A Generalization of Linear Indexed Grammars Equivalent to Simple Context-Free Tree Grammars

Makoto Kanazawa

National Institute of Informatics, 2–1–2 Hitotsubashi, Chiyoda-ku, Tokyo, 101–8430, Japan

Abstract. I define a generalization of linear indexed grammars that is equivalent to simple context-free tree grammars in the same way that linear indexed grammars are equivalent to tree-adjoining grammars.

1 Introduction

The equivalence in string generating power of *tree-adjoining grammars*, *head grammars*, and *linear indexed grammars* is one of the most celebrated results in the mathematics of grammar formalisms for natural language [11,27].[1] The title of Joshi et al.'s paper [11], "The convergence of mildly context-sensitive grammar formalisms", referred to this equivalence, but was somewhat misleading in that the relevant class of string languages—*tree-adjoining languages*—was properly included in a larger class, the class of *multiple context-free languages*, which has widely been regarded as a formal counterpart of the informal notion of mild context-sensitivity. In fact, this latter class has also been found to be characterized by a wide array of different formalisms [30,3,31,22,20,21,8,6,24].

Elsewhere [12], I have argued that a class of string languages that falls in between these two classes, namely, the class equivalently captured by *well-nested multiple context-free grammars* [13],[2] *coupled-context-free grammars* [10], *non-duplicating macro grammars* [25], *simple* (i.e., linear and non-deleting) *context-free tree grammars* [16], and *second-order abstract categorial grammars* of lexicon complexity 3 (see [14]), may be more attractive than the broader class as a formalization of mild context-sensitivity. I will not repeat the arguments here,[3] but one counterargument might be that this intermediate class (the class

[1] I exclude *combinatory categorial grammars*, another formalism that was shown to be equivalent, from the discussion here, for two reasons. First, the equivalence was proved with respect to a certain *restricted* version of combinatory categorial grammars, and is not known to hold for more general combinatory grammars that are actually used in practice [26]. Second, the definition of that version of combinatory categorial grammar is mathematically not as natural as the other three formalisms.

[2] The same formalism is called *well-nested linear context-free rewriting systems* by some people [5].

[3] Simple context-free tree grammars are also of interest because of their capacity to *lexicalize* tree-adjoining grammars preserving the set of derived trees [19].

of well-nested multiple context-free languages) does not look as robust as the other two. The formalisms that capture it are all basically similar—they either define local sets of derivation trees that are evaluated bottom-up using "linear" functions (MCFGs and second-order ACGs) or present the same mechanism in a top-down rewriting perspective (coupled-context-free grammars, non-duplicating macro grammars, and simple context-free tree grammars). In contrast, at the level of tree-adjoining languages, linear indexed grammars have *non-local* (and non-regular) sets of derivation trees, and at the level of multiple context-free languages, *deterministic tree-walking transducers* [31] map trees to strings in a decidedly non-compositional way.

In this paper, I respond to this qualm by defining a natural generalization of linear indexed grammars, which generates the class of well-nested multiple context-free languages. This generalization, which I call *arboreal indexed grammars*, uses a "stack" attached to nonterminal symbols that stores tuples of trees, and is equivalent to simple context-free tree grammars in exactly the same way that linear indexed grammars are equivalent to tree-adjoining grammars (or more precisely, *monadic* simple context-free tree grammars [17]), in the following sense:

- For any simple context-free tree grammar, there is an arboreal indexed grammar such that the derived trees of the former may be obtained from the derivation trees of the latter by relabeling of nodes and deletion of some unary-branching nodes.
- For any arboreal indexed grammar, there is a simple context-free tree grammar whose derived trees are precisely the result of stripping the derivation trees of the arboreal indexed grammar of the "stack" part of their node labels.

The formalism of arboreal indexed grammar is closely related to the notion of Dyck tree language I introduced in [15]. This paper does not use this notion, however, and is completely self-contained.

Arboreal indexed grammars may be useful for devising new parsing algorithms for well-nested multiple context-free languages.

2 Indexed Grammars and Context-Free Tree Grammars

2.1 Indexed Grammars

An *indexed grammar* [1,9] is like a context-free grammar except that each occurrence of a nonterminal in a derivation tree has a string of indices attached to it, which acts as a pushdown stack. The stack is passed from a node to each of its nonterminal children, except that the production applied at that node may either push a symbol onto the stack or pop its topmost symbol.

A formal definition goes as follows. When B is a nonterminal and χ is a string of indices, we write $B[\chi]$ for $B\chi$; thus, $B[]$ is just B. An *indexed grammar* is a quintuple $G = (N, \Sigma, I, P, S)$, where

Table 1. Standard interpretation of indexed grammar productions ($\chi \in I^*$).

(TERM)	(DIST)	(PUSH)	(POP)
$A[] \to a$	$A[] \to B_1[] \dots B_n[]$	$A[] \to B[l]$	$A[l] \to B[]$
$\begin{array}{c}A[\chi]\\ \mid \\ a\end{array}$	$\begin{array}{c}A[\chi]\\ \diagup \quad \diagdown \\ B_1[\chi] \quad \cdots \quad B_n[\chi]\end{array}$	$\begin{array}{c}A[\chi]\\ \mid \\ B[l\chi]\end{array}$	$\begin{array}{c}A[l\chi]\\ \mid \\ B[\chi]\end{array}$

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:[4]

$$A[] \to a, \qquad \text{(TERM)}$$
$$A[] \to B_1[] \dots B_n[], \qquad \text{(DIST)}$$
$$A[] \to B[l], \qquad \text{(PUSH)}$$
$$A[l] \to B[], \qquad \text{(POP)}$$

where $a \in \Sigma \cup \{\varepsilon\}$, $n \geq 1$, $A, B, B_1, \dots, B_n \in N$, and $l \in I$.

A *derivation tree* of G is a finite labeled tree τ with node labels from $NI^* \cup \Sigma \cup \{\varepsilon\}$ such that

– each leaf node of τ is labeled by some $a \in \Sigma \cup \{\varepsilon\}$, and
– each internal node of τ is sanctioned by one of the productions of G,

where a node is said to be *sanctioned* by a production if it and its children are labeled as depicted in Table 1. For example, for a node to be sanctioned by a (TERM) production $A[] \to a$, it must be labeled by $A[\chi]$ for some $\chi \in I^*$, and its only child must be labeled by a. An internal node of a derivation tree is called a (TERM) node, (DIST) node, (PUSH) node, or (POP) node, depending on the type of production sanctioning it.

When a derivation tree has root label $A[\chi]$, we call it a derivation tree *from* $A[\chi]$. A *complete* derivation tree is a derivation tree from $S[]$. The *language* of G is defined by

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

where $\mathbf{y}(\tau)$ denotes the *yield* of τ, the left-to-right concatenation of the labels of its leaf nodes.

Note that in a derivation tree from $A[]$, each (POP) node must match exactly one (PUSH) node; a (PUSH) node may have zero, one, or more (POP) nodes matching it.[5]

[4] This is actually a normal form for indexed grammars which is more general than the normal form ("reduced form") given by Aho [1].

[5] I leave to the reader a formal definition of the intuitively clear notion of a (POP) node *matching* a (PUSH) node.

