Grammars with Regulated Rewriting

Jürgen Dassow
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik

Lecture
in the
5^{th} PhD Program Formal Languages and Applications

References [1] – [4] are summarizing books/papers, the remaining references give the papers where the regulations were introduced.

References

[1] J. DASSOW, Grammars with regulated Rewriting. In C. MARTIN-VIDE, V. MITRANA and GH. PĂUN (eds.), *Formal Languages and Applications*, Studies in Fuzziness and Soft Computing 148, Springer-Verlag, Berlin 2004.

[2] J. DASSOW and GH. PĂUN, *Regulated Rewriting in Formal Language Theory*. Akademie-Verlag, Berlin and Springer-Verlag, Berlin, 1989.

[3] J. DASSOW, GH. PĂUN and G. ROZENBERG, Grammars with Controlled derivations. In *Handbook of Formal Languages*, Vol. II, Chapter 3, 101–154, Springer-Verlag, 1997

[4] A. SALOMAA, *Formal Languages*. Academic Press, New York, 1973.

[5] S. Abraham, Some questions of phrase-structure grammars. *Comput. Linguistics* **4** (1965) 61-70.

[6] A.V. Aho, Indexed grammars - an extension of context-free grammars. *J. ACM* **15** (1968) 647-671.

[7] A.B. Cremers and O. Mayer, On matrix languages. *Inform. Control* **23** (1973) 86-96.

[8] I. Fris, Grammars with a partial ordering of the rules. *Inform. Control* **12** (1968) 415-425.

[9] S. Ginsburg and E.H. Spanier, Control sets on grammars. *Math. Syst. Th.* **2** (1968) 159-177.

[10] J. Kelemen, Conditional grammars: motivations, definitions and some properties. In: *Proc. Conf. Automata Languages and Math. Sciences*, Salgotarjan, 1984, 110-123.

[11] Gh. Paun, A new generative device: valence grammars. *Rev. Roum. Math. Pures Appl.* **25** (1980) 911-924.

[12] D.J. Rosenkrantz, Programmed grammars and classes of formal languages. *J. ACM* **16** (1969) 107-131.

[13] A.P.J. van der Walt, Random context grammars. In: *Proc. Symp. on Formal Languages*, 1970.

Preliminaries I

N – set of all natural numbers – $\{0, 1, 2, 3, \ldots\}$,
Z – set of all integers
Q – set of all rational numbers

$V = \{a_1, a_2, \ldots, a_n\}$ – alphabet (with letters in a fixed order)
V^* – set of all words over V
V^+ – set of all non-empty words over V

$|w|$ – length of the word w
$\#_U(w)$ – number of occurrences of letters of $U \subset V$ in $w \in V^*$

Preliminaries II

$\Psi_V(w) = (\#_{a_1}(w), \#_{a_2}(w), \ldots, \#_{a_n}(w))$ – Parikh vector of w
$\Psi_V(L) = \{\Psi_V(w) \mid w \in L\}$ – Parikh language of $L \subset V^*$

$L \subset V^*$ is called <u>semilinear</u> if and only if its Parikh language $\Psi_V(L)$ is a finite union of linear subsets of $\mathbf{N}^n$
(i.e., a finite union of sets which can be represented

$$\{v_0 + \sum_{i=1}^{m} \gamma_i v_i \mid \gamma_i \in \mathbf{N}, 1 \leq i \leq m\}$$

for some $v_0, v_1, \ldots, v_m \in \mathbf{N}^n$

Preliminaries III

$G = (N, T, S, P)$ – phrase-structure grammar
N – set of nonterminals (usually denoted by capitals)
T – set of nonterminals (usually denoted by small letters)
P – set of productions (or rules) $\alpha \to \beta$
S – axiom
$V_G = N \cup T$ (sometimes only denoted by V)

G length-increasing iff $|\alpha| \leq |\beta|$ for all $\alpha \to \beta \in P$
G context-free iff all rules of P have the form $A \to \beta$ with $A \in N$, $\beta \in V_G^*$
G regular iff all rules of P have the form $A \to \beta$ with $A \in N$, $\beta \in TN \cup T \cup \{\lambda\}$

Preliminaries IV

REG	– family of all regular languages
CF	– family of all context-free languages
CS	– family of all length-increasing/context-sensitive languages
RE	– family of all recursively enumerable languages

$\{ww \mid w \in \{a,b\}^* \} \notin CF$,
$\{a^n c^m b^n d^m \mid n,m \geq 1\} \notin CF$,
$\{a^{2^n} \mid n \geq 0\} \notin CF$,
$\{a^n b^n c^n \mid n \geq 1\} \notin CF$

Preliminaries V

Any $L \in CS$ can be generated by grammar $G = (N, T, S, P)$ where all rules are of the form $AB \to CD$, $A \to BC$, $A \to B$ and $A \to a$ with $A, B, C, D \in N$ and $a \in T$.

