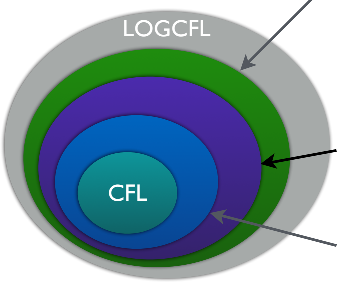
1

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

Makoto Kanazawa
National Institute of Informatics, Tokyo, Japan

2

Convergence of well-nested mildly context-sensitive grammar formalisms



MCTAG ≡ MCFG ≡ HR ≡ OUT(DTWT) ≡ yDT_{fc}(REGT) ≡ LUSCG ≡ MG ≡ $ACG_{(2,4)}$

$yCFT_{sp}$ ≡ $ACG_{(2,3)}$ ≡ $MCFG_{wn}$ ≡ CCFG

TAG ≡ $yCFT_{sp}(1)$ ≡ HG ≡ LIG ≡ CCG

This was the topic of my invited talk at FG 2009.

All formalisms for the intermediate level are essentially the same.

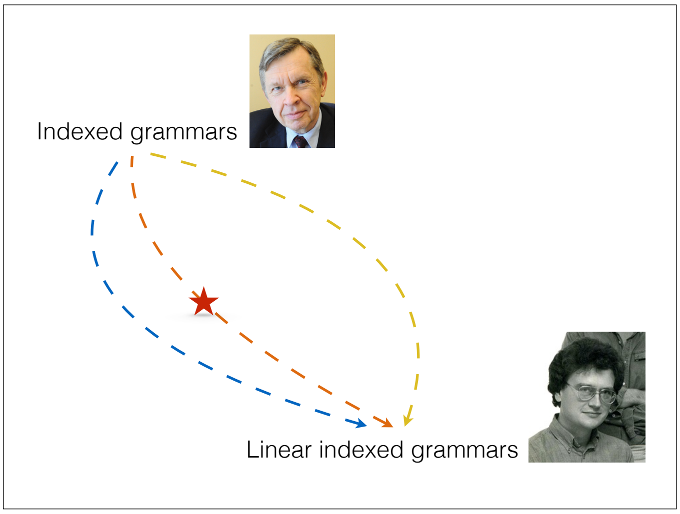
An analogue of LIG equivalent to $yCFT_{sp}(m)$?

3

indexed grammars	≡	OI context-free tree grammars	Fischer 1968
?	≡	simple context-free tree grammars of rank *m* CFT	
linear indexed grammars	≡	tree-adjoining grammars (monadic simple context-free tree grammars)	Vijay-Shanker and Weir 1994

We will define a new type of indexed grammars that is equivalent to simple context-free grammars of rank *m*.

4



We want to stop in the middle of a journey from indexed grammars to linear indexed grammars, but there is no clear path from the former to the latter.

5

Indexed grammars

$G = (N, \Sigma, I, P, S)$

indices

Productions

$A[] \to a$	$(a \in \Sigma \cup \{\varepsilon\})$	(TERM)
$A[] \to B_1[] \ldots B_n[]$	$(n \geq 1)$	(DIST)
$A[] \to B[f]$		(PUSH)
$A[f] \to B[]$		(POP)

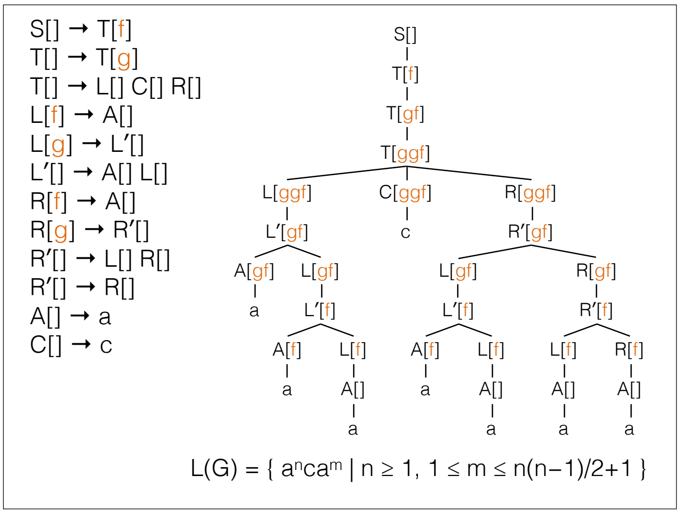
This is a normal form of indexed grammars that is more general than "reduced form" of Aho 1968.