2.2 Context-Free Tree Grammars

A *ranked alphabet* is a union $\Delta = \bigcup_{n \in \mathbb{N}} \Delta^{(n)}$ of disjoint finite sets of symbols. If $f \in \Delta^{(n)}$, then n is the *rank* of f.

Let Σ be an (unranked) alphabet and Δ be a ranked alphabet. We define the set $\mathbb{T}_{\Sigma,\Delta}$ of trees over Σ, Δ as follows:

1. If $f \in \Sigma \cup \Delta^{(0)}$, then $f \in \mathbb{T}_{\Sigma,\Delta}$.
2. If $f \in \Sigma \cup \Delta^{(n)}$ and $t_1, \dots, t_n \in \mathbb{T}_{\Sigma,\Delta}$ $(n \geq 1)$, then $f(t_1, \dots, t_n) \in \mathbb{T}_{\Sigma,\Delta}$.

The notation $\mathbb{T}_\Sigma$ denotes the set of unranked trees over Σ; thus $\mathbb{T}_\Sigma = \mathbb{T}_{\Sigma,\varnothing}$.

We set aside special symbols $x_1, x_2, \dots$ called the *variables*. The set consisting of the first n variables $x_1, \dots, x_n$ is denoted X_n. The set $\mathbb{T}_{\Sigma,\Delta}(X_n)$ is defined to be $\mathbb{T}_{\Sigma,\Delta \cup X_n}$, where $\Delta \cup X_n$ is the ranked alphabet where symbols in X_n have rank 0. If $t[x_1, \dots, x_n] \in \mathbb{T}_{\Sigma,\Delta}(X_n)$ and $t_1, \dots, t_n \in \mathbb{T}_{\Sigma,\Delta}$, then $t[t_1, \dots, t_n]$ denotes the result of substituting t_i for each occurrence of x_i in t $(i = 1, \dots, n)$. Note that if x_i does not occur in $t[x_1, \dots, x_n]$, then t_i is deleted in $t[t_1, \dots, t_n]$, and if x_i occurs more than once in $t[x_1, \dots, x_n]$, then t_i is duplicated in $t[t_1, \dots, t_n]$. A tree $t[x_1, \dots, x_n] \in \mathbb{T}_{\Sigma,\Delta}(X_n)$ is called an *n-context* if each x_i occurs exactly once in it. If $t[x_1, \dots, x_n]$ is an n-context, then each t_i is neither deleted nor duplicated in $t[t_1, \dots, t_n]$.

We deviate from the standard practice and define context-free tree grammars using unranked alphabets of terminals. (This makes it easier to relate them to indexed grammars, but is not essential.) A *context-free tree grammar* [23,2] is a quadruple $G = (N, \Sigma, P, S)$, where

1. $N = \bigcup_{n \in \mathbb{N}} N^{(n)}$ is a finite ranked alphabet of nonterminals,
2. Σ is a finite unranked alphabet of terminals,
3. S is a nonterminal of rank 0, and
4. P is a finite set of productions of the form
$$A(x_1, \dots, x_n) \to t[x_1, \dots, x_n],$$
where $A \in N^{(n)}$ and $t[x_1, \dots, x_n] \in \mathbb{T}_{\Sigma,N}(X_n)$.

We say that G is of rank m if the rank of nonterminals of G does not exceed m.

The one-step rewriting relation $\Rightarrow_G$ on $\mathbb{T}_{\Sigma,N}$ is defined as follows: $u_1 \Rightarrow_G u_2$ if there is a 1-context $u[x_1] \in \mathbb{T}_{\Sigma,N}[X_1]$, a nonterminal $A \in N^{(n)}$, a production $A(x_1, \dots, x_n) \to t[x_1, \dots, x_n]$, and trees $t_1, \dots, t_n \in \mathbb{T}_{\Sigma,N}$ such that $u_1 = u[A(t_1, \dots, t_n)]$ and $u_2 = u[t[t_1, \dots, t_n]]$. The *language* of a context-free tree grammar G is[6]
$$L(G) = \{\, t \in \mathbb{T}_\Sigma \mid S \Rightarrow_G^* t \,\}.$$
We call elements of $L(G)$ *derived trees* of G.

We assume that when Σ contains a special symbol ε, it is always used to label a leaf node and is interpreted as the empty string. The *string language* of a context-free tree grammar G is defined to be
$$\{\, \mathbf{y}(t) \mid t \in L(G) \,\}.$$

[6] This is the *OI*, as opposed to *IO*, interpretation of the grammar [2].

Example 1. Here is a very simple example of a context-free tree grammar. Let $G = (N, \Sigma, P, S)$, where

$$\begin{aligned} N &= N^{(0)} \cup N^{(1)} = \{S, C\} \cup \{A, B\}, \\ \Sigma &= \{a, f, g\}, \end{aligned}$$

and P consists of the following productions:

$$\begin{aligned} S &\to A(a), \\ A(x_1) &\to A(g(C, x_1)), \\ A(x_1) &\to B(x_1), \\ C &\to a, \\ B(x_1) &\to x_1, \\ B(x_1) &\to B(f(a, x_1, x_1)). \end{aligned}$$

Some elements of $L(G)$ are:

$$\begin{aligned} &a, \\ &g(a, a), \\ &g(a, g(a, a)), \\ &f(a, a, a), \\ &f(a, g(a, a), g(a, a)), \\ &f(a, g(a, g(a, a)), g(a, g(a, a))), \\ &f(a, f(a, a, a), f(a, a, a)), \\ &f(a, f(a, g(a, a), g(a, a)), f(a, g(a, a), g(a, a))), \\ &f(a, f(a, g(a, g(a, a)), g(a, g(a, a))), f(a, g(a, g(a, a)), g(a, g(a, a)))). \end{aligned}$$

A context-free tree grammar is *simple* if for each production $A(x_1, \ldots, x_n) \to t[x_1, \ldots, x_n]$, the right-hand side $t[x_1, \ldots, x_n]$ is an n-context. *Monadic* simple context-free tree grammars, i.e., simple context-free tree grammars of rank 1, are, inessential details aside, the same as tree-adjoining grammars [17].

2.3 From Context-Free Tree Grammars to Indexed Grammars

In this section, we review Guessarian's [7] method (in slightly adapted form) of converting a context-free tree grammar G to an indexed grammar $\mathrm{Ind}(G)$ that generates the same string language.