Any $L \in RE$ can be generated by grammar $G = (N, T, S, P)$ where all rules are of the form $AB \to CD$, $A \to BC$, $A \to B$, $A \to a$ and $A \to \lambda$ with $A, B, C, D \in N$ and $a \in T$.

For any language $L \in REG$, there is a finite automaton $\mathcal{A} = (X, Z, z_0, F, \delta)$ (with input set X, state Z, initial state z_0, set F of accepting states and transition function δ) such that the set $T(\mathcal{A})$ of words accepted by $\mathcal{A}$ coincides with L.

Motivation – English

Mary	and John	are		
a woman	and a man,	respectively.		

Mary,	John	and William	are	
a woman,	a man	and a man,	respectively.	

Mary,	John,	William	and Jenny	are
a woman,	a man,	a man	and a woman,	respectively.

Sentences of this type form a sublanguage L with
$h(L) = \{ww \mid w \in \{a,b\}\}$ for some morphism h

Motivation – Swiss German

Jan säit das mer em Hans hälfed.
(Jan says that we helped Hans.)

Jan säit das mer em Hans es Huus hälfed aastriche.
(Jan said that we helped Hans to paint the house.)

Jan säit das mer d'chind em Hans es Huus lönd hälfed aastriche.
(Jan said that we allowed the children to help Hans to paint the house.)

Sentences of this type form a sublanguage L with
$h(L) = \{ww \mid w \in \{a, b\}\}$ for some morphism h

Regularly Controlled Grammar – Definition

Definition ([9]) :
i) A regularly controlled (context-free) grammar is a 5-tuple $G = (N, T, S, P, R)$ where
— N, T, P and S are specified as in a context-free grammar,
— R is a regular set over P.

ii) The language $L(G)$ generated by G consists of all words $w \in T^*$ such that there is a derivation

$$S \Longrightarrow_{p_1} w_1 \Longrightarrow_{p_2} w_2 \Longrightarrow_{p_3} \ldots \Longrightarrow_{p_n} w_n = w$$

with

$$p_1 p_2 p_3 \ldots p_n \in R.$$

Regularly Controlled Grammar – Example I

$$G = (\{S, A, B\}, \{a, b\}, S, \{p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}, R)$$

$$\begin{array}{llll} p_0 = S \rightarrow AB, & & & \\ p_1 = A \rightarrow aA, & p_2 = B \rightarrow aB, & p_3 = A \rightarrow bA, & p_4 = B \rightarrow bB, \\ p_5 = A \rightarrow a, & p_6 = B \rightarrow a, & p_7 = A \rightarrow b, & p_8 = B \rightarrow b \end{array}$$

$$R = p_0\{p_1p_2, p_3p_4\}^*\{p_5p_6, p_7p_8\}$$

$$L(G) = \{ww \mid w \in \{a, b\}^+ \}$$

Regularly Controlled Grammar – Example II

$G = (\{S, A, B\}, \{a, b, c, d\}, S, \{p_0, p_1, p_2, p_3, p_4, p_5, p_6\}, R)$

$p_0 = S \to AB$,
$p_1 = A \to aA$, $p_2 = B \to bB$, $p_3 = A \to cA$, $p_4 = B \to dB$,
$p_5 = A \to c$, $p_6 = B \to d$

$R = p_0(p_1p_2)^+(p_3p_4)^*p_5p_6$

$L(G) = \{a^nc^mb^nd^m \mid n \geq 1, m \geq 1\}$

Appearance Checking

N and T – set of nonterminals and terminals, respectively
P – (finite) set of (context-free) productions over $V = N \cup T$
F – subset of P

We say that $x \in V^+$ directly derives $y \in V^*$ in appearance checking mode by application of $p = A \rightarrow w \in P$ (written as $x \Longrightarrow_p^{ac} y$) if one of the following conditions hold:

$$x = x_1 A x_2 \text{ and } y = x_1 w x_2$$

or

$$A \text{ does not appear in } x,\ p \in F \text{ and } x = y.$$

Regularly Controlled Grammar with Appearance Checking – Definition

Definition ([9]) :

i) A regularly controlled (context-free) grammar with appearance checking is a 6-tuple $G = (N, T, S, P, R, F)$ where

— N, T, P, S and R are specified as in a regularly controlled grammar and

— F is a subset of P.