6

Derivation trees of IG

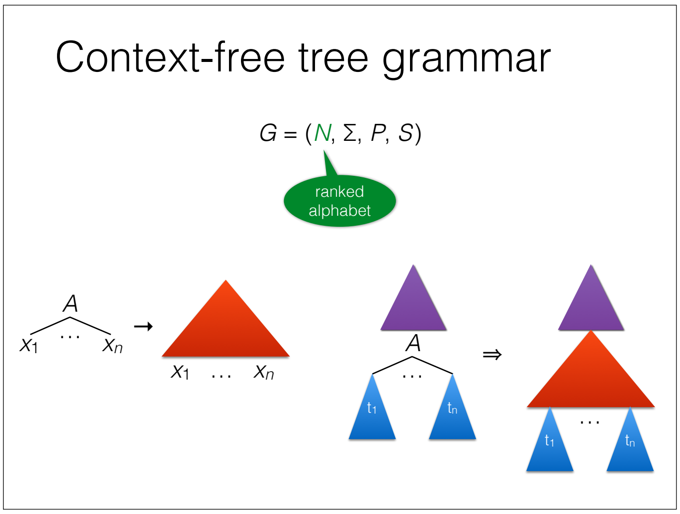
(TERM)	$A[] \to a$	$A[\chi]$ — a
(DIST)	$A[] \to B_1[] \ldots B_n[]$	$A[\chi]$ — $B_1[\chi] \ldots B_n[\chi]$
(PUSH)	$A[] \to B[f]$	$A[\chi]$ — $B[f\chi]$
(POP)	$A[f] \to B[]$	$A[f\chi]$ — $B[\chi]$

In derivation trees, a nonterminal occurs with stack of indices attached to it.
Each node of a derivation tree is sanctioned by a production.



7

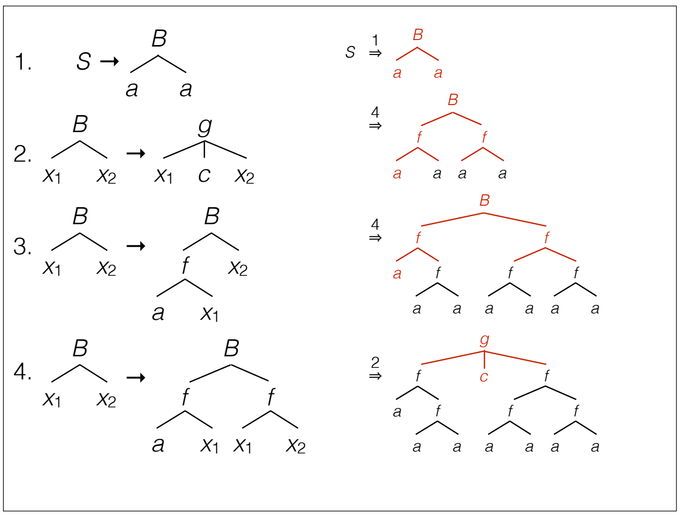
Example of an indexed grammar.



8

Tree rewriting system.

A rank n nonterminal labels a node with n children.

