Sigma \cup \{\varepsilon\}) \circ\circ] \cup \{\varepsilon\})\Sigma^*$. A *left-linear* LIG is a LIG all of whose productions are left-linear. Show that the left-linear LIGs generate exactly the context-free languages. (See Gazdar 1988, Michaelis and Wartena 1997, 1998.)

4.4. Try to modify the definition of the LIG G' corresponding to a given TAG G in such a way that derived trees of G are obtained from derivation trees of G' simply by relabeling, without deleting any nodes.

4.5. Prove Theorem 4.4 with the condition $|v_1 w_1 v_2 v_3 w_2 v_4| \leq p$.

4.6. Prove the following weak form of Ogden's lemma for linear indexed grammars:

Let $G = (N, \Sigma, I, P, S)$ be a LIG. There is a natural number p such that for every string $z \in L(G)$ and $J \subseteq [1, |z|]$, if $|J| \geq p$, then z can be written as $z = u_1 v_1 w_1 v_2 u_2 v_3 w_2 v_4 u_3$ so that

1. at least one of the following sets is non-empty:

$$J \cap [|u_1| + 1, |u_1 v_1|],$$
$$J \cap [|u_1 v_1 w_1| + 1, |u_1 v_1 w_1 v_2|],$$
$$J \cap [|u_1 v_1 w_1 v_2 u_2| + 1, |u_1 v_1 w_1 v_2 u_2 v_3|],$$
$$J \cap [|u_1 v_1 w_1 v_2 u_2 v_3 w_2| + 1, |u_1 v_1 w_1 v_2 u_2 v_3 w_2 v_4|],$$

2. for some $C, D \in N$ and $\eta, \theta \in I^*$, we have

$$\begin{aligned} S[] &\Rightarrow_G^* u_1 C[\eta] u_3, \\ C[] &\Rightarrow_G^* v_1 C[\theta] v_4, \\ C[] &\Rightarrow_G^* w_1 D[] w_2, \\ D[\theta] &\Rightarrow_G^* v_2 D[] v_3, \\ D[\eta] &\Rightarrow_G^* u_2. \end{aligned}$$

(As a consequence, $u_1 v_1^n w_1 v_2^n u_2 v_3^n w_2 v_4^n u_3 \in L(G)$ for all $n \geq 0$.)

(See Palis and Shende 1995 for an analogue of Ogden's lemma for the control language hierarchy.)

4.7. Let $L = \{ \mathsf{a}^m\mathsf{b}^m\mathsf{c}^m\mathsf{d}^m\mathsf{e}^n \mid m, n \geq 0 \} \cup \{ \mathsf{a}^m\mathsf{b}^n\mathsf{c}^n\mathsf{d}^n\mathsf{e}^n \mid m, n \geq 0 \}$.

1. Show that L is a LIL.
2. Show that every LIG G such that $L = L(G)$ has two distinct derivation trees for some string of the form $\mathsf{a}^n\mathsf{b}^n\mathsf{c}^n\mathsf{d}^n\mathsf{e}^n$.

4.8. Complete the proof of Theorem 4.5. (To prove (c) and (d), use induction on derivation trees of G'.)

References

Abeillé, Anne and Owen Rambow. 2000. Tree adjoining grammar: An overview. In A. Abeillé and O. Rambow, eds., *Tree Adjoining Grammars: Formalisms, Linguistic Analysis and Processing*, pages 1–68. Stanford, Calif.: CSLI Publications.

Aho, Alfred V. 1968. Indexed grammars—an extension of context-free grammars. *Journal of the Association for Computing Machinery* 15:647–671.

Bowers, John S. 1975. Adjectives and adverbs in English. *Foundations of Language* 13:529–662.

Chomsky, Noam. 2004. *Generative Enterprise Revisited*. Berlin: Mouton de Gruyter.

Gazdar, Gerald. 1988. Applicability of indexed grammars to natural languages. In U. Reyle and C. Rohrer, eds., *Natural Language Parsing and Linguistic Theories*, pages 69–94. Dordrecht: Reidel.

Gorn, Saul. 1967. Explicit definitions and linguistic dominoes. In J. F. Hart and S. Takasu, eds., *Systems and computer science*, pages 77–115. Universit of Toronto Press in association with the University of Western Ontario.

Huybregts, Riny. 1984. The weak inadequacy of context free phrase structure grammars. In G. J. de Haan, M. Trommelen, and W. Zonneveld, eds., *Van Periferie naar Kern*, pages 81–90. Dordrecht: Foris Publications.

Joshi, Aravind K., Leon S. Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences* 10:136–163.

Joshi, Aravind K. and Yves Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages, Vol. 3: Beyond Words*, pages 69–123. Berlin: Springer.

Kallmeyer, Laura. 2010. *Parsing Beyond Context-Free Grammars*. Berlin: Springer.

Klein, Ewan. 1981. The syntax and semantics of nominal comparatives. In M. Moneglia, ed., *Atti de Seminario su Tempo e Verbale Strutture Quantificate in Forma Logica*. Florence: Presso l'Accademia della Crusca.

Knuth, Donald E. 1997. *The Art of Computer Programming. Volume 1: Fundamental Algorithms. Third Edition*. Reading, Massachusetts: Addison-Wesley.

Michaelis, Jens and Christian Wartena. 1997. How linguistic constraints on movement conspire to yield languages analyzable with a restricted form of LIGs. In G.-J. M. Kruijff, G. V. Morrill, and R. T. Oehrle, eds., *Formal Grammar 1997: Linguistic Aspects of Logical and Computational Perspectives on Language*, pages 158–168.

Michaelis, Jens and Christian Wartena. 1998. Unidirectional inheritance of indices: A weakly context free facet of LIGs. In G. Bouma, G.-J. M. Kruijff, and R. T. Oehrle, eds., *Proceedings of the FHCG'98: Joint Conference on Formal Grammar, Head-Driven Phrase Structure Grammar and Categorial Grammar*, pages 258–267.

Ogden, William. 1968. A helpful result for proving inherent ambiguity. *Mathematical Systems Theory* 2(3):191–194.

Palis, M. A. and S. M. Shende. 1995. Pumping lemmas for the control language hierarchy. *Mathematical Systems Theory* 28(3):199–213.

Rogers, James. 2003. Syntactic structures as multidimensional trees. *Research on Language and Computation* 1:265–305.

Shamir, Eli. 2013. Pumping, shrinking and pronouns: From context-free to indexed grammars. In A.-H. Dediu, C. Martín-Vide, and B. Truthe, eds., *Language and Automata Theory and Applications, LATA 2013*, pages 516–522. Berlin: Springer.

Shieber, Stuart M. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 8(3):333–343.

Vijay-Shanker, K. and D. J. Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory* 27:511–546.

Vijayashanker, K. 1987. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, University of Pennsylvania.

Weir, David J. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania.

Weir, David J. 1992. A geometric hierarchy beyond context-free languages. *Theoretical Computer Science* 104(2):235–261.