We refer to a node of a tree by a "Dewey decimal notation" [18] or "Gorn address", which is a string of positive integers separated by dots ".". If t is a tree, define the *domain* of t, $\mathrm{dom}(t)$, by

$$\begin{aligned} \mathrm{dom}(a) &= \{\varepsilon\}, \\ \mathrm{dom}(f(t_1, \ldots, t_n)) &= \{\varepsilon\} \cup \{\, i.p \mid 1 \leq i \leq n, p \in \mathrm{dom}(t_i) \,\}. \end{aligned}$$

If $p \in \mathrm{dom}(t)$, the label of the node at p, written $\mathrm{lab}(t, p)$ is defined by

$$\mathrm{lab}(a, \varepsilon) = a, \qquad \mathrm{lab}(f(t_1, \dots, t_n), \varepsilon) = f,$$
$$\mathrm{lab}(f(t_1, \dots, t_n), i.p) = \mathrm{lab}(t_i, p).$$

Let $G = (N, \Sigma, P, S)$ be a context-free tree grammar. Let t_i be the right-hand side tree of the ith production in P. Let

$$N' = \{S'\} \cup \{\, (i, p) \mid 1 \leq i \leq |P| \text{ and } p \in \mathrm{dom}(t_i) \,\},$$
$$I = \{\, (i, p) \in N' \mid \mathrm{lab}(t_i, p) \in N \,\}.$$

Define the indexed grammar $\mathrm{Ind}(G) = (N', \Sigma, I, P', S')$, where P' consists of the following productions:

- If the left-hand side of the ith production is S, then P' contains the production
$$S'[] \to (i, \varepsilon)[]. \tag{DIST$_1$}$$
- If $\mathrm{lab}(t_i, p) \in \Sigma$ and $n = \max\{\, j \mid p.j \in \mathrm{dom}(t_i) \,\}$, then P' contains the production
$$(i, p)[] \to (i, p.1)[] \dots (i, p.n)[]. \tag{DIST$_2$}$$
- If p is a leaf of t_i and $\mathrm{lab}(t_i, p) = a \in \Sigma$, then P' contains the production
$$(i, p)[] \to a. \tag{TERM}$$
- If $\mathrm{lab}(t_i, p) = A \in N$ and the left-hand side nonterminal of the jth production is A, then P' contains the production
$$(i, p)[] \to (j, \varepsilon)[(i, p)]. \tag{PUSH}$$
and the production
$$(j, q)[(i, p)] \to (i, p.k)[] \tag{POP}$$
for each $q \in \mathrm{dom}(t_j)$ such that $\mathrm{lab}(t_j, q) = x_k$.

If τ is a derivation tree of $\mathrm{Ind}(G)$, then let $h(\tau)$ be the result of removing all unary-branching nodes sanctioned by (TERM), (DIST$_1$), (PUSH), or (POP) productions, and then changing the label of each remaining internal node from $(i, p)[\chi]$ to $\mathrm{lab}(t_i, p)$. Clearly, $\mathbf{y}(\tau) = \mathbf{y}(h(\tau))$. We can prove the following:

Proposition 2. *If τ is a complete derivation tree of* $\mathrm{Ind}(G)$*, then $h(\tau)$ is a derived tree of G. Conversely, if t is a derived tree of G, then there exists a complete derivation tree τ of* $\mathrm{Ind}(G)$ *such that $t = h(\tau)$.*

Corollary 3. *For every context-free tree grammar G, $\{\, \mathbf{y}(t) \mid t \in L(G) \,\} = L(\mathrm{Ind}(G))$.*

We need two new notions to prove this proposition.[7] For an indexed grammar, a *derivation tree fragment* is defined like a derivation tree except that labels of the form $A[\chi]$ are allowed on leaf nodes. For a context-free tree grammar $G = (N, \Sigma, P, S)$, we extend the rewriting relation $\Rightarrow_G^*$ to $\mathbb{T}_{N \cup \Sigma}(X_n)$ in an obvious way.

Example 4. The result of applying the method to the context-free tree grammar G of Example 1 is the following indexed grammar $\mathrm{Ind}(G)$:

$$\begin{aligned}
S'[] &\to (1,\varepsilon)[] \\
(1,\varepsilon)[] &\to (i,\varepsilon)[(1,\varepsilon)] \quad (i = 2,3) \\
(1,1)[] &\to a, \\
(2,\varepsilon)[] &\to (i,\varepsilon)[(2,\varepsilon)] \quad (i = 2,3) \\
(2,1)[] &\to (2,1.1)[]\,(2,1.2)[] \\
(2,1.1)[] &\to (4,\varepsilon)[(2,1.1)] \\
(2,1.2)[(i,\varepsilon)] &\to (i,1)[] \quad (i = 1,2) \\
(3,\varepsilon)[] &\to (i,\varepsilon)[(3,\varepsilon)] \quad (i = 5,6) \\
(3,1)[(i,\varepsilon)] &\to (i,1)[] \quad (i = 1,2) \\
(4,\varepsilon)[] &\to a \\
(5,\varepsilon)[(i,\varepsilon)] &\to (i,1)[] \quad (i = 3,6) \\
(6,\varepsilon)[] &\to (i,\varepsilon)[(6,\varepsilon)] \quad (i = 3,6) \\
(6,1)[] &\to (6,1.1)[]\,(6,1.2)[]\,(6,1.3)[] \\
(6,1.1)[] &\to a \\
(6,1.2)[(i,\varepsilon)] &\to (i,1)[] \quad (i = 3,6) \\
(6,1.3)[(i,\varepsilon)] &\to (i,1)[] \quad (i = 3,6)
\end{aligned}$$

Fig. 1 shows two derivation trees of $\mathrm{Ind}(G)$ corresponding to the derived trees $g(a,a)$ and $f(a, g(a,a), g(a,a))$ of G.

3 Linear Indexed Grammars and an Alternative Conception of Indexed Grammars

3.1 Linear Indexed Grammars

Gazdar [4] introduced *linear indexed grammars*,[8] which are a variant of indexed grammars where the stack attached to an internal node is passed to *exactly one* of its children, except when the stack is empty, in which case it may be a (TERM) node. A linear indexed grammar is a quintuple $G = (N, \Sigma, I, P, S)$ just

[7] Due to space limitations, I had to leave out all proofs.

[8] "Linear indexed grammar" seems to be a coinage of Vijay-Shanker [29].