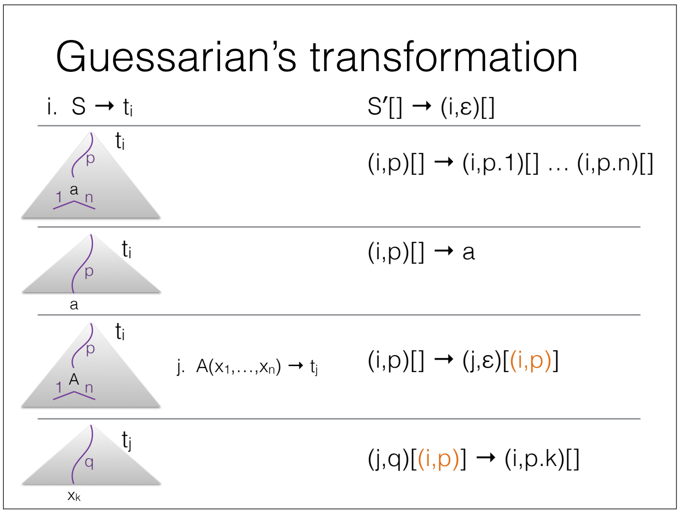


9

Example of a derivation. The red portion is where the tree grew.

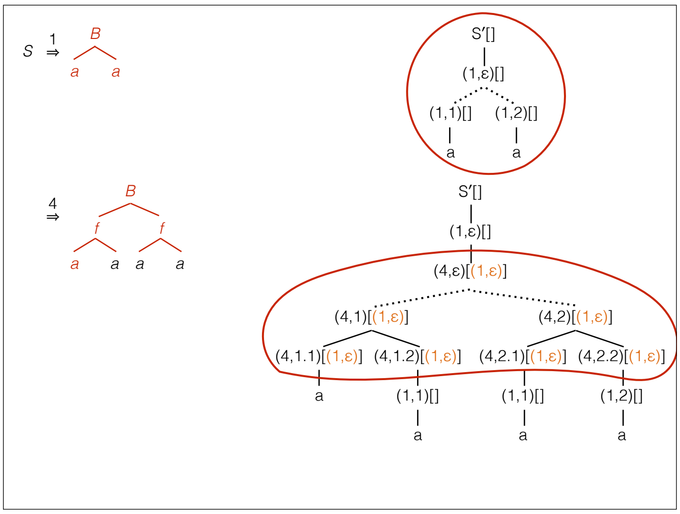
$y(L(G)) = \{ a^n \ c \ a^m \mid n \geq 1, 1 \leq m \leq n(n-1)/2 + 1 \}$

Note that production 4 duplicates x_1.



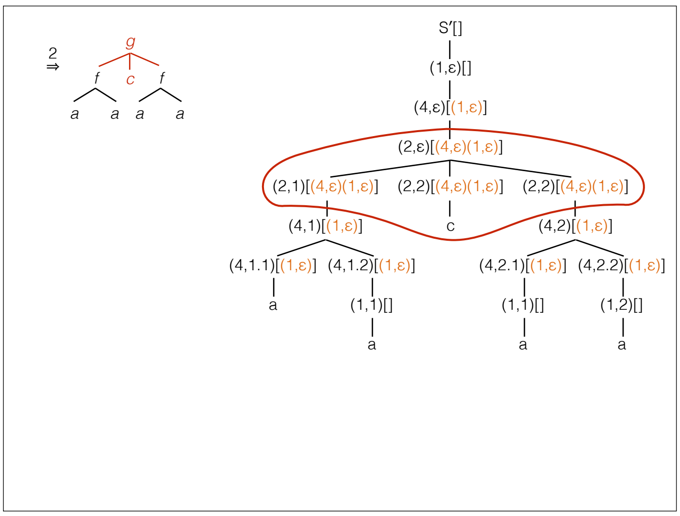
10

Nonterminals and indices are both of the form (i,p), where i is a production number and p is the address of a node in the right-hand side tree of the *i*-th production.

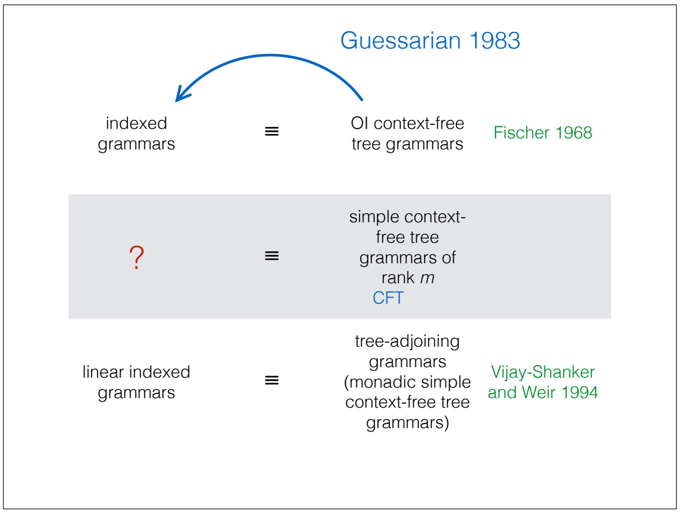


11

Can build derivation tree in accordance with derivation in CFTG.



