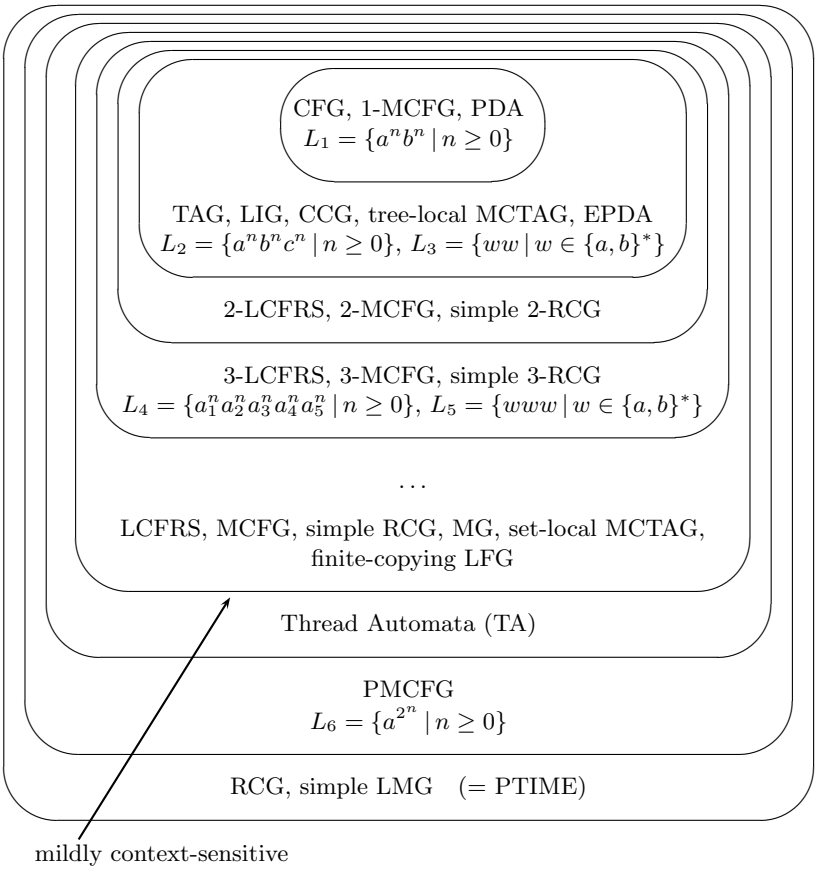
Appendix A: Hierarchy of Grammar Formalisms

The following figure recalls the language hierarchy that we have developed in the course of the book.



L. Kallmeyer, *Parsing Beyond Context-Free Grammars*, Cognitive Technologies,
DOI 10.1007/978-3-642-14846-0, © Springer-Verlag Berlin Heidelberg 2010

For each class the different formalisms and automata that generate/accept exactly the string languages contained in this class are listed. Furthermore, examples of typical languages for this class are added, i.e., of languages that belong to this class while not belonging to the next smaller class in our hierarchy. The inclusions are all proper inclusions, except for the relation between LCFRS and Thread Automata (TA). Here, we do not know whether the inclusion is a proper one. It is possible that both devices yield the same class of languages.

Appendix B: List of Acronyms

The following table lists all acronyms that occur in this book.

(2,2)-BRCG	Binary bottom-up non-erasing RCG with at most two variables per left-hand side argument
2-SA	Two-Stack Automaton
ACG	Abstract Categorial Grammar
AFL	Abstract Family of Languages
CCG	Combinatory Categorial Grammar
CFG	Context-Free Grammar
CNF	Chomsky Normal Form
EPDA	Extended Push-Down Automaton
FSA	Finite State Automaton
GCFG	Generalized Context-Free Grammar
GNF	Greibach Normal Form
HPSG	Head-Driven Phrase Structure Grammar
IG	Indexed Grammar
LCFRS	Linear Context-Free Rewriting System
LFG	Lexical Functional Grammar
LIG	Linear Indexed Grammar
LTAG	Lexicalized TAG
LMG	Literal Movement Grammar
MCFG	Multiple Context-Free Grammar
MCTAG	Multicomponent Tree Adjoining Grammar
MG	Minimalist Grammar
NRCG	Negative Range Concatenation Grammar
NPDA	Nested Push-Down Automaton
PDA	Push-Down Automaton
PMCFG	Parallel Multiple Context-Free Grammar
PRCG	Positive Range Concatenation Grammar
RCG	Range Concatenation Grammar
SD-2SA	Strongly-Driven Two-Stack Automaton

SNMCTAG	tree-local MCTAG with shared nodes
SRCG	Simple Range Concatenation Grammar
TA	Thread Automaton
TAG	Tree Adjoining Grammar
TSG	Tree Substitution Grammar
TT-MCTAG	Tree-Tuple MCTAG with Shared Nodes
V-TAG	Vector-TAG

Solutions

Problems of Chapter 2

2.1 $L_2 := \{a^n b^n \mid n \geq 0\}$

1. $G = \langle N, T, P, S\rangle$ with $N = \{S\}$, $T = \{a, b\}$, start symbol S and productions $S \to aSb$, $S \to \varepsilon$.
2. Assume that such a CFG exists. Its productions are then all of the form $X \to \alpha a \beta b \gamma$ with $X \in N$, $\alpha, \beta, \gamma \in N^*$ such that if such a production is applied when generating a string $a_1 \ldots a_n b_1 \ldots b_n$, then the a and b of the production necessarily end up at positions i and $n+i$ for some $i, 1 \leq i \leq n$. Then replacing each of these productions $X \to \alpha a \beta b \gamma$ with $X \to \alpha a \beta a \gamma$ and $X \to \alpha b \beta b \gamma$ leads to a CFG generating the copy language. This contradicts the fact that the copy language is not context-free. □

2.2 A first homomorphism can the homomorphism f from (Shieber, 1985).

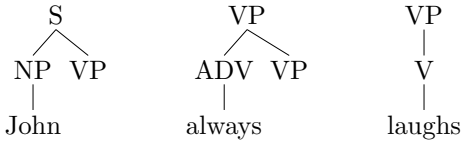
After having applied f to Swiss German, we intersect with the regular language $w\{a, b\}^* x \{c, d\}^* y$, which leads to $\{wv_1xv_2y \mid v_1 \in \{a, b\}^*, v_2 \in \{c, d\}^*$ such that $|v_1| = |v_2|$ and for all i, $1 \leq i \leq |v_1|$, if the ith symbol in v_1 is an a (a b), the ith symbol in v_2 is a c (a d)$\}$.

Finally we apply a second homomorphism g to the result of the intersection with $g(w) := g(x) := g(y) := \varepsilon, g(a) := g(c) := a, g(b) := g(d) := b$.

This leads to the copy language.

2.3

There are several possibilities. The simplest one:



Linguistically, this is unsatisfying since the S node comes with the lexical item *John* even though it is the maximal projection of the verb. A lexicalized TSG for the given CFG where the S node comes with the verb is not possible.

2.4

1. The copy language $L := \{ww \mid w \in T^*\}$ is letter equivalent to $L' := \{ww^R \mid w \in T^* \text{ and } w^R \text{ is } w \text{ in reverse order}\}$, which is a CFL: It is generated by the CFG with productions $S \to \varepsilon$ and $S \to xSx$ for all $x \in T$. Consequently (with Parikh's theorem) L' and also L are semilinear. □
2. Assume that $\{a^{2^n} \mid n \geq 0\}$ satisfies the constant growth property with c_0 and C. Then take a $w = a^{2^m}$ with $|w| = 2^m > max(\{c_0\} \cup C)$. Then, according to the definition of constant growth, for $w' = a^{2^{m+1}}$ there must be a $w'' = a^{2^k}$ with $|w'| = |w''| + c$ for some $c \in C$. I.e., $2^{m+1} = 2^k + c$. Consequently (since $k \leq m$) $c \geq 2^m$. Contradiction. □

Problems of Chapter 3

3.1

The items can have the form $[\bullet A, i, -]$, $0 \leq i \leq n$ (for predicted categories) and $[A\bullet, i, j]$, $0 \leq i < j \leq n$ (for completed categories). The goal item is $[S\bullet, 0, n]$. We need the following deduction rules:

1. An operation *scan-predict* that, starting from a predicted A-item, predicts a B-item, based on the existence of a production $A \to aB$:
$$\frac{[\bullet A, i, -]}{[\bullet B, i+1, -]} \quad \text{there is a production } A \to aB \in P \text{ with } w_{i+1} = a.$$
2. An operation *scan* that turns a predicted A into a completed A based on the existence of a production $A \to a$:
$$\frac{[\bullet A, i, -]}{[A\bullet, i, i+1]} \quad \text{there is a production } A \to a \in P \text{ with } w_{i+1} = a.$$
3. An operation *complete* that turns a predicted A into a completed A based on the existence of a completed B and a production $A \to aB$:
$$\frac{[\bullet A, i, -][B\bullet, i+1, j]}{[A\bullet, i, j]} \quad \text{there is a production } A \to aB \in P \text{ with } w_{i+1} = a.$$

3.2

To show: If $[A \to \alpha \bullet \beta, i, j]$ then $S \overset{*}{\Rightarrow} w_1 \cdots w_i A\gamma \Rightarrow w_1 \cdots w_i \alpha\beta\gamma \overset{*}{\Rightarrow} w_1 \cdots w_j \beta\gamma$ for some $\gamma \in (N \cup T)^*$.

We show this by an induction on the deduction rules:

- Axioms: $\dfrac{}{[S \to \bullet\alpha, 0, 0]} \quad S \to \alpha \in P$

  Trivially, if $[S \to \bullet\alpha, 0, 0]$ is obtained by this rule, then $S \overset{*}{\Rightarrow} \varepsilon S \Rightarrow \varepsilon\alpha$.