ii) The language $L(G)$ generated by G with appearance checking consists of all words $w \in T^*$ such that there is a derivation

$$S \Longrightarrow_{p_1}^{ac} w_1 \Longrightarrow_{p_2}^{ac} w_2 \Longrightarrow_{p_3}^{ac} \ldots \Longrightarrow_{p_n}^{ac} w_n = w$$

with

$$p_1 p_2 p_3 \ldots p_n \in R.$$

Regularly Controlled Grammar with Appearance Checking – Example

$G = (\{S, A, X\}, \{a\}, S, \{p_1, p_2, p_3, p_4, p_5\}, R, F),$

$p_1 = S \rightarrow AA, \quad p_2 = S \rightarrow X, \quad p_3 = A \rightarrow S, \quad p_4 = A \rightarrow X, \quad p_5 = S \rightarrow a$

$R = (p_1^* p_2 p_3^* p_4)^* p_5^*,$

$F = \{p_2, p_4\}$

$L(G) = \{a^{2^m} \mid m \geq 1\}$

Regularly Controlled Grammars versus Chomsky Grammars

λrC	–	family of all languages generated by regularly controlled grammars (without appearance checking)
λrC_{ac}	–	family of all languages generated by regularly controlled grammars with appearance checking
rC	–	family of all languages generated by regularly controlled grammars without erasing rules (and without appearance checking)
rC_{ac}	–	family of all languages generated by regularly controlled grammars with appearance checking and without erasing rules

Theorem :
i) All languages of λrC over a unary alphabet are regular.
ii) $CF \subset rC \subset rC_{ac} \subset CS$
iii) $CF \subset rC \subseteq \lambda rC \subset \lambda rC_{ac} = RE$

Matrix Grammar – Definition I

Definition ([5]) :
i) A matrix grammar with appearance checking is a quintuple
$G = (N, T, S, M, F)$ where

— N, T and S are specified as in a context-free grammar,

— $M = \{m_1, m_2, \ldots m_n\}$, $n \geq 1$, is a finite set of sequences
$m_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,k(i)})$, $k(i) \geq 1$, $1 \leq i \leq n$,
where any $p_{i,j}$, $1 \leq i \leq n$, $1 \leq j \leq k(i)$, is a context-free production,

— F is a subset of all productions occuring in the elements of M,
i.e., $F \subseteq \{p_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq k(i)\}$.

ii) We say that M is a matrix grammar without appearance checking if and only if $F = \emptyset$.

Matrix Grammar – Definition II

iii) For m_i, $1 \leq i \leq n$, and $x, y \in V_G^*$, we define $x \Longrightarrow_{m_i} y$ by

$$x = x_0 \Longrightarrow_{p_{i,1}}^{ac} x_1 \Longrightarrow_{p_{i,2}}^{ac} x_2 \Longrightarrow_{p_{i,3}}^{ac} \ldots \Longrightarrow_{p_{i,k(i)}}^{ac} x_{k(i)} = y$$

iv) The language $L(G)$ generated by G (with appearance checking) is defined as the set of all words $w \in T^*$ such that there is a derivation

$$S \Longrightarrow_{m_{j_1}} y_1 \Longrightarrow_{m_{j_2}} y_2 \Longrightarrow_{m_{j_3}} \ldots \Longrightarrow_{m_{j_k}} y_k = w$$

for some $k \geq 1$, $1 \leq j_i \leq n$, $1 \leq i \leq k$.

Matrix Grammar – Example I

$G = (\{S, A, B\}, \{a, b\}, S, \{m_0, m_1, m_2, m_3, m_4\}, \emptyset)$

$$m_0 = (S \to AB), \quad m_1 = (A \to aA,\ B \to aB), \quad m_2 = (A \to bA,\ B \to bB),$$
$$m_3 = (A \to a,\ B \to a), \quad m_4 = (A \to b,\ B \to b)$$

$L(G) = \{ww \mid w \in \{a, b\}^+ \}$

$G' = (\{S, A, B\}, \{a, b, c, d\}, \{m_0, m_1, \ldots, m_4\}, S, \emptyset)$

$$m_0 = (S \to ACBD, \quad m_1 = (A \to aA,\ B \to bB), \quad m_2 = (A \to a,\ B \to b),$$
$$m_3 = (C \to cC,\ D \to dD), \quad m_4 = (C \to c,\ D \to d)$$