12

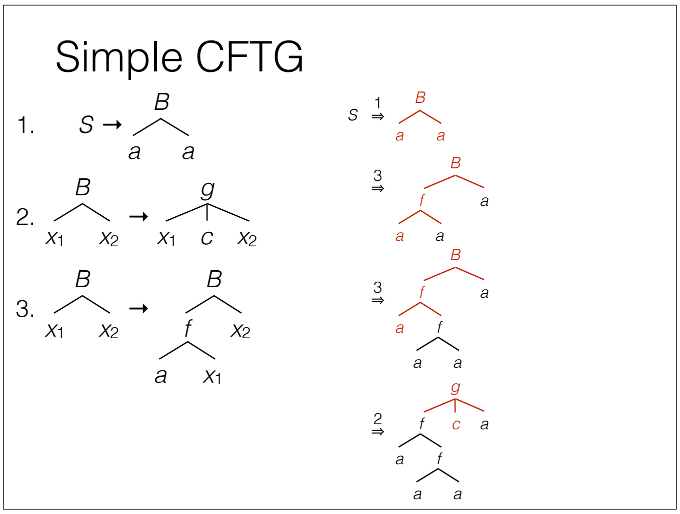


13

Guessarian's transformation: OI CFT → Ind

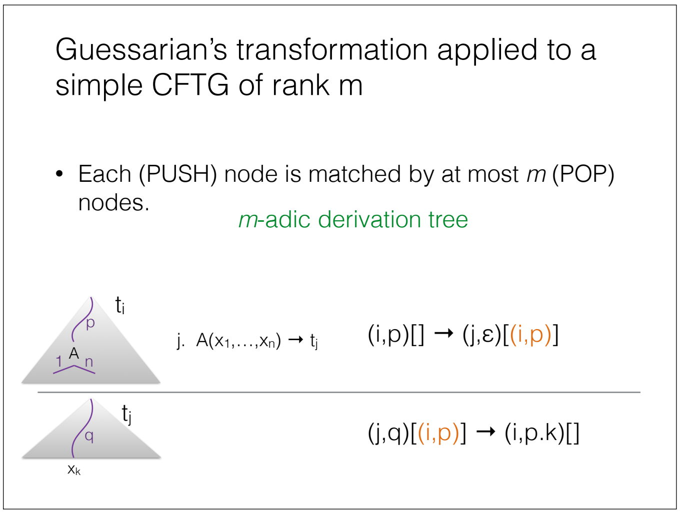
Can apply the same transformation to CFTsp(m).

What do we get?



14

"Simple" means no variable is deleted or duplicated by any production.



15

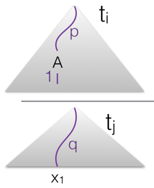
We call a derivation tree where each (PUSH) node is matched by at most m (POP) nodes an m-adic derivation tree.

("m-copying derivation tree" might be better?)

Guessarian's transformation applied to a monadic simple CFTG

- Each (PUSH) node is matched by at most one (POP) node.

monadic derivation tree



j. $A(x_1) \to t_j$ $\qquad$ $(i,p)[] \to (j,\varepsilon)[(i,p)]$

$(j,q)[(i,p)] \to (i,p.1)[]$

16

"non-copying" derivation tree.

Output of Guessarian's transformation on $CFT_{sp}(m)$

Guessarian's IG

$$H \in CFT_{sp}(m) \Rightarrow D(Ind(H)) = D_m(Ind(H))$$

set of m-adic derivation trees

17

Every derivation tree of Ind(H) is *m*-adic.