- Predict: $\dfrac{[A \to \alpha \bullet B\beta, i, j]}{[B \to \bullet\gamma, j, j]} \quad B \to \gamma \in P$

  We assume that the claim holds for the antecedent item. We then obtain (because of the production $B \to \gamma$) $S \overset{*}{\Rightarrow} w_1 \cdots w_i A\gamma' \Rightarrow w_1 \cdots w_i \alpha B\beta\gamma' \overset{*}{\Rightarrow} w_1 \cdots w_j B\beta\gamma' \Rightarrow w_1 \cdots w_j \gamma\gamma'$.
- Scan: $\dfrac{[A \to \alpha \bullet a\beta, i, j]}{[A \to \alpha a \bullet \beta, i, j+1]} \quad w_{j+1} = a$

  Since the claim holds for the antecedent item, with the side condition, it holds immediately for the consequent item.
- Complete: $\dfrac{[A \to \alpha \bullet B\beta, i, j], [B \to \gamma\bullet, j, k]}{[A \to \alpha B \bullet \beta, i, k]}$

  Since the induction claim holds for the antecedent items, we have in particular $B \overset{*}{\Rightarrow} w_{j+1} \cdots w_k$. With this and the claim for the antecedent A-item, we obtain $S \overset{*}{\Rightarrow} w_1 \cdots w_i A\gamma' \Rightarrow w_1 \cdots w_i \alpha B\beta\gamma' \overset{*}{\Rightarrow} w_1 \cdots w_j B\beta\gamma' \Rightarrow w_1 \cdots w_k \beta\gamma'$.

The algorithm is sound since, as a special case, we obtain for the goal items that $[S \to \alpha\bullet, 0, n]$ implies $S \Rightarrow \alpha \overset{*}{\Rightarrow} w_1 \cdots w_n$.

3.3

The space complexity of the CYK algorithm is determined by the memory requirement for the chart. Since this is an $|N| \times n \times n$ table, we obtain a space complexity $\mathcal{O}(n^2)$.

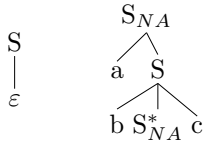
3.4

The most complex rule is **complete** with three indices ranging form 0 to n and with $|P|^2$ possible productions and $l_{rhs} = max_{A \to \alpha \in P}(|\alpha| + 1)$ possible positions for the dot in the antecedent A-items. Note that, once the A-production is fixed, the position of the dot determines the left-hand side symbol of the second production. Let $m_{rhs} = max_{A \in N}|\{A \to \alpha \in P\}|$. Then we have $\leq l_{rhs} \cdot |P| \cdot m_{rhs}(n+1)^3$ possible different applications of **complete**. Consequently, as in the CYK case, the time complexity of the fixed recognition problem is $\mathcal{O}(n^3)$.

Problems of Chapter 4

4.1

1. TAG for L_3:



2. Assume that a TAG G for L_3 without adjunction constraints exists. Assume without loss of generality that G contains no substitution nodes. G has at least one auxiliary tree β with leaves labeled with terminals. β must contain equal numbers of as, bs and cs. (Otherwise one could derive a word with different numbers of as, bs and cs.) One can adjoin β at its root, which leads to a derived auxiliary β'. If the yield of β is $a^i b^i c^i$ ($i \geq 1$), there are the following possibilities for the foot node:
   a) The foot node is left of all as or right of all cs $\Rightarrow$ using β' a word with substring $a^i b^i c^i a^i b^i c^i$ can be derived. Contradiction.
   b) The foot node is right of the kth a for some k, $1 \leq k \leq i \Rightarrow$ using β' a word with substring $a^{i-k} b^i c^i a^{i-k} b^i c^i$ can be derived. Contradiction.
   c) The foot node is right of the kth b for some k, $1 \leq k \leq i \Rightarrow$ using β' a word with substrings $a^i b^k a^i b^k$ and $b^{i-k} c^i b^{i-k} c^i$ can be derived. Contradiction.
   d) The foot node is left of the kth c for some k, $1 \leq k \leq i \Rightarrow$ using β' a word with substring $a^i b^i c^{k-1} a^i b^i c^{k-1}$ can be derived. Contradiction.

   Consequently, there is no TAG without adjunction constraints for L_3. □

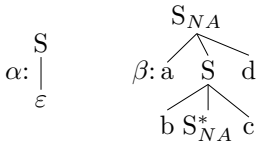
4.2 L_3 is a TAL $\Rightarrow$ with closure under concatenation $\{w_1 w_2 \,|\, w_1, w_2 \in L_3\} = \{a^n b^n c^n a^m b^m c^m \,|\, n, m \geq 0\}$ is a TAL. □

4.3 The copy language is a TAL and $a^* b^* a^* b^*$ is a regular language $\Rightarrow$ with closure under intersection with regular languages, $\{ww \,|\, w \in \{a, b\}^*\} \cap a^* b^* a^* b^* = \{a^i b^j a^i b^j \,|\, i, j \geq 0\}$ is a TAL. □

4.4 Assume that L is a TAL. Then the weak pumping lemma holds for some constant c. $\Rightarrow$ For each word $w \in L$ with $|w| \geq c$ there is a $w' \in L$ with $|w'| \leq |w| + c$. This is a contradiction since for all $w \in L$ with $|w| > c$ this is not the case (for each $w \in L$ the $w' \in L$ following it wrt word length has twice its length). □

4.5

1. TAG for L_4:

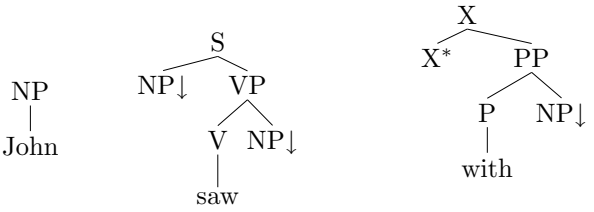


2. Assume that L_5 is a TAL and satisfies the weak pumping lemma with some constant c. Take $w = a^{c+1} b^{c+1} c^{c+1} d^{c+1} e^{c+1}$. According to the pumping lemma one can find $w_1, \ldots w_4$, at least one of them not empty, such that they can be inserted repeatedly at four positions into w yielding a new word in L_5. At least one of the $w_1, \ldots w_4$ must contain two different terminal symbols since they altogether must contain equal numbers of as, bs, cs, ds and es. Then, when doing a second insertion of the $w_1, \ldots w_4$,

the as, bs, cs, ds and es get mixed and the resulting word is not in L_5. Contradiction. □

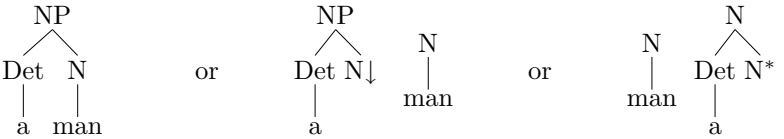
4.6

(30) John saw a man with a telescope

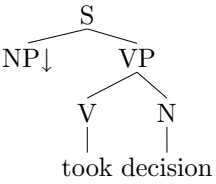


X=N or X=VP

Concerning the determiners, there are three possibilities:

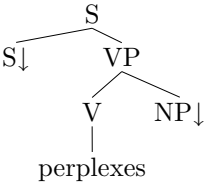


(31) Mary took a decision

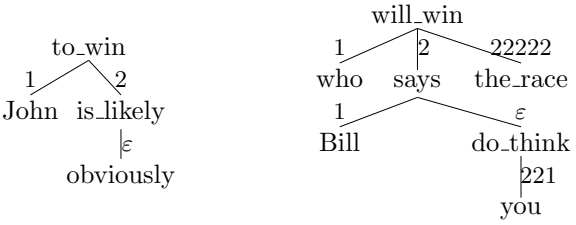


4.7

Since extraction out of sentential subjects is not allowed, it makes sense to add them by substitution. Otherwise one could adjoin them to a sentence with wh-extracted elements. (Of course such an adjunction could also be prevented with appropriate adjunction constraints.)



4.8



4.9

$S_0 \to S[\#]$
$S[..] \to aS_x[..]d \qquad S_x[..] \to S[x..]$
$S \to T \qquad T[x..] \to bT[..]c$
$T[\#] \to \varepsilon$

Problems of Chapter 5

5.1 Replace the deduction rules **move-unary** and **move-binary** with a single new rule **move-up**:

$$\frac{[\gamma, (p \cdot 1)_\top, i_0, f_{11}, f_{12}, i_1], \ldots, [\gamma, (p \cdot m)_\top, i_{m-1}, f_{m1}, f_{m2}, i_m]}{[\gamma, p_\bot, i_0, f_{11} \oplus \cdots \oplus f_{m1}, f_{12} \oplus \cdots \oplus f_{m2}, i_m]}$$

As a side condition, we require that the node address $p \cdot (m+1)$ does not exist in γ.

5.2

We keep the items $[\gamma, p_t, i, f_1, f_2, j]$ we had before (passive items) and add new active items $[\gamma, p, k, i, f_1, f_2, j]$ with

- γ an elementary tree, p a node position in γ, k the number of daughters of the node at p in γ that we have already seen;
- i, f_1, f_2, j as before.

The rules **move-unary** and **move-binary** are replaced with the following new rules:

1. A rule **left-corner predict** that, once we have seen the first daughter of a node, predicts the appropriate active item for its mother with $k = 1$.

   Left-corner predict: $\dfrac{[\gamma, (p \cdot 1)_\top, i, f_1, f_2, j]}{[\gamma, p, 1, i, f_1, f_2, j]}$

2. A rule **complete** that, from an item telling us that we have seen the first k daughters of a node and another item telling us that we have the passive item of the $(k+1)$th daughter, deduces the active item telling us that we have seen the $k+1$ daughters of the mother.

   Complete: $\dfrac{[\gamma, p, k, i_1, f_1, f_2, i_2][\gamma, (p \cdot (k+1))_\top, i_2, f_3, f_4, i_3]}{[\gamma, p, k+1, i_1, f_1 \oplus f_3, f_2 \oplus f_4, i_3]}$

3. A rule **convert** that converts an active item where we have seen all the daughters into a passive item.

   Convert: $\dfrac{[\gamma, p, k, i, f_1, f_2, j]}{[\gamma, p_\bot, i, f_1, f_2, j]}$ address $p \cdot (k+1)$ not defined in γ

Anything else remains as before.

5.3

In our items we have only three indices, i, j, k, where i and k delimit the total span of the relevant part of the tree (these were i and l in the original algorithm). j gives the start position of the part below the foot node for left auxiliary trees.