$L(G') = \{a^n c^m b^n d^m \mid n \geq 1, m \geq 1\}$

Matrix Grammar – Example II

$G = (\{S, A, A', B, C, D, X\}, \{a, b\}, M, S, F)$

$M = \{m_0, m_0', m_1, m_2, m_3, m_4, m_5, m_5'\}$

$$
\begin{array}{ll}
m_0 = (S \rightarrow AB), & m_0' = (S \rightarrow AD) \\
m_1 = (A \rightarrow A'A',\ B \rightarrow B), & m_2 = (A \rightarrow X,\ B \rightarrow C), \\
m_3 = (A \rightarrow a,\ D \rightarrow D), & m_4 = (A \rightarrow X,\ D \rightarrow b), \\
m_4 = (A' \rightarrow A,\ C \rightarrow C), & m_5 = (A' \rightarrow X,\ C \rightarrow B), \\
m_5' = (A' \rightarrow X,\ C \rightarrow D) &
\end{array}
$$

$F = \{A \rightarrow X, A' \rightarrow X\}$

$L(G) = \{a^{2^m} b \mid m \geq 0\}$

Matrix Grammars versus Regularly Controlled Grammars

λM	–	family of all languages generated by matrix grammars (without appearance checking)
λM_{ac}	–	family of all languages generated by matrix grammars with appearance checking
M	–	family of all languages generated by matrix grammars without erasing rules (and without appearance checking)
M_{ac}	–	family of all languages generated by matrix grammars with appearance checking and without erasing rules

Theorem :

i) $M = rC$,

ii) $\lambda M = \lambda rC$,

iii) $M_{ac} = rC_{ac}$,

iv) $\lambda M_{ac} = \lambda rC_{ac}$

Unordered Vector Grammar – Definition

Definition ([7]) :
i) An (unordered) vector grammar is a quadruple $G = (V, T, S, M)$ where N, T, M and S are defined as for matrix grammars.

ii) The language $L(G)$ generated by G is defined as the set of all words $w \in T$ such that there is a derivation

$$S \Longrightarrow_{p_1} w_1 \Longrightarrow_{p_2} w_2 \Longrightarrow_{p_3} \ldots \Longrightarrow_{p_n} w$$

where $p_1 p_2 \ldots p_n$ is a permutation of some element of M^*.

Unordered Vector Grammar – Example

$$G = (\{S, A, B\}, \{a, b\}, \{m_0, m_1, m_2, m_3, m_4\}, S, \emptyset)$$

$$\begin{array}{ll} m_0 = (S \to AB), & \\ m_1 = (A \to aA,\ B \to aB), & m_2 = (A \to bA,\ B \to bB), \\ m_3 = (A \to a,\ B \to a), & m_4 = (A \to b,\ B \to b) \end{array}$$

$$\{wxw'x \mid x \in \{a, b\}, w \in \{a, b\}^*, w' \in Perm(\{w\})\}$$

Unordered Vector Grammars versus Matrix Grammars

λuV	–	family of all languages generated by unordered vector grammars
uV	–	family of all languages generated by unordered vector grammars without erasing rules

Theorem :
$CF \subset uV = \lambda uV \subset M$

Theorem :
Each language in uV is semilinear.

Programmed Grammar – Definition I

Definition ([12]) :

i) A programmed grammar is a quadruple $G = (N, T, S, Lab, P, P_G)$ where

— N, T and S are specified as in a context-free grammar,

— Lab is a finite set of labels,

— P is a finite set of context-free productions (set of core productions)

P_G is a finite set of quadruples $r = (q, p, \sigma, \varphi)$ where $q \in Lab$, $p \in P$, and σ and φ are subsets of Lab.

If $r = (q, p, \sigma, \varphi)$, then σ and φ are called the success field and failure field of r, respectively.

ii) If $r = (q, p, \sigma, \emptyset)$ holds for any $r \in P_G$, then we say that G is a programmed grammar without appearance checking. Otherwise G is a programmed grammar with appearance checking.

Programmed Grammar – Definition II

iii) The language $L(G)$ generated by G is defined as the set of all words $w \in T^*$ such that there is a derivation

$$S = w_0 \Longrightarrow_{r_1} w_1 \Longrightarrow_{r_2} w_2 \Longrightarrow_{r_3} \dots \Longrightarrow_{r_k} w_k = w,$$

$k \geq 1$ and, for $r_i = (q_i, A_i \to v_i, \sigma_i, \varphi_i)$, one of the following conditions hold:

$w_{i-1} = w'_{i-1} A_i w''_{i-1}$, $w_i = w'_{i-1} v_i w''_{i-1}$ for some $w'_{i-1}, w''_{i-1} \in V_G^*$ and $q_{i+1} \in \sigma_i$

or

A_i does not occur in w_{i-1}, $w_{i-1} = w_i$ and $q_{i+1} \in \varphi_i$.