Converse transformation for $CFT_{sp}(m)$

$$G \in Ind \Rightarrow \exists\, H \in CFT_{sp}(m)\ (L(H) = \{\ strip(\tau) \mid \tau \in D_m(G)\ \})$$

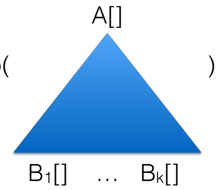
result of stripping indices

$H = CFT_{sp}{}^m(G)$

18

Transformation from indexed grammars to simple context-tree grammars of rank *m*.
Generalizes the construction used by Vijay-Shanker and Weir.

$H = CFT_{sp}^{m}(G)$

$\langle AB_1\ldots B_k\rangle(x_1 \ldots x_k) \Rightarrow_H^{*} t[x_1,\ldots,x_k] = \mathrm{strip}($  $)$

m-adic
derivation tree
fragment of G

19

$<AB_1\ldots B_k>$ is a nonterminal of rank $k \le m$.

G

$A[] \rightarrow a$

$A[] \rightarrow C_1[] \ldots C_n[]$

$H = CFT_{sp}^{m}(G)$

$\langle A\rangle \rightarrow A(a)$

$\langle AB_1\ldots B_k\rangle(x_1 \ldots x_k) \rightarrow A(\langle C_1B_1\ldots B_{k_1}\rangle(x_1 \ldots x_{k_1}) \ldots \langle C_nB_{k_{n-1}+1}\ldots B_k\rangle(x_{k_{n-1}+1} \ldots x_k))$

$0 \le k_1 \le \ldots \le k_{n-1} \le k$

20

$A[] \rightarrow C[f]$
$D_1[f] \rightarrow E_1[]$
…
$D_n[f] \rightarrow E_n[]$

$\langle AB_1\ldots B_k\rangle(x_1 \ldots x_k) \rightarrow A(\langle CD_1\ldots D_n\rangle(D_1(\langle E_1B_1\ldots B_{k_1}\rangle(x_1 \ldots x_{k_1})) \ldots D_n(\langle E_nB_{k_{n-1}+1}\ldots B_k\rangle(x_{k_{n-1}+1} \ldots x_k))))$

$0 \le k_1 \le \ldots \le k_{n-1} \le k$

21

22

Guessarian's IG

$H \in \mathrm{CFT}_{sp}(m) \Rightarrow D(\mathrm{Ind}(H)) = D_m(\mathrm{Ind}(H))$

set of m-adic derivation trees

converse transformation

$G \in \mathrm{Ind} \Rightarrow L(\mathrm{CFT}_{sp}{}^{m}(G)) = \{ \mathrm{strip}(\tau) \mid \tau \in D_m(G) \}$

result of stripping indices

Simple context-free tree grammars of rank m are equivalent to a special class of indexed grammars whose derivation trees are all *m*-adic.

Also, they correspond to indexed grammars coupled with a "global" restriction on derivation trees.

We can change how the productions of indexed grammars

23

Standard conception of IG

(TERM)	$A[] \rightarrow a$	$A[\chi]$ | a
(DIST)	$A[] \rightarrow B_1[] \ldots B_n[]$	$A[\chi]$ $B_1[\chi] \;\cdots\; B_n[\chi]$
(PUSH)	$A[] \rightarrow B[f]$	$A[\chi]$ | $B[f\chi]$
(POP)	$A[f] \rightarrow B[]$	$A[f\chi]$ | $B[\chi]$

How can one capture the "m-copying" property by restricting possible applications of productions?

The standard conception of IG is top-down: the stack of a parent plus the production being applied determines the stacks of the children.

(TERM) empties the stack.

24

Bottom-up conception of IG

(TERM)	$A[] \rightarrow a$	$A[]$ | a
(DIST)	$A[] \rightarrow B_1[] \ldots B_n[]$	$A[\chi_i]$ $B_1[\chi_1] \;\cdots\; B_n[\chi_n]$ χ_i, χ_k compatible, $i = \mathrm{argmax}_j \lvert\chi_j\rvert$
(PUSH)	$A[] \rightarrow B[f]$	$A[\chi]$ | $B[f\chi]$ or $A[]$ | $B[]$
(POP)	$A[f] \rightarrow B[]$	$A[f\chi]$ | $B[\chi]$

Need to switch to bottom-up view.