The *Scan* and *Predict* rules remain more or less the same except for the reduced number of indices:

ScanTerm $\dfrac{[\alpha, p, la, i, j, k, 0]}{[\alpha, p, ra, i, j, k+1, 0]}$ $\alpha(p) = w_{k+1}$

Scan-ε $\dfrac{[\alpha, p, la, i, j, k, 0]}{[\alpha, p, ra, i, j, k, 0]}$ $\alpha(p) = \varepsilon$

PredictAdjoinable $\dfrac{[\alpha, p, la, i, j, k, 0]}{[\beta, 0, la, k, -, k, 0]}$ $\beta \in f_{SA}(\alpha, p)$

PredictNoAdj $\dfrac{[\alpha, p, la, i, j, k, 0]}{[\alpha, p, lb, k, -, k, 0]}$ $f_{OA}(\alpha, p) = 0$

PredictAdjoined $\dfrac{[\beta, p, lb, k, -, k, 0]}{[\delta, p', lb, k, -, k, 0]}$ $p = foot(\beta), \beta \in f_{SA}(\delta, p')$

For the *Complete* rule, we obtain the following:

Complete

$$\frac{[\alpha, p, rb, i, j, k, 1][\beta, p', lb, i, -, i, 0]}{[\beta, p', rb, i, i, k, 0]} \quad p' = foot(\beta), \beta \in f_{SA}(\alpha, p)$$

Complete2 (remains the same)

$$\frac{[\beta, p, rb, i, j, k, sat?][\beta, p, la, h, -, i, 0]}{[\beta, p, ra, h, j, k, 0]} \quad \beta(p) \in N$$

For the *Adjoin* rule, we obtain the following:

Adjoin

$$\frac{[\beta, 0, ra, i, j, k, 0][\alpha, p, rb, j, l, k, 0]}{[\alpha, p, rb, i, l, k, 1]} \quad \beta \in f_{SA}(\alpha, p)$$

The *Move* rules and also the *Initialize* rule and the goal item remain the same, except for the reduced number of indices.

5.4

PredictSubstituted:

$$\frac{[\gamma, p, la, \sim, \sim, \sim, \sim, i, 0]}{[\alpha, \varepsilon, la, i, i, -, -, i, 0]} \quad \begin{array}{l}\alpha \in I, \gamma(p) \text{ substitution node,} \\ l(\gamma, p) = l(\alpha, \varepsilon)\end{array}$$

Substitute:

$$\frac{[\gamma, p, la, \sim, \sim, \sim, \sim, i, 0], [\alpha, \varepsilon, ra, i, i, -, -, j, 0]}{[\gamma, p, rb, \sim, i, -, -, j, 0]} \quad \begin{array}{l}\alpha \in I, \gamma(p) \text{ substitution node,} \\ l(\gamma, p) = l(\alpha, \varepsilon)\end{array}$$

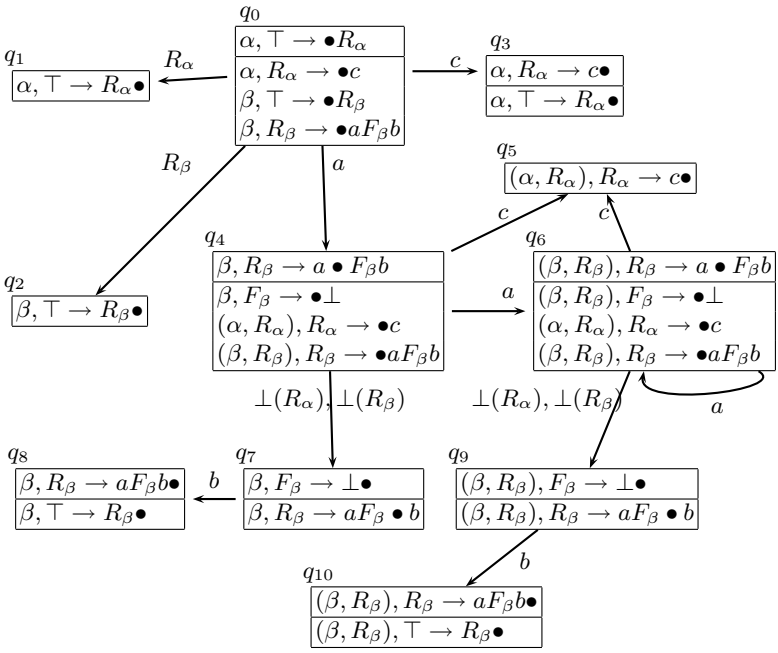
Note that adjunction is not allowed at substitution nodes; therefore the adjunction flag 0 in the consequent item does not cause any problems.

5.5

Multiple adjunctions are avoided since *reduce_subtree* is applied only if $X_1 \dots X_m \in CS^+(N)$, i.e., if the stack contains a cross-section of the node N that is not N itself. After having traversed the tree adjoined at N and having returned to the node N by applying *reduce_aux_tree*, it is the node N itself that is on the stack.

5.6

1.



2.

Stack	remaining input	operation
q_0	$aacbb$	*shift*
$q_0 a q_4$	$acbb$	*shift*
$q_0 a q_4 a q_6$	cbb	*shift*
$q_0 a q_4 a q_6 c q_5$	bb	*reduce_subtree*
$q_0 a q_4 a q_6 (\bot[R_\alpha]) q_9$	bb	*shift*
$q_0 a q_4 a q_6 (\bot[R_\alpha]) q_9 b q_{10}$	b	*reduce_subtree*
$q_0 a q_4 (\bot, [R_\beta, R_\alpha]) q_7$	b	*shift*
$q_0 a q_4 (\bot, [R_\beta, R_\alpha]) q_7 b q_8$	ε	*reduce_aux_tree*
$q_0 (R_\beta, [R_\alpha]) q_2$	ε	*reduce_aux_tree*
$q_0 R_\alpha q_1$	ε	accept

Problems of Chapter 6

6.1 There are of course different possibilities, for instance the following two MCFGs:

1. Rewriting rules: $S \to f_1[A] \quad A \to f_2[A] \quad A \to f_3[\,]$
   Operations:
   $f_1[\langle X, Y, Z, U\rangle] = \langle XYZU\rangle$
   $f_2[\langle X, Y, Z, U\rangle] = \langle aX, bY, cZ, dU\rangle$
   $f_3[\,] = \langle a, b, c, d\rangle$
2. Rewriting rules: $S \to f_1[A] \quad A \to f_2[A] \quad A \to f_3[\,]$
   Operations:
   $f_1[\langle X, Y\rangle] = \langle XY\rangle$
   $f_2[\langle X, Y\rangle] = \langle aXb, cYd\rangle$
   $f_3[\,] = \langle ab, cd\rangle$

6.2

Clauses:
$S(XYZ) \to A(Y)B(X, Z)$
$A(aX) \to A(X) \qquad A(a) \to \varepsilon$
$B(bX, bYb) \to B(X, Y) \qquad B(\varepsilon, \varepsilon) \to \varepsilon$

1. $yield(A) = \{\langle a^n\rangle \mid n \geq 1\}$
   $yield(B) = \{\langle b^n, (bb)^n\rangle \mid n \geq 0\}$.
2. $\{b^m a^n (bb)^m \mid n \geq 1, m \geq 0\}$.
3. For $w = abbb$,
   $r\text{-}yield(B) = \{(\langle i, i\rangle, \langle j, j\rangle) \mid 0 \leq i, j \leq 4\} \cup \{(\langle 1, 2\rangle, \langle 2, 4\rangle), (\langle 3, 4\rangle, \langle 1, 3\rangle)\}$.

6.3

1. $\{a^n b^n c^m d^m a^n b^n c^m d^m \mid n, m \geq 0\}$
2. $\{w_1 w_2 \mid w_1 \in \{a, b\}^*, w_2$ is the image of w_1 under the homomorphism f with $f(a) = b, f(b) = a\}$

6.4

1. Instantiations of $A(aX, bY) \to A(X, Y)$:
   $A(\langle 0, 1\rangle, \langle 1, 2\rangle) \to A(\langle 1, 1\rangle, \langle 2, 2\rangle)$
   $A(\langle 0, 1\rangle, \langle 2, 3\rangle) \to A(\langle 1, 1\rangle, \langle 3, 3\rangle)$
   $A(\langle 0, 2\rangle, \langle 2, 4\rangle) \to A(\langle 1, 2\rangle, \langle 3, 4\rangle)$
   $A(\langle 3, 4\rangle, \langle 1, 2\rangle) \to A(\langle 4, 4\rangle, \langle 2, 2\rangle)$
   $A(\langle 3, 4\rangle, \langle 2, 3\rangle) \to A(\langle 4, 4\rangle, \langle 3, 3\rangle)$
2. Derivation of $abba$:
   $S(\langle 0, 4\rangle) \Rightarrow A(\langle 0, 2\rangle, \langle 2, 4\rangle)$
   $\Rightarrow A(\langle 1, 2\rangle, \langle 3, 4\rangle)$
   $\Rightarrow A(\langle 3, 4\rangle, \langle 1, 2\rangle)$
   $\Rightarrow A(\langle 4, 4\rangle, \langle 2, 2\rangle)$
   $\Rightarrow \varepsilon$

6.5 Let $k > 0$ be fixed.

We assume that the language $L = \{w^{2k+1} \mid w \in \{a, b\}^*\}$ is a k-MCFL.

Then its intersection with the regular language $(a^+b^+)^{2k+1}$ must be a k-MCFL as well. This intersection yields $L' = \{(a^n b^m)^{2k+1} \mid n, m > 0\}$.