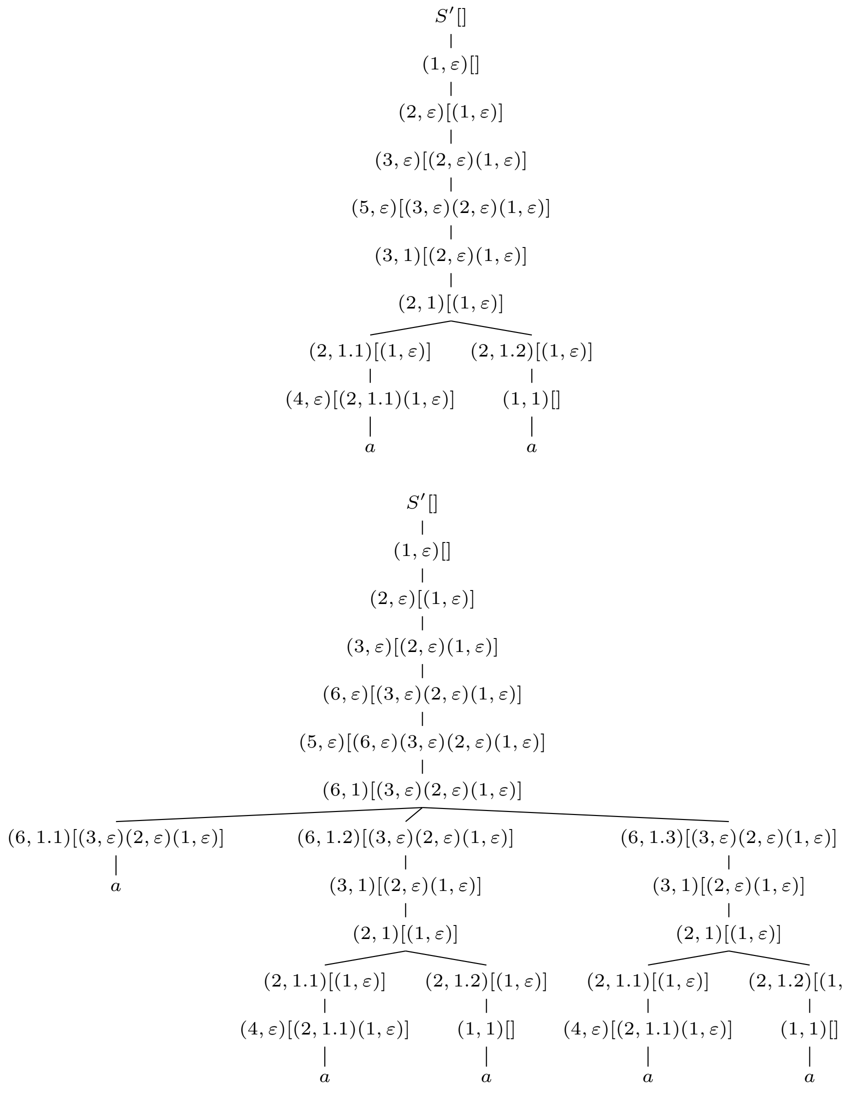
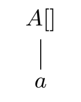
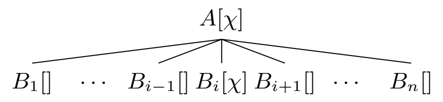
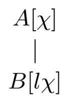
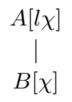

$S'[]$
$(1,\varepsilon)[]$
$(2,\varepsilon)[(1,\varepsilon)]$
$(3,\varepsilon)[(2,\varepsilon)(1,\varepsilon)]$
$(5,\varepsilon)[(3,\varepsilon)(2,\varepsilon)(1,\varepsilon)]$
$(3,1)[(2,\varepsilon)(1,\varepsilon)]$
$(2,1)[(1,\varepsilon)]$
$(2,1.1)[(1,\varepsilon)]$
$(2,1.2)[(1,\varepsilon)]$
$(4,\varepsilon)[(2,1.1)(1,\varepsilon)]$
$(1,1)[]$
a
a
$S'[]$
$(1,\varepsilon)[]$
$(2,\varepsilon)[(1,\varepsilon)]$
$(3,\varepsilon)[(2,\varepsilon)(1,\varepsilon)]$
$(6,\varepsilon)[(3,\varepsilon)(2,\varepsilon)(1,\varepsilon)]$
$(5,\varepsilon)[(6,\varepsilon)(3,\varepsilon)(2,\varepsilon)(1,\varepsilon)]$
$(6,1)[(3,\varepsilon)(2,\varepsilon)(1,\varepsilon)]$
$(6,1.1)[(3,\varepsilon)(2,\varepsilon)(1,\varepsilon)]$
$(6,1.2)[(3,\varepsilon)(2,\varepsilon)(1,\varepsilon)]$
$(6,1.3)[(3,\varepsilon)(2,\varepsilon)(1,\varepsilon)]$
a
$(3,1)[(2,\varepsilon)(1,\varepsilon)]$
$(3,1)[(2,\varepsilon)(1,\varepsilon)]$
$(2,1)[(1,\varepsilon)]$
$(2,1)[(1,\varepsilon)]$
$(2,1.1)[(1,\varepsilon)]$
$(2,1.2)[(1,\varepsilon)]$
$(2,1.1)[(1,\varepsilon)]$
$(2,1.2)[(1,\varepsilon)]$
$(4,\varepsilon)[(2,1.1)(1,\varepsilon)]$
$(1,1)[]$
$(4,\varepsilon)[(2,1.1)(1,\varepsilon)]$
$(1,1)[]$
a
a
a
a

Fig. 1. Derivation trees of an indexed grammar obtained from a context-free tree grammar.

Table 2. Interpretation of linear indexed grammar productions ($\chi \in I^*$).

(TERM)	(DIST′)	(PUSH)	(POP)
$A[] \to a$	$A[\circ\circ] \to B_1[] \dots B_{i-1}[]\, B_i[\circ\circ]\, B_{i+1}[] \dots B_n[]$	$A[\circ\circ] \to B[l\circ\circ]$	$A[l\circ\circ] \to B[\circ\circ]$
			

like an indexed grammar except that each production takes one of the following forms:[9]

$$A[] \to a, \qquad \text{(TERM)}$$
$$A[\circ\circ] \to B_1[] \dots B_{i-1}[]\, B_i[\circ\circ]\, B_{i+1}[] \dots B_n[], \qquad \text{(DIST}'\text{)}$$
$$A[\circ\circ] \to B[l\circ\circ], \qquad \text{(PUSH)}$$
$$A[l\circ\circ] \to B[\circ\circ], \qquad \text{(POP)}$$

where $A, B, B_1, \dots, B_n \in N, n \geq 1, l \in I, a \in \Sigma \cup \{\varepsilon\}$. The expression $\circ\circ$ serves as a variable ranging over the strings of indices and serves to indicate which of the children of a node the stack gets passed to. In linear indexed grammars, an occurrence of "[]" in a production indicates empty stack, rather than a variable stack, as in the case of indexed grammars. Thus, a (TERM) production can only sanction a node with empty stack, and all but one of the children of a node sanctioned by a (DIST′) production must have empty stack. The (PUSH) and (POP) productions are interpreted exactly like the productions of indexed grammars of the same name, except for the notation. See Table 2.

The definition of the language generated by a grammar is as before:

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

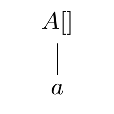
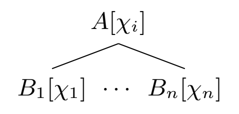
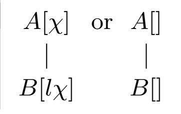
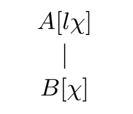
3.2 A Bottom-Up Conception of Indexed Grammar Derivation Trees

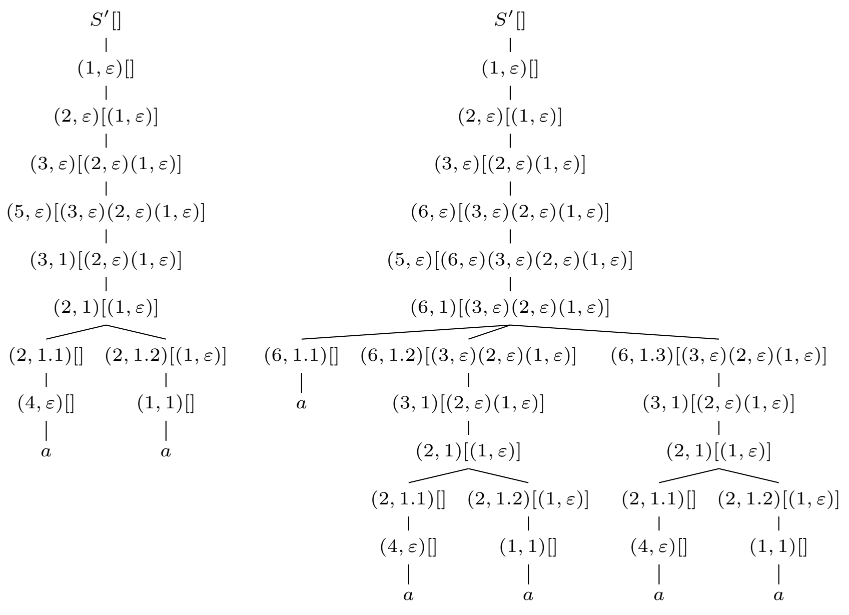
Because of the difference in how the stack works in linear indexed grammars, a derivation tree of a linear indexed grammar is often *not* a possible derivation tree of an indexed grammar. However, there is an alternative view of indexed grammars that brings the two formalisms closer together.