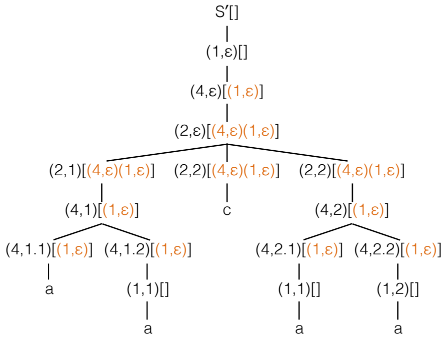
(TERM) requires the stack of the parent node to be empty.

(DIST) merges the stacks of the children nodes.

(PUSH) pops an index when the stack is not empty.

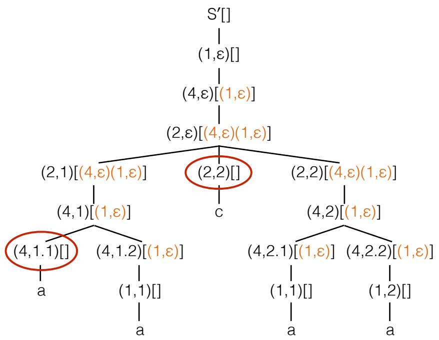
Standard, top-down derivation tree

25



Bottom-up derivation tree

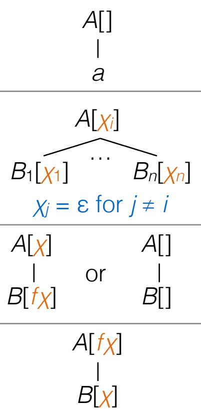
26



Monadic indexed grammar

27

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



Under the bottom-up conception, the restriction on how (DIST) productions may be used precisely carves out monadic derivation trees.

monadic IG ≈ linear IG

$D_1(G) = D(G^{mon})$

G interpreted as monadic IG

28

Monadic indexed grammars are very close to linear indexed grammars.
The difference is that in linear indexed grammars, a (DIST) production specifies which child inherits the stack of the parent.

$D_m(G) = D(G^{(m)})$?

G interpreted as m-adic IG

(DIST) $A[] \rightarrow B_1[] \ldots B_n[]$

$A[\chi_i]$

$B_1[\chi_1] \quad \cdots \quad B_n[\chi_n]$

$|\{ j \mid \chi_j \neq \varepsilon \}| \leq m$

29

How can one define "m-adic indexed grammars" by restricting how productions may be used in the bottom-up conception of IG?
Simply restricting (DIST) will not do.

Arboreal indexed grammars

- The stack stores a tuple of trees $[s_1,\ldots,s_n]$.
- A (PUSH) production pops the same index from all trees in the tuple.

30

We need yet another conception of indexed grammars.
Viewed from the top down, (PUSH) pushes k copies of an index.

Arboreal indexed grammar

(TERM)	$A[] \to a$	$A[]$ / a
(DIST)	$A[] \to B_1[] \dots B_n[]$	$A[\sigma_1 \dots \sigma_n]$ / $B_1[\sigma_1] \; \dots \; B_n[\sigma_n]$
(PUSH)	$A[] \to B[f]$	$A[\sigma_1 \dots \sigma_k]$ / $B[f(\sigma_1),\dots,f(\sigma_k)]$
(POP)	$A[f] \to B[]$	$A[f(\sigma)]$ / $B[\sigma]$

31

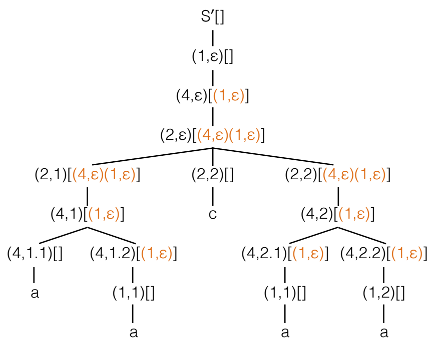
σ, σ_i are possibly empty tuples of trees.