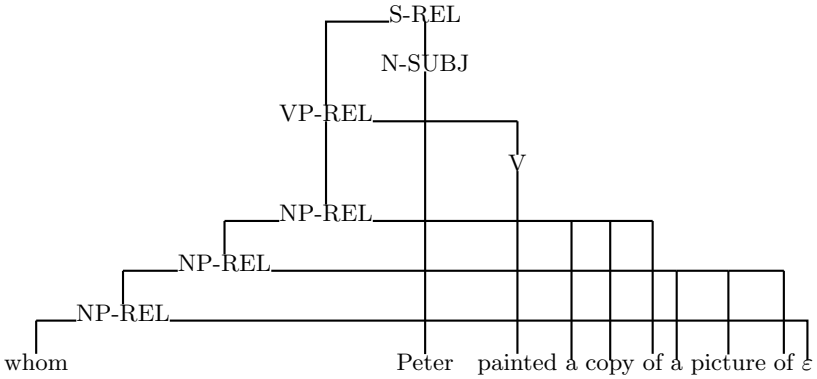
If L' is a k-MCFL, then it satisfies the pumping lemma for k-MCFLs. I.e., there must be a word $(a^n b^m)^{2k+1}$ for some $m, n > 0$ such that $2k$ substrings of this word can be iterated. At least one of these strings is not empty and none of them can contain different terminals, otherwise pumping would immediately lead to strings not in L'. However, if at most $2k$ strings are pumped, each of them containing only as or only bs, we necessarily obtain strings that are not in the language L'. This contradicts the assumption that L' satisfies the pumping lemma for k-MCFLs.

Consequently, neither L' nor L are k-MCFLs. □

6.6 The string language is the regular language

$$\textit{whom Peter painted ((a copy of) + (a picture of))}^*$$

For the string *whom Peter painted a copy of a picture of* (of length 9), we obtain the following derivation tree:



Problems of Chapter 7

7.1 The only unconstrained prediction concerns S predicates starting at position 0:

Initialize: $\dfrac{}{[S \to f[\mathbf{B}]; (R_1 = \langle 0, 0\rangle \bullet x); \mathbf{\Gamma_B}]}$ $\quad S \to f[\mathbf{B}] := (x)$

Predictions are now triggered by the dot preceding a component variable $B_k^{(i)}$:

Predict: $\dfrac{[A \to g[\mathbf{B}]; (\Phi_1, R_l = \alpha \bullet B_k^{(i)} x, \Phi_2); \mathbf{\Gamma}]}{[B_k \to f[\mathbf{C}]; (R_i = \langle j, j\rangle \bullet y, \Psi_1, \Psi_2); \mathbf{\Gamma_C}]}$ $\quad B_k \to f[\mathbf{C}] := (\Psi_1, y, \Psi_2)$ with y the ith element, j greatest range boundary in α

The same holds for **complete**:

Complete:

$$\frac{[A \to g[\mathbf{B}]; (\Phi_1, R_j = \alpha \bullet B_k^{(i)} x, \Phi_2); \mathbf{\Gamma}], \quad [B_k \to f[\mathbf{C}]; (\Psi, R_l = \langle l_l, r_l \rangle \bullet, \Psi_1, y, \Psi_2); \mathbf{\Gamma}']}{[B_k \to f[\mathbf{C}]; (\Psi, R_l = \langle l_l, r_l \rangle, R_i = \langle m, m \rangle \bullet y, \Psi_1, \Psi_2); \mathbf{\Gamma}']}$$

y is the ith element in the range constraint vector of $B_k \to f[\mathbf{C}]$, m is the greatest range boundary in α

The other operations and the goal item are the same as in the incremental algorithm with unconstrained prediction.

7.2

$S(XYZU) \to A(X,Z)B(U,Y)$ $S(XYZ) \to A(X,Z)C(Y)$
$A(aX,aZ) \to A(X,Z)$ $A(\varepsilon,c) \to \varepsilon$
$B(Xb,Yb) \to B(X,Y)$ $B(\varepsilon,c) \to \varepsilon$
$C(aXY) \to D(X)C(Y)$ $D(d) \to \varepsilon$

1. Simplifying the grammar:
   a) Transform the grammar into an ordered simple RCG. (If the superscript is the identity, we omit it.)
   The only problematic rule is $S(XYZU) \to A(X,Z)B(U,Y)$. It transforms into $S(XYZU) \to A(X,Z)B^{\langle 2,1\rangle}(Y,U)$.
   Add $B^{\langle 2,1\rangle}(Yb,Xb) \to B(X,Y)$ and $B^{\langle 2,1\rangle}(c,\varepsilon) \to \varepsilon$.
   Then, $B^{\langle 2,1\rangle}(Yb,Xb) \to B(X,Y)$ transforms into $B^{\langle 2,1\rangle}(Yb,Xb) \to B^{\langle 2,1\rangle}(Y,X)$.
   In the following, for reasons of readability, we replace $B^{\langle 2,1\rangle}$ with a new symbol E.
   Result:
   $S(XYZU) \to A(X,Z)E(Y,U)$ $S(XYZ) \to A(X,Z)C(Y)$
   $A(aX,aZ) \to A(X,Z)$ $A(\varepsilon,c) \to \varepsilon$
   $B(Xb,Yb) \to B(X,Y)$ $B(\varepsilon,c) \to \varepsilon$
   $E(Yb,Xb) \to E(Y,X)$ $E(c,\varepsilon) \to \varepsilon$
   $C(aXY) \to D(X)C(Y)$ $D(d) \to \varepsilon$
   b) Remove useless rules.
   - $N_T = \{A, B, E, D, S\}$.
     Consequently, remove $S(XYZ) \to A(X,Z)C(Y)$ and $C(aXY) \to D(X)C(Y)$.
   - In the result, $N_S = \{S, A, E\}$.
     Consequently, remove also $D(d) \to \varepsilon$, $B(Xb,Yb) \to B(X,Y)$ and $B(\varepsilon,c) \to \varepsilon$.

   Result:
   $S(XYZU) \to A(X,Z)E(Y,U)$
   $A(aX,aZ) \to A(X,Z)$ $A(\varepsilon,c) \to \varepsilon$
   $E(Yb,Xb) \to E(Y,X)$ $E(c,\varepsilon) \to \varepsilon$
   c) Remove ε-rules.
   $N_\varepsilon = \{A^{01}, A^{11}, E^{10}, E^{11}, S^1\}$.
   Resulting productions:

$S^1(XYZU) \to A^{11}(X,Z)E^{11}(Y,U)$ $S^1(YZU) \to A^{01}(Z)E^{11}(Y,U)$
$S^1(XYZ) \to A^{11}(X,Z)E^{10}(Y)$ $S^1(YZ) \to A^{01}(Z)E^{10}(Y)$
$A^{11}(aX,aZ) \to A^{11}(X,Z)$ $A^{11}(a,aZ) \to A^{01}(Z)$
$A^{01}(c) \to \varepsilon$
$E^{11}(Yb,Xb) \to E^{11}(Y,X)$ $E^{11}(Yb,b) \to E^{10}(Y)$
$E^{10}(c) \to \varepsilon$

2. The string language generated by this grammar is

$$\{a^n c b^m a^n c b^m \mid n,m \geq 0\}.$$

7.3

We have to consider the maximal number of possible applications of the complete rule.

$$\textbf{Complete}: \frac{[B,\boldsymbol{\rho_B}],[C,\boldsymbol{\rho_C}]}{[A,\boldsymbol{\rho_A}]} \quad \begin{array}{l} A(\boldsymbol{\rho_A}) \to B(\boldsymbol{\rho_B})C(\boldsymbol{\rho_C}) \\ \text{is an instantiation of a } c \in P \text{ wrt. } w \end{array}$$

Since the maximal arity is k, we have maximal $2k$ range boundaries in each of the antecedent items of this rule. For variables X_1, X_2 being in the same left-hand side argument of the clause c, X_1 left of X_2 and no other variables in between, the right boundary of X_1 gives us immediately the left boundary of X_2. In the worst case, A, B, C all have arity k and each left-hand side argument contains only two variables. This leads to $3k$ independent range boundaries and consequently a time complexity of $\mathcal{O}(n^{3k})$ for the entire algorithm.

7.4

The antecedent of the **Suspend** operation

$$\frac{[B(\boldsymbol{\psi}) \to \boldsymbol{\Psi}, pos', \langle i,j\rangle, \boldsymbol{\rho}_B], [A(\boldsymbol{\phi}) \to \ldots B(\boldsymbol{\xi}) \ldots, pos, \langle k,l\rangle, \boldsymbol{\rho}_A]}{[A(\boldsymbol{\phi}) \to \ldots B(\boldsymbol{\xi}) \ldots, pos', \langle k,l+1\rangle, \boldsymbol{\rho}]}$$

contains two active items, the A-item and B-item.

Since our grammar is ε-free, all daughters (here the B-item) have yields with non-empty components. Therefore the position of the B-item is greater than the one of the A-item, $pos' > pos$, in a **suspend** operation. Consequently, the A-item is always added first to the chart and it is therefore sufficient to trigger **suspend** operations by the B-items.

Problems of Chapter 8

8.1 Clauses:

$S(XY) \to A(X,Y)B(X,Y)$
$A(aX,aY) \to A(X,Y)$ $B(bX,bY) \to B(X,Y)$
$A(bX,Y) \to A(X,Y)$ $B(aX,Y) \to B(X,Y)$
$A(X,bY) \to A(X,Y)$ $B(X,aY) \to B(X,Y)$
$A(\varepsilon,\varepsilon) \to \varepsilon$ $B(\varepsilon,\varepsilon) \to \varepsilon$

Input: $w = abba$

Instantiated clauses used:

	instantiation	clause
1.	$S(\langle 0,4\rangle) \to A(\langle 0,2\rangle,\langle 2,4\rangle)B(\langle 0,2\rangle,\langle 2,4\rangle)$	$S(XY) \to A(X,Y)B(X,Y)$
2.	$A(\langle 0,2\rangle,\langle 2,4\rangle) \to A(\langle 0,2\rangle,\langle 3,4\rangle)$	$A(X,bY) \to A(X,Y)$
3.	$A(\langle 0,2\rangle,\langle 3,4\rangle) \to A(\langle 1,2\rangle,\langle 4,4\rangle)$	$A(aX,aY) \to A(X,Y)$
4.	$A(\langle 1,2\rangle,\langle 4,4\rangle) \to A(\langle 2,2\rangle,\langle 4,4\rangle)$	$A(bX,Y) \to A(X,Y)$
5.	$A(\langle 2,2\rangle,\langle 4,4\rangle) \to \varepsilon$	$A(\varepsilon,\varepsilon) \to \varepsilon$
6.	$B(\langle 0,2\rangle,\langle 2,4\rangle) \to B(\langle 1,2\rangle,\langle 2,4\rangle)$	$B(aX,Y) \to B(X,Y)$
7.	$B(\langle 1,2\rangle,\langle 2,4\rangle) \to B(\langle 2,2\rangle,\langle 3,4\rangle)$	$B(bX,bY) \to B(X,Y)$
8.	$B(\langle 2,2\rangle,\langle 3,4\rangle) \to B(\langle 2,2\rangle,\langle 4,4\rangle)$	$B(X,aY) \to B(X,Y)$
9.	$B(\langle 2,2\rangle,\langle 4,4\rangle) \to \varepsilon$	$B(\varepsilon,\varepsilon) \to \varepsilon$