Note that the labels of nodes in a derivation tree of a linear indexed grammar may be determined both top-down and bottom-up; once you know which production an internal node is sanctioned by, knowing its label uniquely determines its children's labels, and vice versa. Derivation trees of indexed grammars are deterministic only in the top-down direction, because when an internal node

[9] Again, this is a normal form for linear indexed grammars, which we adopt here for convenience. Note that some authors, e.g., [27], place the top of the stack at the right end, contrary to the original convention of Aho [1].

Table 3. Bottom-up interpretation of indexed grammar productions ($\chi, \chi_1, \dots, \chi_n \in I^*$). In (DIST), $\chi_1, \dots, \chi_n$ must be pairwise compatible, and $i = \text{argmax}_j\, |\chi_j|$.

(TERM)	(DIST)	(PUSH)	(POP)
$A[] \to a$	$A[] \to B_1[] \dots B_n[]$	$A[] \to B[l]$	$A[l] \to B[]$
			



$S'[]$
$(1,\varepsilon)[]$
$(2,\varepsilon)[(1,\varepsilon)]$
$(3,\varepsilon)[(2,\varepsilon)(1,\varepsilon)]$
$(5,\varepsilon)[(3,\varepsilon)(2,\varepsilon)(1,\varepsilon)]$
$(3,1)[(2,\varepsilon)(1,\varepsilon)]$
$(2,1)[(1,\varepsilon)]$
$(2,1.1)[]$
$(2,1.2)[(1,\varepsilon)]$
$(4,\varepsilon)[]$
$(1,1)[]$
a
a
$S'[]$
$(1,\varepsilon)[]$
$(2,\varepsilon)[(1,\varepsilon)]$
$(3,\varepsilon)[(2,\varepsilon)(1,\varepsilon)]$
$(6,\varepsilon)[(3,\varepsilon)(2,\varepsilon)(1,\varepsilon)]$
$(5,\varepsilon)[(6,\varepsilon)(3,\varepsilon)(2,\varepsilon)(1,\varepsilon)]$
$(6,1)[(3,\varepsilon)(2,\varepsilon)(1,\varepsilon)]$
$(6,1.1)[]$
$(6,1.2)[(3,\varepsilon)(2,\varepsilon)(1,\varepsilon)]$
$(6,1.3)[(3,\varepsilon)(2,\varepsilon)(1,\varepsilon)]$
a
$(3,1)[(2,\varepsilon)(1,\varepsilon)]$
$(3,1)[(2,\varepsilon)(1,\varepsilon)]$
$(2,1)[(1,\varepsilon)]$
$(2,1)[(1,\varepsilon)]$
$(2,1.1)[]$
$(2,1.2)[(1,\varepsilon)]$
$(2,1.1)[]$
$(2,1.2)[(1,\varepsilon)]$
$(4,\varepsilon)[]$
$(1,1)[]$
$(4,\varepsilon)[]$
$(1,1)[]$
a
a
a
a

Fig. 2. Derivation trees of an indexed grammar under the bottom-up conception.

is sanctioned by a (TERM) production, the label of its unique child does not determine the stack portion of its label.

We can, however, adopt an alternative interpretation of indexed grammar productions and construct derivation trees bottom-up. With this alternative conception, (TERM) productions of an indexed grammar are interpreted in exactly the same way as in linear indexed grammars: they sanction a node only when its stack is empty. (POP) productions are interpreted in the same way as before, but (DIST) and (PUSH) productions are reinterpreted, as indicated in Table 3 (we call two strings *compatible* if one of them is a prefix of the other). Note that there are two ways in which a node may be sanctioned by a (PUSH) production; the second case is for a (PUSH) node with no matching (POP) node.

Example 5. Fig. 2 shows the same derivation trees in Fig. 1 relabeled in the new bottom-up way.

Indexed grammar derivation trees in the new sense correspond one-to-one with derivation trees in the original sense. The derivation trees under the two conceptions differ only in the stack portion of the labels of internal nodes, and they give rise to the same notion of the generated language. Let us adopt this new, bottom-up conception from now on, since it allows us to view derivation trees of linear indexed grammars as derivation trees of indexed grammars of a special kind.

Let G be a linear indexed grammar, and let G' be the indexed grammar that is the result of erasing all occurrences of $\circ\circ$ from productions of G. Then every derivation tree of G (from $A[]$ for some nonterminal A) is a derivation tree of G' in which each (PUSH) node has exactly one matching (POP) node.

3.3 Monadic Indexed Grammars

Suppose we apply the method of Guessarian [7] reviewed in Section 2.3 to a monadic simple context-free tree grammar G. The resulting indexed grammar $\mathrm{Ind}(G)$ is very close to a linear indexed grammar. (For example, the grammar consisting of the first five productions of the grammar in Example 1 is a monadic simple context-free tree grammar, and the productions of the corresponding indexed grammar are the first 11 lines of the grammar in Example 4, with $i \neq 6$.) In any complete derivation tree of $\mathrm{Ind}(G)$, every (PUSH) node has *at most one* matching (POP) node. More precisely, a (PUSH) node has no matching (POP) node when the (PUSH) production sanctioning it is related to a nonterminal of G of rank 0; it has exactly one matching (POP) node when the production is related to a nonterminal of rank 1. Actually, it is easy to modify Guessarian's method and convert a monadic simple context-free tree grammar into a linear indexed grammar, instead of an indexed grammar.[10] However, the kind of indexed grammar that $\mathrm{Ind}(G)$ exemplifies is interesting in its own right.