Programmed Grammar – Examples I

$$G = (\{S, A, B\}, \{a, b\}, S, \{q_0, q_1, ;q_8\}, P, \{r_0, r_1, r_2, \ldots, r_8\})$$

$$P = \{S \to AB,\ A \to aA,\ B \to aB,\ A \to bA,\ B \to bB,$$
$$A \to a,\ B \to a,\ A \to b,\ B \to b\}$$

$r_0 = (q_0, S \to AB, \{q_1, q_3, q_5, q_7\}, \emptyset)$,

$r_1 = (q_1, A \to aA, \{q_2\}, \emptyset)$, $r_2 = (q_2, B \to aB, \{q_1, q_3, q_5, q_7\}, \emptyset)$,

$r_3 = (q_3, A \to bA, \{q_4\}, \emptyset)$, $r_4 = (q_4, B \to bB, \{q_1, q_3, q_5, q_7\}, \emptyset)$,

$r_5 = (q_5, A \to a, \{q_6\}, \emptyset)$, $r_6 = (q_6, B \to a, \emptyset, \emptyset)$,

$r_7 = (q_7, A \to b, \{q_8\}, \emptyset)$, $r_8 = (q_8, B \to b, \emptyset, \emptyset)$,

$$L(G) = \{ww \mid w \in \{a, b\}^+ \}$$

Programmed Grammar – Examples II

$G' = (\{S,A\},\{a\},\{q_1,q_2,q_3\},P',\{r_1,r_2,r_3\},S)$

$P' = \{S \to AA,\ A \to S,\ S \to a\}$

$r_1 = (q_1, S \to AA, \{q_1\}, \{q_2\}), \quad r_2 = (q_2, A \to S, \{q_2\}, \{q_1, q_3\}),$
$r_3 = (q_3, S \to a, \{q_3\}, \emptyset)$

$L(G') = \{a^{2^m} \mid m \geq 0\}$

Programmed Grammars versus Matrix Grammars

λP	–	family of all languages generated by programmed grammars (without appearance checking)
λP_{ac}	–	family of all languages generated by programmed grammars with appearance checking
P	–	family of all languages generated by programmed grammars without erasing rules (and without appearance checking)
P_{ac}	–	family of all languages generated by programmed grammars with appearance checking and without erasing rules

Theorem :

i) $P = M$,

ii) $\lambda P = \lambda M$,

iii) $P_{ac} = M_{ac}$,

iv) $\lambda P_{ac} = \lambda M_{ac}$

Regularly Controlled Grammar – Another Interpretation I

$R \subset X^*$ – regular set
$R = T(\mathcal{A})$ for some finite (nondeterministic) automaton $\mathcal{A} = (X, Z, z_0, Q, \delta)$
$\mathcal{A}$ – description as a graph with labelled edges

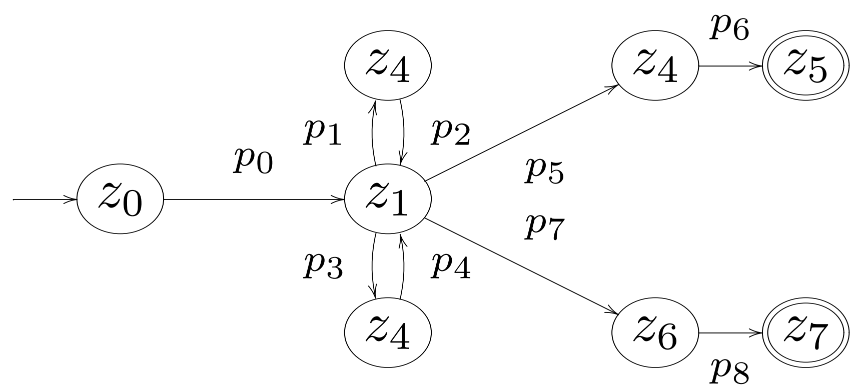
$z' = \delta(z, x)$ corresponds to $z \xrightarrow{x} z'$

$x_1x_2 \ldots x_n \in T(\mathcal{A}) = R$ if and only if $x_1x_2 \ldots x_n$ is a sequence of edge labels given by a path from z_0 to some state in Q

Control by a regular set corresponds to
control by sequences of edge labels of paths from a source node to a target node
(in a graph with edge labelling, one source node and a set of target nodes)

Regularly Controlled Grammar – Another Interpretation II

$$R = p_0\{p_1p_2, p_3p_4\}^*\{p_5p_6, p_7p_8\}$$



source node – z_0
target nodes – z_5, z_7

Programmed Grammars – Another Interpretation I

$G = (N, T, S, Lab, P, P_G)$ – programmed grammar without appearance checking
$r = (q(r), p(r), \sigma(r), \varphi(r))$