Derivation:

$$
\begin{aligned}
S(\langle 0,6\rangle) &\Rightarrow A(\langle 0,2\rangle,\langle 2,4\rangle)B(\langle 0,2\rangle,\langle 2,4\rangle) && \text{(with 1.)}\\
&\Rightarrow A(\langle 0,2\rangle,\langle 3,4\rangle)B(\langle 0,2\rangle,\langle 2,4\rangle) && \text{(with 2.)}\\
&\Rightarrow A(\langle 1,2\rangle,\langle 4,4\rangle)B(\langle 0,2\rangle,\langle 2,4\rangle) && \text{(with 3.)}\\
&\Rightarrow A(\langle 2,2\rangle,\langle 4,4\rangle)B(\langle 0,2\rangle,\langle 2,4\rangle) && \text{(with 4.)}\\
&\Rightarrow B(\langle 0,2\rangle,\langle 2,4\rangle) && \text{(with 5.)}\\
&\Rightarrow B(\langle 1,2\rangle,\langle 2,4\rangle) && \text{(with 6.)}\\
&\Rightarrow B(\langle 2,2\rangle,\langle 3,4\rangle) && \text{(with 7.)}\\
&\Rightarrow B(\langle 2,2\rangle,\langle 4,4\rangle) && \text{(with 8.)}\\
&\Rightarrow \varepsilon && \text{(with 9.)}
\end{aligned}
$$

8.2

- G_1 is simple since in all clauses, RHS arguments are single variables and each variable occurring in the clause occurs exactly once in its LHS and exactly once in its RHS.
  $L(G_1) = \{ca^{2n}b^kca^{2n}b^kca^{2n} \mid n > 0, k \geq 0\}$.
- G_2 is a non-simple RCG, since the variable X is used twice in the RHS of $S(X) \to S_1(X)S_2(X)$ and therefore the grammar is not linear.
  $L(G_2) = \{a^nb^nc^k \mid n \geq 0, k \geq 1\} \cap \{a^kb^nc^n \mid k \geq 1, n \geq 0\} = \{a^nb^nc^n \mid n \geq 1\}$

8.3

1. L_1 can be generated by a simple RCG G_1:
   $G_1 = \langle\{S, A\}, \{c, d\}, \{X, Y, U, V\}, S, P\rangle$ with P the following set of clauses:
   $$
   \begin{aligned}
   S(XYUV) &\to A(X,Y,U,V)\\
   A(cX,cY,cU,cV) &\to A(X,Y,U,V)\\
   A(dX,dY,dU,dV) &\to A(X,Y,U,V)\\
   A(\varepsilon,\varepsilon,\varepsilon,\varepsilon) &\to \varepsilon
   \end{aligned}
   $$
2. L_2 can only be generated by a non-simple RCG G_2.
   $G_2 = \langle\{S, eq\}, \{a\}, \{X, Y, Z\}, S, P\rangle$ with P the following set of clauses:
   $$
   \begin{aligned}
   S(XYZ) &\to S(X)eq(X,Y,Z)\\
   S(a) &\to \varepsilon\\
   eq(aX,aY,aZ) &\to A(X,Y,Z)\\
   eq(a,a,a) &\to \varepsilon
   \end{aligned}
   $$

8.4

1. L_1 can be generated by a linear non-erasing simple LMG G_3 containing the same clauses as the simple RCG for the same language.
2. L_2 can be generated by the following simple LMG G_4.
   $G_4 = \langle \{S\}, \{a\}, \{X\}, S, P \rangle$ with P the following set of clauses:
   $S(XXX) \rightarrow S(X)$
   $S(a) \rightarrow \varepsilon$

8.5

Equivalent simple 2-RCG:
$S(X) \rightarrow \langle \alpha \rangle (X)$
$\langle \alpha \rangle (LR) \rightarrow \langle adj, \alpha, \varepsilon \rangle (L, R)$
$\langle \beta \rangle (aLb, cRd) \rightarrow \langle adj, \beta, 2 \rangle (L, R)$
$\langle adj, \alpha, \varepsilon \rangle (L, R) \rightarrow \langle \beta \rangle (L, R)$
$\langle adj, \beta, 2 \rangle (L, R) \rightarrow \langle \beta \rangle (L, R)$
$\langle adj, \alpha, \varepsilon \rangle (\varepsilon, \varepsilon) \rightarrow \varepsilon$
$\langle adj, \beta, 2 \rangle (\varepsilon, \varepsilon) \rightarrow \varepsilon$

Problems of Chapter 9

9.1

Assume that we have a k-RCG.
For binarized RCGs, the **complete** rule amounts to the following:

$$\textbf{Complete:}\ \frac{[A_1, \boldsymbol{\rho}_1], [A_2, \boldsymbol{\rho}_2]}{[A_0, \boldsymbol{\rho}]} \quad A_0(\boldsymbol{\rho}_0) \rightarrow A_1(\boldsymbol{\rho}_1) A_k(\boldsymbol{\rho}_2) \text{ an instantiated clause}$$

In contrast to simple RCG, we cannot assume that the variables in the left-hand side occur also in the right-hand side. Therefore, in the worst case, there is no relation at all between these variables. We then obtain for the left-hand side that we have ≤ 3 range boundaries per argument and $\leq 3k$ range boundaries per clause. For the right-hand side, we get ≤ 2 range boundaries per argument and therefore $\leq 2{\cdot}2k$ range boundaries per clause. Consequently, we have a total of $\leq 3k + 4k = 7k$ range boundaries per clause that can have values between 0 and n.

Therefore, parsing with this algorithm for the restricted type of RCG considered here is $\mathcal{O}(n^{7k})$.

9.2

Clause $A(aX, Ya, \varepsilon) \rightarrow C(XY)$.
Range constraint vector $\langle \mathbf{r}, C \rangle$ with

- $\mathbf{r} = (\langle r_1, r_2 \rangle, \langle r_3, r_4 \rangle, \langle r_5, r_6 \rangle, \langle r_7, r_8 \rangle, \langle r_9, r_{10} \rangle)$,

- $C = \{r_1 \leq r_2, \ldots, r_9 \leq r_{10},$
  $r_1 + 1 = r_2, r_7 + 1 = r_8,$
  $r_2 = r_3, r_4 = r_5, r_6 = r_7,$
  $r_9 = r_{10}\}$

9.3

Clauses of the RCG G:

$S(XY) \to A(X,X)B(Y,Y)$

$A(aX,bY) \to A(X,Y)$	$B(cX,dY) \to B(X,Y)$
$A(bX,Y) \to A(X,Y)$	$B(dX,Y) \to B(X,Y)$
$A(X,aY) \to A(X,Y)$	$B(X,cY) \to B(X,Y)$
$A(\varepsilon,\varepsilon) \to \varepsilon$	$B(\varepsilon,\varepsilon) \to \varepsilon$

1. $L(G) = \{w_1 w_2 \mid w_1 \in \{a,b\}^*$ with $|w_1|_a = |w_1|_b$ and $w_2 \in \{c,d\}^*$ with $|w_2|_c = |w_2|_d\}$.
2. $First(A,1) = First(A,2) = \{a,b,\varepsilon\}$,
   $First(B,1) = First(B,2) = \{c,d,\varepsilon\}$.
3. Possible filters:
   - The components of A belong to $\{a,b\}^*$ while the components of B belong to $\{c,d\}^*$.
   - For every range vector $(\langle l_1, r_1\rangle, \langle l_2, r_2\rangle)$ in the yield of A or B, it holds that $r_1 = r_2$.

Problems of Chapter 10

10.1

Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Q_F, Z_0\rangle$ be the original EPDA.
$M' = \langle Q \cup \{q_0', q_f\}, \Sigma, \Gamma \cup \{Z_0'\}, \delta', q_0', \{q_f\}, Z_0'\rangle$ with

- $q_0' \neq q_f, q_0', q_f \notin Q,\ Z_0' \notin \Gamma$;
- δ' extends δ as follows:
  $\delta'(q_0', \varepsilon, Z_0') = \{(q_0, \varepsilon, Z_0', \ddagger\#)\}$
  $\delta'(q_3, \varepsilon, Z_0') = \{(q_f, \varepsilon, Z_0', \varepsilon)\}$
  In all other cases, δ' is defined as the original δ.