Interpretation of productions bottom-up deterministic.

Viewed from the top down, a (PUSH) node anticipates how many (POP) nodes match it, and push that many copies of the same index onto the stack.

With no restriction on (PUSH),

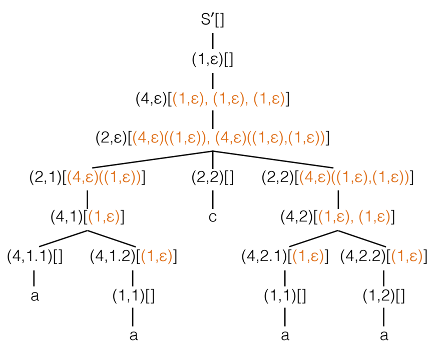
Bottom-up derivation tree



32

Viewed bottom-up, (DIST) merges indices.

Arboreal derivation tree



33

There is no merging of indices in arboreal derivation trees.

Taking the longest path in tuple of trees gives bottom-up linear derivation tree.

34

m-adic arboreal indexed grammar

(TERM)	$A[\,] \to a$	$\begin{array}{c} A[\,] \\ \vert \\ a \end{array}$
(DIST)	$A[\,] \to B_1[\,] \ldots B_n[\,]$	$\begin{array}{c} A[\sigma_1 \ldots \sigma_n] \\ B_1[\sigma_1] \quad \cdots \quad B_n[\sigma_n] \end{array}$
(PUSH)	$A[\,] \to B[f]$	$\begin{array}{c} A[\sigma_1 \ldots \sigma_k] \\ \vert \\ B[f(\sigma_1),\ldots,f(\sigma_k)] \\ \boldsymbol{k \leq m} \end{array}$
(POP)	$A[f] \to B[\,]$	$\begin{array}{c} A[f(\sigma)] \\ \vert \\ B[\sigma] \end{array}$

(PUSH) may pop (or push, viewed top-down) at most m copies of an index.

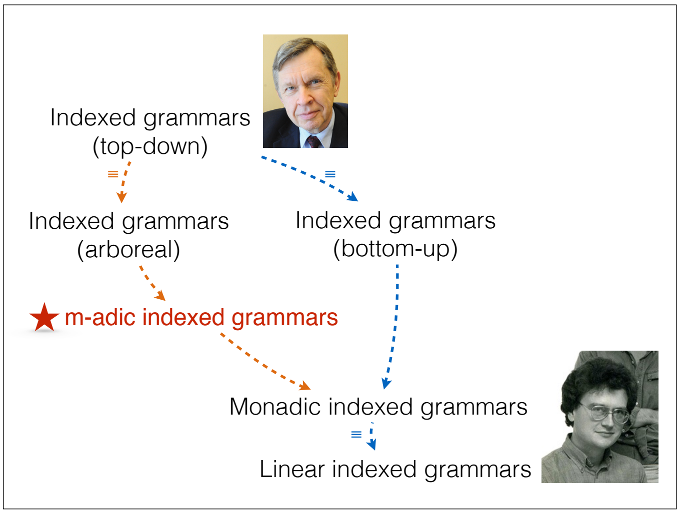
35

indexed grammars	≡	OI context-free tree grammars	Fischer 1968
m-adic arboreal indexed grammars	≡	simple context-free tree grammars of rank *m*	
linear indexed grammars	≡	tree-adjoining grammars (monadic simple context-free tree grammars)	Vijay-Shanker and Weir 1994

36

arboreal indexed grammars	≡	OI context-free tree grammars	Fischer 1968
m-adic arboreal indexed grammars	≡	simple context-free tree grammars of rank *m*	
monadic indexed grammars	≡	tree-adjoining grammars (monadic simple context-free tree grammars)	Vijay-Shanker and Weir 1994

Alternative conception of indexed grammar: arboreal.
Monadic indexed grammars are monadic arboreal indexed grammars if we view monadic trees as strings.


Indexed grammars
(top-down)
Indexed grammars
(arboreal)
Indexed grammars
(bottom-up)
m-adic indexed grammars
Monadic indexed grammars
Linear indexed grammars