Let us call a derivation tree of an indexed grammar *monadic* if every (PUSH) node in it has at most one matching (POP) node. If G is an indexed grammar, let us write $D(G)$ for the set of complete derivation trees of G and $D_1(G)$ for the set of monadic complete derivation trees of G. Define

$$L_1(G) = \{\, \mathbf{y}(\tau) \mid \tau \in D_1(G) \,\}.$$

(Recall $L(G) = \{\, \mathbf{y}(\tau) \mid \tau \in D(G) \,\}$.) We can prove the following:

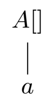
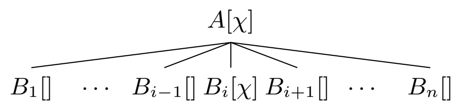
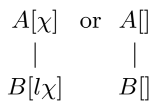
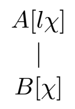
Proposition 6. *For every indexed grammar G, there is a linear indexed grammar G' such that $L_1(G) = L(G')$.*

Proposition 7. *For every linear indexed grammar G, there is an indexed grammar G' such that $L(G) = L_1(G') = L(G')$.*

Consider an indexed grammar G together with $D_1(G)$ and $L_1(G)$. This can be thought of as another variant of indexed grammar where (DIST) production

[10] A variant of this modification is given by Vijay-Shanker and Weir [28] for tree-adjoining grammars.

Table 4. Interpretation of monadic indexed grammar productions ($\chi \in I^*, 1 \leq i \leq n$).

(TERM)	(DIST)	(PUSH)	(POP)
$A[] \to a$	$A[] \to B_1[] \dots B_n[]$	$A[] \to B[l]$	$A[l] \to B[]$
			

$A[] \to B_1[] \dots B_n[]$ sanctions a node only if all but one of its children has empty stack. Let us call an indexed grammar with this interpretation a *monadic indexed grammar*. (See Table 4.) The indexed grammar obtained from a monadic simple context-free tree grammar by Guessarian's method can be regarded equivalently as a monadic indexed grammar. As Propositions 6 and 7 show, monadic indexed grammars are equivalent to linear indexed grammars, and the derivation trees of the two formalisms are also almost identical.

Vijay-Shanker and Weir [27] give a method of converting a linear indexed grammar to an equivalent head grammar. Combined with a conversion from head grammars to tree-adjoining grammars, it gives a method of converting a linear indexed grammar into a tree-adjoining grammar that generates the same string language. This result can be strengthened. For a (linear/monadic/general) indexed grammar G, let us call the result $\widehat{\tau}$ of erasing all indices from a (complete) derivation tree τ of G a *stripped (complete) derivation tree*. We can easily turn Vijay-Shanker and Weir's method into one that establishes the following:

Proposition 8. *For every linear or monadic indexed grammar G, there is a monadic simple context-free tree grammar G' that generates the set of all stripped complete derivation trees of G.*

I use monadic indexed grammars, rather than linear indexed grammars, as the point of departure for my generalization of linear indexed grammars. This is not strictly necessary, but will greatly simplify the definition of the generalized formalism.

4 Arboreal Indexed Grammars

Given Proposition 8, an obvious variant of indexed grammars corresponding to simple context-free tree grammars of rank m suggests itself: indexed grammars interpreted in such a way that all (PUSH) nodes must have at most m matching (POP) nodes.

4.1 From m-adic Indexed Grammars to Simple Context-Free Tree Grammars of Rank m

Let us call a derivation tree fragment of an indexed grammar *m-adic* if each (PUSH) node in it has at most m matching (POP) nodes. Write $D_m(G)$ for the set of m-adic complete derivation trees of an indexed grammar G, and let $L_m(G) = \{\, \mathbf{y}(\tau) \mid \tau \in D_m(G) \,\}$. It is easy to check the following:

Proposition 9. *If G is a simple context-free tree grammar of rank m, then $D(\mathrm{Ind}(G)) = D_m(\mathrm{Ind}(G))$, and consequently, $L(\mathrm{Ind}(G)) = L_m(\mathrm{Ind}(G))$.*

I now present a generalization of the construction underlying Proposition 8. Consider an indexed grammar G. A *path* in a derivation tree fragment of G is a sequence of nodes, always passing from a parent node to one of its children. We say that a path ρ is *clean* if every (PUSH) node on ρ is matched by a (POP) node on ρ.

Lemma 10. *If a (POP) node matches a (PUSH) node in a derivation tree fragment, the path from the child of the (PUSH) node to the (POP) node is a clean path.*

Lemma 11. *Let τ be a derivation tree fragment from $A[]$ such that $\mathbf{y}(\tau) = w_0\, B_1[]\, w_1 \dots B_n[]\, w_n$ for some $w_0, \dots, w_n \in \Sigma^*$ and for every $i = 1, \dots, n$, the path ρ_i from the root to the leaf node labeled by $B_i[]$ is a clean path. Let τ' be the result of changing the label of each node on $\rho_1, \dots, \rho_n$ from $C[\chi]$ to $C[\chi l]$. Then τ' is a derivation tree fragment from $A[l]$ with $\mathbf{y}(\tau') = w_0\, B_1[l]\, w_1 \dots B_n[l]\, w_n$.*

Let $m \geq 1$. Given an indexed grammar $G = (N, \Sigma, I, P, S)$, define a simple context-free tree grammar $\mathrm{CFT}^m_{\mathrm{sp}}(G) = (N', \Sigma', P', \langle S \rangle)$ where

$$N'^{(k)} = \begin{cases} \{\, \langle A B_1 \dots B_k \rangle \mid A, B_1, \dots, B_k \in N \,\} & \text{if } k \leq m, \\ \varnothing & \text{otherwise,} \end{cases}$$
$$\Sigma' = \Sigma \cup \{\varepsilon\} \cup N,$$

and P' consists of the following productions:

(A) If $A[] \to a$ is a (TERM) production in P, P' contains the production
$$\langle A \rangle \to A(a).$$

(B) For each nonterminal $A \in N$, P' contains the production
$$\langle AA \rangle(x_1) \to x_1.$$

(C) For each nonterminal $\langle A B_1 \dots B_k \rangle \in N'$, if $A[] \to C_1[] \dots C_n[]$ is a (DIST) production in P and $0 \leq k_1 \leq \dots \leq k_n = k$, then P' contains the production
$$\langle A B_1 \dots B_k \rangle(x_1, \dots, x_k) \to$$
$$A(\langle C_1 B_1 \dots B_{k_1} \rangle(x_1, \dots, x_{k_1}), \dots, \langle C_n B_{k_{n-1}+1} \dots B_{k_n} \rangle(x_{k_{n-1}+1}, \dots, x_{k_n})).$$

(D) For each nonterminal $\langle A B_1 \dots B_k \rangle \in N'$, if $A[] \to C[l]$ is a (PUSH) production in P, $D_1[l] \to E_1[], \dots, D_n[l] \to E_n[]$ are (POP) productions in P $(0 \leq n \leq m)$, and $0 \leq k_1 \leq \dots \leq k_n = k$, then P' contains the production
$$\langle A B_1 \dots B_k \rangle(x_1, \dots, x_k) \to$$

$$\begin{aligned} A(\langle C D_1 \dots D_n \rangle (D_1(\langle E_1 B_1 \dots B_{k_1} \rangle (x_1, \dots, x_{k_1})),& \\ \dots,& \\ D_n(\langle E_n B_{k_{n-1}+1} \dots B_{k_n} \rangle (x_{k_{n-1}+1}, \dots, x_{k_n})))).& \end{aligned}$$

(When $n = 0$, this production is $\langle A \rangle \to A(\langle C \rangle)$.)