10.2

$M = \langle \{q_0,q_1,q_2,q_3\}, \{a,b\}, \{\#, A, B\}, \delta, q_0, \{q_3\}, \#\rangle$ with
$\delta(q_0, \varepsilon, \#) = \{(q_3, \varepsilon, \varepsilon, \varepsilon), (q_1, \varepsilon, \#, \varepsilon)\}$
$\delta(q_1, a, X) = \{(q_1, \varepsilon, XA, \varepsilon)\}$
$\delta(q_1, b, X) = \{(q_1, \varepsilon, XB, \varepsilon)\}$
$\delta(q_1, \varepsilon, X) = \{(q_2, \varepsilon, X, \varepsilon)\}$
$\delta(q_2, \varepsilon, A) = \{(q_2, \ddagger A, \varepsilon, \varepsilon)\}$
$\delta(q_2, \varepsilon, B) = \{(q_2, \ddagger B, \varepsilon, \varepsilon)\}$
$\delta(q_2, \varepsilon, \#) = \{(q_3, \varepsilon, \varepsilon, \varepsilon)\}$
$\delta(q_3, a, A) = \{(q_3, \varepsilon, \varepsilon, \varepsilon)\}$
$\delta(q_3, b, B) = \{(q_3, \varepsilon, \varepsilon, \varepsilon)\}$

where $X \in \Gamma$

10.3

1. The language is $\{ww^R \mid w \in \{a,b\}^+\}$.
2. Successful configurations for $w = abba$:

thread set	remaining input	operation
$\boldsymbol{\varepsilon : S}$	abba	
$\varepsilon : S$, $\mathbf{1 : S'}$	abba	$S \longrightarrow [S]S'$
$\varepsilon : S$, $\mathbf{1 : S_A}$	bba	$S \xrightarrow{a} S_A$
$\varepsilon : S$, $1 : S_A$, $\mathbf{11 : S'}$	bba	$S_A \longrightarrow [S_A]S'$
$\varepsilon : S$, $1 : S_A$, $\mathbf{11 : B_2}$	ba	$S' \xrightarrow{b} B_2$
$\varepsilon : S$, $1 : S_A$, $\mathbf{11 : ret}$	a	$B_2 \xrightarrow{b} ret$
$\varepsilon : S$, $\mathbf{1 : A_2}$	a	$[S_A]ret \longrightarrow A_2$
$\varepsilon : S$, $\mathbf{1 : ret}$	ε	$A_2 \xrightarrow{a} ret$

References

Abeillé, Anne. 1988. Parsing French with Tree Adjoining Grammar: some linguistic accounts. In *Proceedings of COLING*, pages 7–12, Budapest.

Abeillé, Anne. 2002. *Une Grammaire Électronique du Français*. CNRS Editions, Paris.

Aho, A. V. 1968. Indexed grammars – an extension of context-free grammars. *Journal of the ACM*, 15(4):647–671.

Alonso Pardo, M. A., M.-J. Nederhof, and E. Villemonte de la Clergerie. 2000. Tabulation of automata for Tree-Adjoining Languages. *Grammars*, 3:89–110.

Barthélemy, François, Pierre Boullier, Philippe Deschamp, and Éric de la Clergerie. 2001. Guided parsing of Range Concatenation Languages. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 42–49.

Barton, G. Edward, Jr. 1985. The computational difficulty of ID/LP parsing. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 76–81, Chicago.

Becker, Tilman. 1994. A new automaton model for TAGs: 2-SA. *Computational Intelligence*, 10(4):422–430.

Becker, Tilman, Aravind K. Joshi, and Owen Rambow. 1991. Long-distance scrambling and Tree Adjoining Grammars. In *Proceedings of ACL-Europe*.

Becker, Tilman, Owen Rambow, and Michael Niv. 1992. The Derivational Generative Power of Formal Systems or Scrambling is Beyond LCFRS. Technical Report IRCS-92-38, Institute for Research in Cognitive Science, University of Pennsylvania.

Bellman, Richard 1957. *Dynamic Programming*. Princeton University Press.

Bertsch, Eberhard and Mark-Jan Nederhof. 2001. On the complexity of some extensions of RCG parsing. In *Proceedings of the Seventh International Workshop on Parsing Technologies*, pages 66–77, Beijing, China, October.

Boullier, Pierre. 1996. Another facet of LIG parsing. In *Proceedings of ACL 1996*.

Boullier, Pierre. 1998a. A generalization of mildly context-sensitive formalisms. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+4)*, pages 17–20, University of Pennsylvania, Philadelphia.

Boullier, Pierre. 1998b. A Proposal for a Natural Language Processing Syntactic Backbone. Technical Report 3342, INRIA.

Boullier, Pierre. 1999a. Chinese numbers, mix, scrambling, and range concatenation grammars. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL'99)*, pages 53–60, Bergen, Norway, June.

Boullier, Pierre. 1999b. On TAG Parsing. In *TALN 99, 6e conférence annuelle sur le Traitement Automatique des Langues Naturelles*, pages 75–84, Cargèse, Corse, July.

Boullier, Pierre. 2000a. A cubic time extension of context-free grammars. *Grammars*, 3(2/3):111–131.

Boullier, Pierre. 2000b. Range Concatenation Grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT2000)*, pages 53–64, Trento, Italy, February.

Boullier, Pierre and Benoît Sagot. 2009. Multi-Component Tree Insertion Grammars. In *Proceedings of Formal Grammar 2009*, Bordeaux, France, July. To appear in Lecture Notes in Computer Science, Springer.

Bresnan, Joan. 2001. *Lexical-Functional Syntax*, volume 16 of *Blackwell Textbooks in Linguistics*. Blackwell.

Bresnan, Joean, Ronald M. Kaplan, Stanley Peters, and Annie Zaenen. 1982. Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13(4):613–635. Reprinted in (Savitch et al., 1987).

Burden, Håkan and Peter Ljunglöf. 2005. Parsing linear context-free rewriting systems. In *IWPT'05, 9th International Workshop on Parsing Technologies*, Vancouver, Canada, October.

Candito, Marie-Hélène and Sylvain Kahane. 1998. Can the TAG derivation tree represent a semantic graph? An answer in the light of Meaning-Text Theory. In *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks, IRCS Report 98–12*, pages 25–28, University of Pennsylvania, Philadelphia.

Chiang, David and Tatjana Scheffler. 2008. Flexible composition and delayed tree-locality. In *TAG+9 Proceedings of the Ninth International Workshop on Tree-Adjoining Grammar and Related Formalisms (TAG+9)*, pages 17–24, Tübingen, June.

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory*, 2:113–124.

Chomsky, Noam. 1995. *The Minimalist Program*. MIT Press.

Crabbé, Benoit. 2005. *Représentation informatique de grammaires d'arbres fortement lexicalisées : le cas de la grammaire d'arbres adjoints*. Ph.D. thesis, Université Nancy 2.

Cremers, Armin B. and Otto Mayer. 1973. On matrix languages. *Information and Control*, 23:86–96.

Dassow, Jürgen and Gheorghe Păun. 1989. *Regulated Rewriting in Formal Languages Theory*, volume 18 of *EATCS Monographs on Theoretical Computer Science*. Springer.

de Groote, Philippe. 2001. Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th*

Conference of the European Chapter, Proceedings of the Conference, pages 148–155.

Dras, Mark, David Chiang, and William Schuler. 2004. On relations of constituency and dependency grammars. *Journal of Language and Computation*, 2(2):281–305.

Frank, Anette and Josef van Genabith. 2001. GlueTag. Linear logic based semantics for LTAG – and what it teaches us about LFG and LTAG. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of the LFG01 Conference*, Hong Kong.

Frank, Robert. 1992. *Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives.* Ph.D. thesis, University of Pennsylvania.

Frank, Robert. 2002. *Phrase Structure Composition and Syntactic Dependencies.* MIT Press, Cambridge, Mass.

Gallo, G., G. Longo, S. Nguyen, and S. Pallottino. 1993. Directed Hypergraphs and Applications. *Discrete Applied Mathematics*, 42:177–201.

Gazdar, Gerald. 1988. Applicability of indexed grammars to natural languages. In Uwe Reyle and Christian Rohrer, editors, *Natural Language Parsing and Linguistic Theories.* D. Reidel, pages 69–94.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullman, and Ivan Sag. 1985. *Generalized Phrase Structure Grammar.* Harvard University Press, Cambridge, Massachusetts.

Ginsburg, Seymour 1966. *The Mathematical Theory of Context Free Languages.* McGraw Hill, New York.

Gómez-Rodríguez, Carlos, Marco Kuhlmann, Giorgio Satta, and David Weir. 2009. Optimal reduction of rule length in linear context-free rewriting systems. In *Proceedings of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies Conference (NAACL'09:HLT)*, pages 539–547, Boulder, Colorado.

Groenink, Annius Victor. 1995. Literal movement grammars. In *Proceedings of the 7th EACL Conference.*

Groenink, Annius Victor. 1996. Mild context-sensitivity and tuple-based generalizations of context-free grammar. Report CS-R9634, Centrum voor Wiskunde en Informatica, Amsterdam.

Groenink, Annius Victor. 1997. *Surface Without Structure. Word Order and Tractability in Natural Language Analysis.* Ph.D. thesis, Utrecht University.

Grune, Dick and Ceriel Jacobs. 2008. *Parsing Techniques. A Practical Guide.* Monographs in Computer Science. Springer. Second Edition.

Han, Chung-Hye. 2002. Compositional semantics for relative clauses in lexicalized Tree Adjoining Grammars. In *Proceedings of the Sixth Interna-*

tional Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6), pages 1–10, Venice, May.

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

Huang, Liang and David Chiang. 2005. Better k-best parsing. In *Proceedings of IWPT 2005*, Vancouver, Canada.

Jäger, Gerhard and Jens Michaelis. 2004. An introduction to mildly context-sensitive grammar formalisms. Course Material at ESSLLI 2004, Nancy, France.

Joshi, Aravind K. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Parsing*. Cambridge University Press, pages 206–250.

Joshi, Aravind K., Laura Kallmeyer, and Maribel Romero. 2003. Flexible composition in LTAG: Quantifier scope and inverse linking. In Harry Bunt, Ielka van der Sluis, and Roser Morante, editors, *Proceedings of the Fifth International Workshop on Computational Semantics IWCS-5*, pages 179–194, Tilburg.

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

Joshi, Aravind K. and Yves Schabes. 1997. Tree-Adjoning Grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*. Springer, Berlin, pages 69–123.

Joshi, Aravind K. and K. Vijay-Shanker. 1999. Compositional semantics with lexicalized Tree-Adjoining Grammar (LTAG): How much underspecification is necessary? In H. C. Blunt and E. G. C. Thijsse, editors, *Proceedings of the Third International Workshop on Computational Semantics (IWCS-3)*, pages 131–145, Tilburg.

Kahane, Sylvain, Marie-Hélène Candito, and Yannick de Kercadio. 2000. An alternative description of extraction in TAG. In *Proceedings of TAG+5*, pages 115–122, Paris.

Kallmeyer, Laura. 2005. Tree-local multicomponent tree adjoining grammars with shared nodes. *Computational Linguistics*, 31(2):187–225.