$H = (Lab, E)$ – graph
$(q(r), q') \in E$ if and only if $q' \in \sigma(r)$

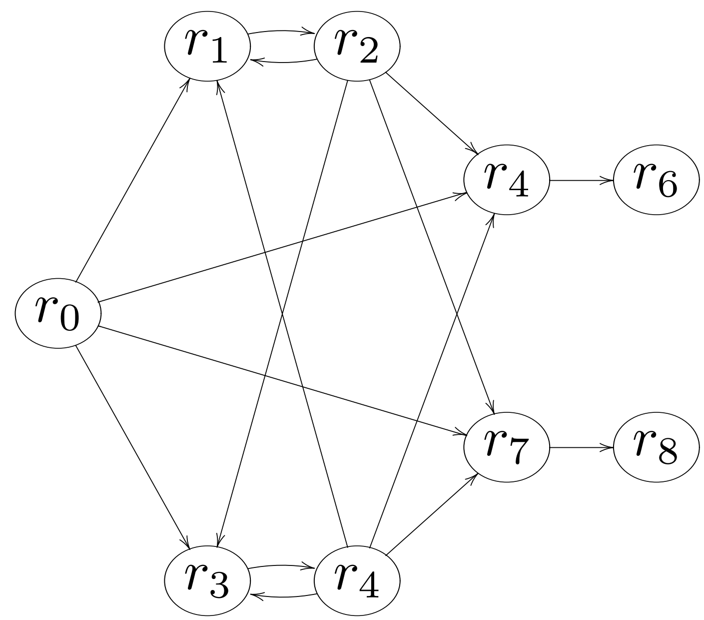
$S \Longrightarrow_{r_1} w_1 \Longrightarrow_{r_2} w_2 \ldots \Longrightarrow_{r_k} w_k = w$ – derivation in G
if and only if
$q_1 q_2 \ldots q_k$ is a sequence of nodes along a path in H

Control in programmed grammars corresponds to
control by node sequences along paths in graphs

Programmed Grammars – Another Interpretation II

$$G = (\{S, A, B\}, \{a, b\}, S, \{q_0, q_1, ;q_8\}, P, \{r_0, r_1, r_2, \ldots, r_8\})$$

$$\begin{aligned}
r_0 &= (q_0, S \to AB, \{q_1, q_3, q_5, q_7\}, \emptyset),\\
r_1 &= (q_1, A \to aA, \{q_2\}, \emptyset),\\
r_2 &= (q_2, B \to aB, \{q_1, q_3, q_5, q_7\}, \emptyset),\\
r_3 &= (q_3, A \to bA, \{q_4\}, \emptyset),\\
r_4 &= (q_4, B \to bB, \{q_1, q_3, q_5, q_7\}, \emptyset),\\
r_5 &= (q_5, A \to a, \{q_6\}, \emptyset),\\
r_6 &= (q_6, B \to a, \emptyset, \emptyset),\\
r_7 &= (q_7, A \to b, \{q_8\}, \emptyset),\\
r_8 &= (q_8, B \to b, \emptyset, \emptyset),
\end{aligned}$$



Valence Grammar – Definition

Definition ([11]) :

i) A valence grammar over a monoid is a quintuple $G = (N, T, S, P, (M, \circ))$ where

— N, T and S are specified as in a context-free grammar,

— $(M, \circ)$ is a monoid with neutral element e,

— P is a finite set of pairs $r = (p, m)$ with a context-free rule p and $m \in M$.

ii) For $x, y \in V_G^*$, $k, l \in M$, we say that (x, k) directly derives (y, l), written as $(x, k) \Longrightarrow (y, l)$, iff there is a pair $r = (A \to w, m) \in P$ such that

— $x = x'Ax''$ and $y = x'wx''$ for some $x', x'' \in V_G^*$ and

— $l = k \circ m$.

iii) The language $L(G)$ generated by G is defined as

$$L(G) = \{w \mid w \in T^*,\ (S, e) \Longrightarrow^* (w, e)\}.$$

Valence Grammar – Examples

$G = (\{S, A, B\}, \{a, b\}, S, \{r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8\}, (\mathbf{Q}, \cdot))$

$r_0 = (S \to AB, 1)$, $r_1 = (A \to aA, 2)$, $r_2 = (B \to aB, 1/2)$,
$r_3 = (A \to bA, 3)$, $r_4 = (B \to bB, 1/3)$, $r_5 = (A \to a, 2)$,
$r_6 = (B \to a, 1/2)$, $r_7 = (A \to b, 3)$, $r_8 = (B \to b, 1/3)$

$L(G) = \{w_1 w_2 \mid w_1 \in \{a, b\}^*,\ w_2 \in Perm(w_1)\}$

$G' = (\{S, A, B\}, \{a, b\}, S, \{r'_0, r'_1, r'_2, r'_3, r'_4, r'_5, r'_6, r'_7, r'_8\}, (\mathbf{Z}, +))$