Lemma 12. *Let $\langle A B_1 \dots B_k \rangle$ be a nonterminal of $\mathrm{CFT}^m_{\mathrm{sp}}(G)$. For every $t[x_1, \dots, x_k] \in \mathbb{T}_{\Sigma \cup \{\varepsilon\} \cup N}[X_k]$, the following are equivalent:*

(i) $\langle A B_1 \dots B_k \rangle (x_1, \dots, x_k) \Rightarrow^*_{\mathrm{CFT}^m_{\mathrm{sp}}(G)} t[x_1, \dots, x_k]$.

(ii) *There is an m-adic derivation tree fragment τ of G such that*
 - $t[B_1, \dots, B_k] = \widehat{\tau}$,
 - *the root of τ is labeled by $A[]$,*
 - $\mathbf{y}(\tau) = w_0\, B_1[]\, w_1 \dots B_k[]\, w_k$ *for some $w_0, w_1, \dots, w_k \in \Sigma^*$, and*
 - *for each $i = 1, \dots, k$, the path from the root of τ to the leaf node labeled by $B_i[]$ is a clean path.*

Theorem 13. *For every indexed grammar G, $L(\mathrm{CFT}^m_{\mathrm{sp}}(G)) = \{ \widehat{\tau} \mid \tau \in D_m(G) \}$.*

Let us call indexed grammars with the restriction to m-adic complete derivation trees *m-adic indexed grammars*. Proposition 9 and Theorem 13 establish the equivalence between m-adic indexed grammars and simple context-free tree grammars of rank m.

4.2 Storing Tuples of Trees in the Stack

Our job is not done yet. The notion of an m-adic indexed grammar has not been defined in terms of how the productions are interpreted. In the case of monadic indexed grammars, the restriction on the way a (DIST) production may sanction a node carved out precisely the set of monadic derivation trees. We cannot obtain the m-adic derivation trees in a similar way, for $m \geq 2$.

In order to express the restriction to m-adic derivation trees, we have to somehow record at each node the number of (POP) nodes below the node that are to match a given (PUSH) node above the node. This means that a (PUSH) production should push k copies of the same index ($k \leq m$) onto the stack, and a (DIST) production should distribute different copies of the same index to (possibly) different children.

Such stack actions cannot be realized with strings of indices acting as pushdown storage. The most natural solution is to store a *tuple of trees*, rather than a string, in the stack. We only need to store a tuple of trees $s_1, \dots, s_k$ with very special properties. First, all the nodes of $s_1, \dots, s_k$ of the same level (i.e., at the same distance from the root) must have the same label. Second, for m-adic indexed grammars, the number of nodes of $s_1, \dots, s_k$ of the same level may not exceed m. (This implies that the trees are at most m-branching; it does *not* imply that the number of leaves is bounded.) The number of components of the

Table 5. Interpretation of m-adic arboreal indexed grammar productions. Here, σ and σ_i range over (m-limited) tuples of trees over I. When σ is empty, $l(\sigma)$ stands for l. The concatenation of tuples $\sigma_1, \dots, \sigma_n$ is written as $\sigma_1 \dots \sigma_n$.

(TERM)	(DIST)	(PUSH)	(POP)
$A[] \to a$	$A[] \to B_1[] \dots B_n[]$	$A[] \to B[l]$	$A[l] \to B[]$
$\begin{array}{c} A[] \\ \mid \\ a \end{array}$	$\begin{array}{c} A[\sigma_1 \dots \sigma_n] \\ \diagup \quad \diagdown \\ B_1[\sigma_1] \ \cdots \ B_n[\sigma_n] \end{array}$	$\begin{array}{c} A[\sigma_1 \dots \sigma_k] \quad (k \le m) \\ \mid \\ B[l(\sigma_1), \dots, l(\sigma_k)] \end{array}$	$\begin{array}{c} A[l(\sigma)] \\ \mid \\ B[\sigma] \end{array}$

tuple may vary from node to node, but of course cannot exceed m. Let us call such a tuple of trees *m-limited.*

We consider trees in which each leaf node has a label from $\Sigma \cup \{\varepsilon\}$ and each internal node has a label of the form $A[s_1, \dots, s_k]$, where $s_1, \dots, s_k$ is an m-limited tuple of trees over I. Such trees may be sanctioned by productions of an indexed grammar as indicated in Table 5, in which case we call them *m-adic arboreal derivation trees.*

If $s_1, \dots, s_k$ is an m-limited tuple of trees over I, all the paths in $s_1, \dots, s_k$ starting from the root and ending in some leaf node give mutually compatible strings of indices. Let $\overline{s_1, \dots, s_k}$ be the string given by a maximal path. If υ is an m-adic arboreal derivation tree of an indexed grammar G, let $\overline{\upsilon}$ be the result of changing each label $C[s_1, \dots, s_k]$ in υ to $C[\overline{s_1, \dots, s_k}]$. Then $\overline{\upsilon}$ is always an ordinary derivation tree. It is easy to see the following:

Lemma 14. *Let G be an indexed grammar. An (ordinary) derivation tree τ of G is m-adic if and only if there is an m-adic arboreal derivation tree υ of G such that $\tau = \overline{\upsilon}$.*

Example 15. The second derivation tree in Fig. 2 was 2-adic. The 2-adic arboreal derivation tree corresponding to it is shown in Fig. 3, where we abbreviate indices (i, p) by i_p.

We call an indexed grammar together with the interpretation of productions given in Table 5 an *m-adic arboreal indexed grammar.* We have established the following:

Theorem 16. (i) *For every m-adic arboreal indexed grammar G, there is a simple context-free tree grammar G' of rank m such that the derived trees of G' are precisely the stripped complete derivation trees of G.*
(ii) *For every simple context-free tree grammar G' of rank m, there is an m-adic arboreal indexed grammar G such that the derived trees of G' are obtained from the stripped complete derivation trees of G by deleting some unary-branching nodes.*

Corollary 17. *Simple context-free tree grammars of rank m and m-adic arboreal indexed grammars are equivalent in string-generating power.*