Kallmeyer, Laura. 2009. A Declarative Characterization of Different Types of Multicomponent Tree Adjoining Grammars. *Research on Language and Computation*, 7(1):55–99.

Kallmeyer, Laura and Aravind K. Joshi. 2003. Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. *Research on Language and Computation*, 1(1–2):3–58.

Kallmeyer, Laura and Wolfgang Maier. 2009. An incremental Earley parser for simple Range Concatenation Grammar. In *Proceedings of IWPT 2009*.

Kallmeyer, Laura and Wolfgang Maier. 2010. Data-driven parsing with probabilistic Linear Context-Free Rewriting Systems. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, Beijing, China.

Kallmeyer, Laura, Wolfgang Maier, and Yannick Parmentier. 2009a. An Earley Parsing Algorithm for Range Concatenation Grammars. In *Proceedings of ACL 2009*, Singapore.

Kallmeyer, Laura, Wolfgang Maier, and Yannick Parmentier. 2009b. Un algorithme d'analyse de type Earley pour grammaires à concaténation d'intervalles. In *Actes de la 16ème conférence sur le Traitement Automatique des Langues Naturelles (TALN 2009)*, Senlis, France.

Kallmeyer, Laura and Yannick Parmentier. 2008. On the relation between Multicomponent Tree Adjoining Grammars with Tree Tuples (TT-MCTAG) and Range Concatenation Grammars (RCG). In Carlos Martín-Vide, Friedrich Otto, and Henning Fernaus, editors, *Language and Automata Theory and Applications. Second International Conference, LATA 2008*, number 5196 in Lecture Notes in Computer Science. Springer, Heidelberg Berlin, pages 263–274.

Kallmeyer, Laura and Maribel Romero. 2008. Scope and situation binding in LTAG using semantic unification. *Research on Language and Computation*, 6(1):3–52.

Kallmeyer, Laura and Giorgio Satta. 2009. A polynomial-time parsing algorithm for TT-MCTAG. In *Proceedings of ACL*, Singapore.

Kanazawa, Makoto. 2008. A Prefix-Correct Earley Recognizer for Multiple Context-Free Grammars. In *Proceedings of the Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+9)*, pages 49–56, Tübingen, June.

Kanazawa, Makoto. 2009. The pumping lemma for well-nested Multiple Context-Free Languages. In V. Diekert and D. Nowotka, editors, *DLT 2009*, volume 5583 of *LNCS*, pages 312–325, Berlin Heidelberg. Springer.

Kaplan, Ronald M. and Joan Bresnan. 1982. Lexical-Functional Grammar: A Formal System for Grammatical Representations. In *The Mental Representation of Grammatical Relations*. MIT Press, pages 173–281.

Kato, Yuki, Hiroyuki Seki, and Tadao Kasami. 2006. Stochastic multiple context-free grammar for RNA pseudoknot modeling. In *Proceedings of The Eighth International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+8)*, pages 57–64, Sydney, Australia, July.

Kay, Martin. 1986. Algorithm schemata and data structures in syntactic processing. In Barbara J. Grosz, Karen Sparck-Jones, and Bonnie Lynn Webber, editors, *Readings in Natural Language Processing*. Morgan Kaufmann, Los Altos, pages 35–70.

Klein, Dan and Christopher D. Manning. 2003. A* Parsing: Fast exact Viterbi parse selection. In *HLT-NAACL*.

Klein, Dan and Christopher D. Manning. 2004. Parsing and hypergraphs. In *New Developments in Parsing Technology*. Kluwer Academic Publishers, Norwell, MA, USA, pages 351–372.

Kracht, Marcus. 2003. *The Mathematics of Language*. Number 63 in Studies in Generative Grammar. Mouton de Gruyter, Berlin.

Kroch, Anthony. 1989. Asymmetries in long-distance extraction in a Tree Adjoining Grammar. In Baltin and Kroch, editors, *Alternative Conceptions of Phrase Structure*. University of Chicago.

Kroch, Anthony and Beatrice Santorini. 1991. The derived constituent structure of the West Germanic verb raising construction. In R. Freidin, editor, *Principles and Parameters in Comparative Grammar*. MIT Press, Cambridge, Mass., pages 268–338.

Kroch, Anthony S. 1987. Unbounded dependencies and subjacency in a Tree Adjoining Grammar. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam, pages 143–172.

Kroch, Anthony S. and Aravind K. Joshi. 1987. Analyzing extraposition in a tree adjoining grammar. In Geoffrey J. Huck and Almerido E. Ojeda, editors, *Syntax and Semantics: Discontinuous Constituency*. Academic Press, Inc., pages 107–149.

Kuhlmann, Marco. 2007. *Dependency Structures and Lexicalized Grammars*. Ph.D. thesis, Saarland University.

Kuhlmann, Marco and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Proceedings of EACL*.

Kuno, Susumu 1965. The predictive analyzer and a path elimination technique. *Communications of the ACM*, 8(7):453–462, July.

Langer, Hagen. 1998. Experimente mit verallgemeinerten Lookahead-Algorithmen. In Bernhard Schröder, Winfried Lenders, Wolfgang Hess, and Thomas Portele, editors, *Computers, linguistics and phonetics between language and speech / KONVENS 98*, pages 69–82, Bonn.

Levy, Roger. 1999. *Probabilistic Models of Word Order and Syntactic Discontinuity*. Ph.D. thesis, Stanford University.

Lichte, Timm. 2007. An MCTAG with Tuples for Coherent Constructions in German. In *Proceedings of the 12th Conference on Formal Grammar 2007*, Dublin, Ireland.

Ljunglöf, Peter. 2004. *Expressivity and Complexity of the Grammatical Framework*. Ph.D. thesis, Department of Computer Science, Gothenburg University and Chalmers University of Technology, November.

Ljunglöf, Peter. 2005. A polynomial time extension of parallel Multiple Context-Free Grammar. In *Logical Aspects of Computational Linguistics*, volume 3492 of *Lecture Notes in Computer Science*. Springer, Berlin/Heidelberg, pages 177–188.

Maier, Wolfgang. 2010. Direct parsing of discontinuous constituents in German. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 58–66, Los Angeles, CA, USA, June. Association for Computational Linguistics.

Maier, Wolfgang and Laura Kallmeyer. 2010. Discontinuity and non-projectivity: Using mildly context-sensitive formalisms for data-driven parsing. In *Proceedings of the Tenth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+10)*, New Haven.

Maier, Wolfgang and Timm Lichte. 2009. Characterizing discontinuity in constituent treebanks. In *Proceedings of Formal Grammar 2009*, Bordeaux, France, July. To appear in Lecture Notes in Computer Science, Springer.

Maier, Wolfgang and Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In *Proceedings of the 13th Conference on Formal Grammar 2008*, Hamburg, Germany.

McAllester, David 2002. On the complexity analysis of static analyses. *Journal of the ACM*, 49(4):512–537.

Merlo, Paola, Harry Bunt, and Joakim Nivre. 2010. *Current Trends in Parsing Technology*. Springer.

Michaelis, Jens. 1998. Derivational minimalism is mildly context-sensitive. In *Proceedings. Logical Aspects of Computational Linguistics*, Grenoble.

Michaelis, Jens. 2001a. Derivational minimalism is mildly context-sensitive. In Michael Moortgat, editor, *Logical Aspects of Computational Linguistics*, volume 2014 of *LNCS/LNAI*, pages 179–198, Berlin, Heidelberg. Springer.

Michaelis, Jens. 2001b. Transforming linear context-free rewriting systems into minimalist grammars. In Philippe de Groote, Glyn Morrill, and Christian Retoré, editors, *Logical Aspects of Computational Linguistics*, volume 2099 of *LNCS/LNAI*, pages 228–244, Berlin, Heidelberg. Springer.

Michaelis, Jens and Marcus Kracht. 1997. Semilinearity as a syntactic invariant. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics. First International Conference, LACL '96, Nancy, France, September 23-25, 1996. Selected Papers*, volume 1328 of *LNCS/LNAI*, Berlin, Heidelberg. Springer.

Nederhof, Mark-Jan. 1997. Solving the correct-prefix property for TAGs. In T. Becker and H.-U. Krieger, editors, *Proceedings of the Fifth Meeting on Mathematics of Language*, pages 124–130, Schloss Dagstuhl, Saarbrücken, August.

Nederhof, Mark-Jan. 1998. An alternative LR algorithm for TAGs. In *Proceedings of ACL*, Montreal, Canada.

Nederhof, Mark-Jan. 1999. The computational complexity of the correct-prefix property for TAGs. *Computational Linguistics*, 25(3):345–360.

Nesson, Rebecca, Giorgio Satta, and Stuart Shieber. 2008. Complexity, parsing, and factorization of tree-local multi-component tree-adjoining gram-

mar. Technical Report TR-05-08, School of Engineering and Applied Sciences, Harvard University, Cambridge, MA.

Nesson, Rebecca and Stuart M. Shieber. 2006. Simpler TAG semantics through synchronization. In *Proceedings of the 11th Conference on Formal Grammar*, Malaga, Spain, 29–30 July.

Parikh, Rohit 1966. On context-free languages. *Jounal of the ACM*, 13:570–581.

Parmentier, Yannick and Wolfgang Maier. 2008. Using constraints over finite sets of integers for range concatenation grammar parsing. In Bengt Nordström and Aarne Ranta, editors, *Advances in Natural Language Processing*, volume 5221 of *LNCS/LNAI*, Gothenburg, Sweden, August. Springer.

Pereira, Fernando C. N. and David Warren. 1983. Parsing as deduction. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 137–144, MIT, Cambridge, Massachusetts.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. The University of Chicago Press, Chicago, London.

Prolo, Carlos. 2000. An efficient LR parser generator for Tree Adjoining Grammars. In *Proceedings of the 6th International Workshop on Parsing Technologies (IWPT-2000)*, pages 207–218, Trento, Italy.

Prolo, Carlos. 2003. *LR Parsing for Tree Adjoining Grammars and Its Application to Corpus-Based Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Radzinski, Daniel. 1991. Chinese number-names, tree adjoining languages, and mild context-sensitivity. *Computational Linguistics*, 17:277–299.

Rambow, Owen. 1994. *Formal and Computational Aspects of Natural Language Syntax*. Ph.D. thesis, University of Pennsylvania.