$r'_0 = (S \to AB, 0)$, $r'_1 = (A \to aA, 2)$, $r'_2 = (B \to aB, -2)$,
$r'_3 = (A \to bA, 3)$, $r'_4 = (B \to bB, -3)$, $r'_5 = (A \to a, 2)$,
$r'_6 = (B \to a, -2)$, $r'_7 = (A \to b, 3)$, $r'_8 = (B \to b, -3)$

$L(G') = \{w_1 w_2 \mid w_1, w_2 \in \{a, b\}^+, 2\#_a(w_1) + 3\#_b(w_1) = 2\#_a(w_2) + 3\#_b(w_2)\}$

Valence Grammars versus other Grammars

Additive valence grammar – $(M, \circ) = (\mathbf{Z}, +)$
Multiplicative valence grammar – $(M, \circ) = (\mathbf{Q}, \cdot)$

λaV – family of all languages generated by additive valence grammars
aV – family of all languages generated by additive valence grammars without erasing rules

λmV – family of all languages generated by multiplicative valence grammars
mV – family of all languages generated by multiplicative valence grammars without erasing rules

Theorem :
$aV = \lambda aV \subset mV = \lambda mV = uV$

Conditional Grammars – Definition

Definition ([8]) :
i) A conditional grammar is a quadruple $G = (N, T, S, P)$ where
— N, T and S are specified as in a context-free grammar, and
— P is a finite set of pairs $r = (p, R)$ where p is a context-free production and R is a regular set over V_G.

ii) For $x, y \in V_G^*$, we say that x directly derives y, written as $x \Longrightarrow y$, iff there is a pair $r = (A \to w, R) \in P$ such that $x = x'Ax''$ and $y = x'wx''$ for some $x', x'' \in V_G^*$ and $x \in R$.

iii) The language $L(G)$ generated by G is defined as

$$L(G) = \{w \mid w \in T^*, \; S \Longrightarrow^* w\}.$$

Conditional Grammars – Examples

$G = (\{S, A, B, A', B'\}, \{a, b\}, S, \{r_0, r_1, \ldots r_8\})$

$r_0 = (S \to AB, S)$,

$r_1 = (A \to aA', V^*BV^*)$, $\quad r_2 = (A \to bA', V^*BV^*)$,

$r_3 = (B \to aB', V^*aA'V^*$, $\quad r_4 = (B \to bB', V^*bA'V^*)$,

$r_5 = (A' \to A, V^*B'V^*)$, $\quad r_6 = (A' \to \lambda, V^*B'V^*)$,

$r_7 = (B' \to B, V^*AV^*)$, $\quad r_8 = (B' \to \lambda, T^*B'T^*)$,

$L(G) = \{ww \mid w \in \{a, b\}^+ \}$

$G' = (\{S, S', A, B\}, \{a\}, S, \{r_1, r_2, r_3, r_4, r_5, r_6\})$

$r_1 = (S \to S'B, SA^*)$, $\quad r_2 = (A \to BB, S'B^+A^+)$,

$r_3 = (S' \to S, S'B^+$, $\quad r_4 = (B \to A, SA^*B^+)$,

$r_5 = (S \to a, SA^*)$, $\quad r_6 = (A \to a, a^+A^+)$

$L(G') = \{a^{2^n} \mid n \geq 0\}$

Semi-Conditional Grammar – Definition

Definition ([10]) :

i) A semi-conditional grammar is a quadruple $G = (N, T, S, P)$ where

— N, T and S are specified as in a context-free grammar, and

— P is a finite set of triples $r = (p, R, Q)$ where p is a context-free production and R and Q are disjoint finite sets of words over V_G.

R (Q) are called the permitted (forbidden) context of r or p, respectively.

ii) For $x, y \in V_G^*$, we say that x directly derives y, written as $x \Longrightarrow y$, iff there is a triple $r = (A \rightarrow w, R, Q) \in P$ such that

— $x = x'Ax''$ and $y = x'wx''$ for some $x', x'' \in V_G^*$,

— any word of R is a subword of x, and no word of Q is a subword of x.

iii) The language L(G) generated by G is defined as

$$L(G) = \{w \mid w \in T^*,\ S \Longrightarrow^* w\}$$

Semi-Conditional Grammar – Examples

$G = (\{S, S', S''\}, \{a\}, S, \{r_1, r_2, r_3, r_4\})$

$r_1 = (S \to S'S', \emptyset, \{SS', S'', a\}), \quad r_2 = (S' \to S'', \emptyset, \{S'S'', S, a\}),$
$r_3 = (S'' \to S, \emptyset, \{S''S, S', a\}, \quad r_4 = (S \to a, \emptyset, \{Sa, S', S''\})$