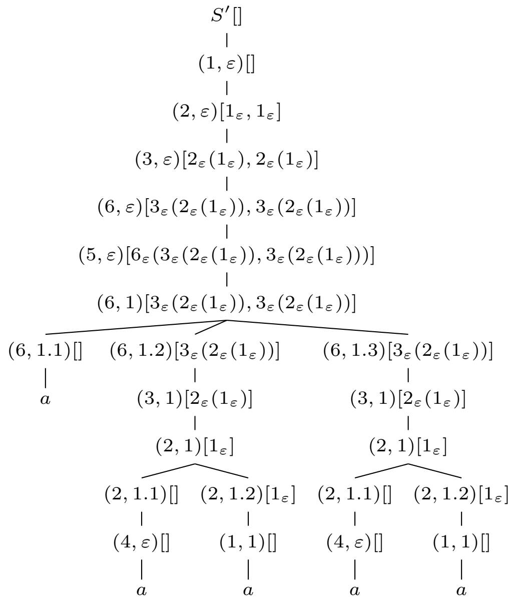

$S'[]$
$(1,\varepsilon)[]$
$(2,\varepsilon)[1_\varepsilon, 1_\varepsilon]$
$(3,\varepsilon)[2_\varepsilon(1_\varepsilon), 2_\varepsilon(1_\varepsilon)]$
$(6,\varepsilon)[3_\varepsilon(2_\varepsilon(1_\varepsilon)), 3_\varepsilon(2_\varepsilon(1_\varepsilon))]$
$(5,\varepsilon)[6_\varepsilon(3_\varepsilon(2_\varepsilon(1_\varepsilon)), 3_\varepsilon(2_\varepsilon(1_\varepsilon)))]$
$(6,1)[3_\varepsilon(2_\varepsilon(1_\varepsilon)), 3_\varepsilon(2_\varepsilon(1_\varepsilon))]$
$(6,1.1)[]$
$(6,1.2)[3_\varepsilon(2_\varepsilon(1_\varepsilon))]$
$(6,1.3)[3_\varepsilon(2_\varepsilon(1_\varepsilon))]$
a
$(3,1)[2_\varepsilon(1_\varepsilon)]$
$(3,1)[2_\varepsilon(1_\varepsilon)]$
$(2,1)[1_\varepsilon]$
$(2,1)[1_\varepsilon]$
$(2,1.1)[]$
$(2,1.2)[1_\varepsilon]$
$(2,1.1)[]$
$(2,1.2)[1_\varepsilon]$
$(4,\varepsilon)[]$
$(1,1)[]$
$(4,\varepsilon)[]$
$(1,1)[]$
a
a
a
a

Fig. 3. An example of a 2-adic arboreal derivation tree.

References

1. Aho, A.V.: Indexed grammars—an extension of context-free grammars. Journal of the Association for Computing Machinery 15(4), 647–671 (1968)
2. Engelfriet, J., Schmidt, E.M.: IO and OI, part I. The Journal of Computer and System Sciences 15(3), 328–353 (1977)
3. Engelfriet, J., Heyker, L.: The string generating power of context-free hypergraph grammars. Journal of Computer and System Sciences 43(2), 328–360 (1991)
4. Gazdar, G.: Applicability of indexed grammars to natural languages. In: Reyle, U., Rohrer, C. (eds.) Natural Language Parsing and Linguistic Theories, pp. 69–94. Reidel, Dordrecht (1988)
5. Gómez-Rodríguez, C., Kuhlmann, M., Satta, G.: Efficient parsing of well-nested linear context-free rewriting systems. In: Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL). pp. 276–284. Los Angeles, USA (2010)
6. de Groote, P., Pogodalla, S.: On the expressive power of abstract categorial grammars: Representing context-free formalisms. Journal of Logic, Language and Information 13(4), 421–438 (2004)
7. Guessarian, I.: Pushdown tree automata. Mathematical Systems Theory 16(1), 237–263 (1983)
8. Harkema, H.: A characterization of minimalist languages. In: Groote, P., Morrill, G., Retoré, C. (eds.) Logical Aspects of Computational Linguistics, Lecture Notes in Computer Science, vol. 2099, pp. 193–211. Springer Berlin Heidelberg (2001)
9. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, Massachusetts (1979)

10. Hotz, G., Pitsch, G.: On parsing coupled-context-free languages. Thoretical Computer Science 161(1–2), 205–253 (1996)
11. Joshi, A.K., Vijay-Shanker, K., Weir, D.: The convergence of mildly context-sensitive grammar formalisms. In: Sells, P., Shieber, S., Wasow, T. (eds.) Processing of Linguistic Structure, pp. 31–81. MIT Press, Cambridge, Massachusetts (1991)
12. Kanazawa, M.: The convergence of well-nested mildly context-sensitive grammar formalisms (July 2009), http://research.nii.ac.jp/~kanazawa/talks/fg2009_talk.pdf, an invited talk given at the 14th Conference on Formal Grammar, Bordeaux, France.
13. Kanazawa, M.: The pumping lemma for well-nested multiple context-free languages. In: Diekert, V., Nowotka, D. (eds.) Developments in Language Theory: 13th International Conference, DLT 2009. pp. 312–325. Springer, Berlin (2009)
14. Kanazawa, M.: Second-order abstract categorial grammars (2009), http://research.nii.ac.jp/~kanazawa/publications/esslli2009_lectures.pdf, lecture notes for a course taught at ESSLLI 2009.
15. Kanazawa, M.: Multi-dimensional trees and a Chomsky-Schützenberger-Weir representation theorem for simple context-free tree grammars. Journal of Logic and Computation (to appear)
16. Kepser, S., Mönnich, U.: Closure properties of linear context-free tree languages with an application to optimality theory. Theoretical Computer Science 354(1), 82–97 (2006)
17. Kepser, S., Rogers, J.: The equivalence of tree adjoining grammars and monadic linear context-free tree grammars. Journal of Logic, Language and Information 20(3), 361–384 (2011)
18. Knuth, D.E.: The Art of Computer Programming, Vol. I: Fundamental Algorithms. Addison-Wesley, Reading, Mass., third edn. (1997)
19. Maletti, A., Engelfriet, J.: Strong lexicalization of tree adjoining grammars. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics. pp. 506–515. Association for Computational Linguistics (2012)
20. Michaelis, J.: Derivational minimalism is mildly contextsensitive. In: Moortgat, M. (ed.) Logical Aspects of Computational Linguistics, Lecture Notes in Computer Science, vol. 2014, pp. 179–198. Springer Berlin Heidelberg (2001)
21. Michaelis, J.: Transforming linear context-free rewriting systems into minimalist grammars. In: Groote, P., Morrill, G., Retoré, C. (eds.) Logical Aspects of Computational Linguistics, Lecture Notes in Computer Science, vol. 2099, pp. 228–244. Springer Berlin Heidelberg (2001)
22. Rambow, O., Satta, G.: Independent parallelism in finite copying parallel rewriting systems. Theoretical Computer Science 223(1–2), 87–120 (1999)
23. Rounds, W.: Mappings and grammars on trees. Mathematical Systems Theory 4(3), 257–287 (1970)
24. Salvati, S.: Encoding second order string ACG with deterministic tree walking transducers. In: Wintner, S. (ed.) Proceedings of FG 2006: The 11th conference on Formal Grammar. pp. 143–156. FG Online Proceedings, CSLI Publications (2007)
25. Seki, H., Kato, Y.: On the generative power of multiple context-free grammars and macro grammars. IEICE Transactions on Information and Systems E91–D(2), 209–221 (2008)
26. Steedman, M.: The Syntactic Process. MIT Press, Cambridge, Massachusetts (2000)
27. Vijay-Shanker, K., Weir, D.J.: The equivalence of four extensions of context-free grammars. Mathematical Systems Theory 27(6), 511–546 (1994)

28. Vijay-Shanker, K., Weir, D.J.: Parsing some constrained grammar formalisms. Computational Linguistics 19(4), 591–636 (1993)
29. Vijayashanker, K.: A Study of Tree Adjoining Grammars. Ph.D. thesis, University of Pennsylvania (1987)
30. Weir, D.J.: Characterizing Mildly Context-Sensitive Grammar Formalisms. Ph.D. thesis, University of Pennsylvania (1988)
31. Weir, D.J.: Linear context-free rewriting systems and deterministic tree-walking transducers. In: Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics. pp. 136–143 (1992)