Rambow, Owen, K. Vijay-Shanker, and David Weir. 1995. D-Tree Grammars. In *Proceedings of ACL*.

Rambow, Owen, K. Vijay-Shanker, and David Weir. 2001. D-Tree Substitution Grammars. *Computational Linguistics*.

Sagot, Benoît. 2005. Linguistic facts as predicates over ranges of the sentence. In *Proceedings of LACL 05*, number 3492 in Lecture Notes in Computer Science, pages 271–286, Bordeaux, France. Springer.

Savitch, Walter J., Emmon Bach, William Marxh, and Gila Safran-Naveh, editors. 1987. *The Formal Complexity of Natural Language*. Studies in Linguistics and Philosophy. Reidel, Dordrecht, Holland.

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

Schabes, Yves and Aravind K. Joshi. 1988. An Earley-type parsing algorithm for Tree Adjoining Grammars. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 258–269.

Schabes, Yves and K. Vijay-Shanker. 1990. Deterministic left to right parsing of tree adjoining languages. In *Proceedings of ACL*, Pittsburgh.

Seki, Hiroyuki and Yuki Kato. 2008. On the generative power of multiple context-free grammars and macro grammars. *IEICE Transactions on Information and Systems*, E91-D(2):209–221, February.

Seki, Hiroyuki, Takahashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

Seki, Hiroyuki, Ryuichi Nakanishi, Yuichi Kaji, Sachiko Ando, and Tadao Kasami. 1993. Parallel multiple context-free grammars, finite-state translation systems, and polynomial-time recognizable subclasses of lexical-functional grammars. In *31st Meeting of the Association for Computational Linguistics (ACL'93)*, pages 121–129.

Shieber, Stuart M. 1984. Direct parsing of ID/LP grammars. *Linguistics and Philosophy*, 7(2):135–154.

Shieber, Stuart M. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343. Reprinted in (Savitch et al., 1987).

Shieber, Stuart M. 1994. Restricting the weak-generative capacity of synchronous Tree-Adjoining Grammars. *Computational Intelligence*, 10(4):271–385.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1 and 2):3–36.

Sikkel, Klaas. 1997. *Parsing Schemata*. Texts in Theoretical Computer Science. Springer, Berlin, Heidelberg, New York.

Sippu, Seppo and Eljas Soisalon-Soininen. 1990. *Parsing Theory*, volume 20 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, Heidelberg.

Søgaard, Anders. 2007. *Complexity, expressivity and logic of linguistic theories*. Ph.D. thesis, University of Copenhagen, Copenhagen, Denmark.

Søgaard, Anders. 2008. Range concatenation grammars for translation. In *Proceedings of the 22nd International Conference on Computational Linguistics*, Manchester, England.

Søgaard, Anders, Timm Lichte, and Wolfgang Maier. 2007. The complexity of linguistically motivated extensions of tree-adjoining grammar. In *Recent Advances in Natural Language Processing 2007*, Borovets, Bulgaria.

Stabler, Edward P. 1997. Derivational Minimalism. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics. First International Conference, LACL '96, Nancy, France, September 23-25, 1996. Selected Papers*, volume 1328 of *LNCS/LNAI*, pages 68–95, Berlin, Heidelberg. Springer.

Steedman, Mark. 2000. *The Syntactic Process*. MIT Press.

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

Vijay-Shanker, K. and Aravind K. Joshi. 1985. Some computational properties of Tree Adjoining Grammars. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 82–93.

Vijay-Shanker, K. and David J. Weir. 1993. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.

Vijay-Shanker, K. and David J. Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27(6):511–546.

Vijay-Shanker, K., David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of ACL*, Stanford.

Villemonte de la Clergerie, Éric. 2002. Parsing mildly context-sensitive languages with thread automata. In *Proceedings of COLING'02*, August.

Villemonte de la Clergerie, Eric. 2006. Designing tabular parsers for various syntactic formalisms. ESSLLI Lecture Notes.

Villemonte de la Clergerie, Éric and Alonso M.A. Pardo. 1998. A tabular interpretation of a class of 2-stack automata. In *Proceedings of COLING-ACL*, pages 1333–1339.

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*, 104:235–261.

XTAG Research Group. 2001. A Lexicalized Tree Adjoining Grammar for English. Technical report, Institute for Research in Cognitive Science, Philadelphia.

Index

Abstract Categorial Grammar, 7

chart parsing, 46, 47
Combinatory Categorial Grammar, 7, 73
  backward application, 73
  backward composition, 73
  forward application, 73
  forward composition, 73
computation sharing, 46
constant growth property, 23
Context-Free Grammar, 11
  Chomsky Normal Form, 12, 41
  closure properties, 12
  CYK parsing, 41, 42
  derivation tree, 15
  Earley parsing, 43
  Greibach Normal Form, 12, 21
  LR parsing, 98
  parse tree, 15
  pumping lemma, 12
  string language, 11
  tree language, 15
  useful symbol, 11
cross-serial dependencies, 17
  – in Dutch, 17
  – in Swiss-German, 18
  – with TAG, 28
CYK parsing
  – for CFG, 42
  – for MCFG, 131
  – for RCG, 178
  – for TAG, 77

dependency parsing, 8
dynamic programming, 44

Earley parsing
  – for CFG, 43
  – for MCFG, 141
  – for RCG, 188
  – for TAG, 82
Embedded Push-Down Automaton, 193, 195
  k-order –, 199
  bottom-up –, 197
  language, 195
  transition, 195
equivalence
  strong –, 15
  weak –, 15

Finite State Automaton, 13
finitely ambiguous, 21
fixed recognition problem, 49

graph
  directed –, 14
  in-degree, 14
  out-degree, 14

Head-Driven Phrase Structure Grammar (HPSG), 7

Indexed grammar, 31, 72
  Linear –, 31, 72

language, 10

alphabet, 10
empty word, 10
homomorphism, 11
length of a word, 11
letter equivalent, 24
Parikh mapping, 24
semilinear, 25
word, 10
Lexical Functional Grammar, 126
c-structure, 126
f-structure, 127
finite-copying –, 127
Lexical Functional Grammar (LFG), 7
lexicalization, 21
strong lexicalization, 21
weak lexicalization, 21
lexicalized grammar, 20
anchor, 20
multicomponent anchor, 21
Linear Context-Free Rewriting System, 24, 33, 111
fan-out, 117
CYK parsing, 131
monotone –, 145
rank, 147
well-nested –, 122
Literal Movement Grammar, 167
clause instantiation, 168
linear –, 169
non-combinatorial –, 169
non-erasing –, 169
simple –, 169
string language, 168
LR parsing
– for CFG, 98
– for TAG, 96

Matrix Grammars, 7
mildly context-sensitive, 23
Minimalist Grammar, 126
Multicomponent TAG, 33, 70
k-TT-MCTAG, 71
k-delayed tree-local –, 71
non-local –, 35
set-local –, 24, 35, 125
tree-local –, 35, 71
tree-local – with flexible composition, 71
Multiple Context-Free Grammar, 24, 36, 110
r-yield, 114
k-MCFG, 110
closure properties, 119
CYK parsing, 131
incremental CYK parsing, 139
incremental Earley parsing, 141
left-corner parsing, 142
mcf-function, 111
parallel –, 169
pumping lemma, 118
range, 113
string language, 111, 114

NP-complete, 50

Parikh Theorem, 25
parsing
$LL(k)$, 44
– and hypergraphs, 48
– as deduction, 6, 41
active item, 44
chart –, 6
completeness, 48
complexity, 49
deduction rules, 44
dependency –, 8
dotted production, 43
item, 42
passive item, 44
soundness, 48
pseudo-code, 41
PTIME, 37, 50
pumping lemma
– for CFG, 12
– for MCFG, 118
– for TAG, 61
Push-Down Automaton, 13

Range Concatenation Grammar, 36, 157
$(2, 2)$-BRCG, 166
arity, 117
clause, 117
bottom-up linear –, 162
bottom-up non-erasing –, 162
clause instantiation, 160
combinatorial clause, 159

CYK parsing, 178
derivation, 161
Earley parsing, 188
erasing clause, 157
filters for parsing, 183
linear –, 162
negative –, 159
non-combinatorial –, 162
non-erasing –, 163
non-linear clause, 158
positive –, 159
range, 113
range constraint vector, 185
range language, 161
simple –, 24, 37, 163
string language, 161
top-down linear –, 162
top-down non-erasing –, 163
top-down parsing, 179

Simple Range Concatenation Grammar, 111, 112
ε-free –, 143
ε-clause, 143
arity, 117
derivation tree, 115, 116
eliminating ε-rules, 143
gap degree, 121
ordered –, 145
tree language, 117
well-nested, 122
binarization, 147
clause instantiation, 115
eliminating useless rules, 142
filters for parsing, 154
simple k-RCG, 112
transformation into an ordered –, 146
substitution, 22
substitution node, 22

tabulation, 46
Thread Automaton, 204, 206
– for LCFRL, 209
– for TAL, 208
configuration, 206
language, 208
thread, 206
transitions, 207
tree, 14
completed –, 22
labeling, 15
ordered tree, 14
syntactic tree, 15
Tree Adjoining Grammar, 26, 55
adjunction, 55
adjunction constraints, 55
auxiliary tree, 54
closure properties, 58
Condition on Elementary Tree Minimality, 64
cross-serial dependencies, 28
CYK parsing, 77
derivation tree, 56, 68
derived tree, 56, 68
Earley Parsing, 82
elementary tree, 55
extended domain of locality, 30, 65
factoring of recursion, 65
initial tree, 54
LR parsing, 96
Predicate Argument Co-occurrence Principle, 63
prefix valid Earley parsing, 93
pumping lemma, 61
string language, 58
tree language, 58
Tree Substitution Grammar, 22
elementary tree, 22
tree language, 22
Two-Stack Automaton, 200
strongly-driven –, 203
two-stack automaton, 200

universal recognition problem, 50
unordered Vector Grammars, 7

valid prefix property, 51
Earley parser for TAG, 92
LR parser for TAG, 107
Vector-TAG with dominance links, 7

Weir Hierarchy, 199