$L(G) = \{a^{2^n} \mid n \geq 0\}$

$G' = (\{S, S', S'', A\}, \{a\}, S, \{r_1, r_2, r_3, r_4, r_5\})$

$r_1 = (S \to S'S', \emptyset, \{S'', A\}), \quad r_2 = (S' \to S'', \emptyset, \{S\}), \quad r_3 = (S'' \to S, \emptyset, \{S'\}),$
$r_4 = (S \to A, \emptyset, \{S', S''\}), \quad r_5 = (A \to a, \emptyset, \{S\})$

$L(G') = \{a^{2^n} \mid n \geq 0\}$

Random Context Grammar

Definition ([13]) :

A random context grammar is a semi-conditional grammar where the permitting and forbidden contexts of all productions are subsets of the set of nonterminals.

A permitting (forbidden, respectively) random context grammar is a random context grammar where all forbidden (permitting, respectively) contexts are empty.

Example :

$G = (\{S, A, A', A_a, A_b, B, B'\}, \{a, b\}, S, \{r_0, r_1, \ldots r_{10}\})$

$r_0 = (S \to AB, \emptyset, \emptyset), \quad r_1 = (A \to aA_a, \{B\}, \emptyset), \quad r_2 = (A \to bA_b, \{B\}, \emptyset),$

$r_3 = (B \to aB', \{A_a\}, \emptyset), \quad r_4 = (B \to bB', \{A_b\}, \emptyset),$

$r_5 = (A_a \to A, \{B'\}, \emptyset), \quad r_6 = (A_b \to A, \{B'\}, \emptyset), \quad r_7 = (B' \to B, \{A\}, \emptyset)$

$r_8 = (A \to A'', \{B\}, \emptyset), \quad r_9 = (B \to \lambda, \{A''\}, \emptyset), \quad r_{10} = (A'' \to \lambda, \emptyset, \emptyset)$

$L(G) = \{ww \mid w \in \{a, b\}^* \}$

Notations

λC	–	family of all languages generated by conditional grammars
C	–	family of all languages generated by conditional grammars without erasing rules
λsC	–	family of all languages generated by semi-conditional grammars
sC	–	family of all languages generated by semi-conditional grammars without erasing rules
λRC	–	family of all languages generated by random context grammars
RC	–	family of all languages generated by random context grammars without erasing rules
λpRC	–	family of all languages generated by permitting random context grammars
pRC	–	family of all languages generated by permitting random context grammars without erasing rules

Generative Capacity

Theorem :

i) $\lambda C = \lambda sC = RE$

ii) $C = sC = CS$

Theorem :

i) $CF \subset pRC \subset RC = M_{ac} \subset \lambda RC$.

ii) $pRC \subseteq \lambda pRC \subset \lambda RC = \lambda M_{ac}$.

iii) $pRC \subseteq M$

iv) $\lambda pRC \subseteq \lambda M$

Ordered Grammar – Definition

Definition ([8]) :
i) An ordered grammar is a quadruple $G = (N, T, S, P)$ where
— N, T and S are specified as in a context-free grammar and
— P is a finite (partially) ordered set of context-free production.

ii) For $x, y \in V_G$, we say that x directly derives y, written as $x \Longrightarrow y$, if and only if there is a production $p = A \to w \in P$ such that $x = x'Ax''$, $y = x'wx''$ and there is no production $q = B \to v \in P$ such that $p \prec q$ and B occurs in x.

iii) The language $L(G)$ generated by G is defined as

$$L(G) = \{w \mid w \in T^*, S \Longrightarrow^* w\}$$

Ordered Grammar – Example

$$G = (\{S, S', S'', A, Z\}, \{a\}, S, P)$$

$$\begin{tikzcd}
S'' \to Z & A \to Z & S' \to Z & S \to Z & \\
S \to S'S' \arrow[u] \arrow[ur] & S'' \to S \arrow[u] \arrow[ur] & S \to A \arrow[u] \arrow[ull] & S' \to S'' \arrow[u] & A \to a \arrow[ul]
\end{tikzcd}$$

$$L(G) = \{a^{2^n} \mid n \geq 0\}$$

Ordered Grammars versus Other Grammars

λfRC	–	family of all languages generated by forbidden random context grammars
fRC	–	family of all languages generated by forbidden random context grammars without erasing rules
λO	–	family of all languages generated by ordered grammars
O	–	family of all languages generated by ordered grammars without erasing rules

Theorem :

i) $O = fRC \subseteq \lambda O = \lambda fRC \subset RE$.

ii) $CF \subset O \subset rC_{ac}$

Indexed Grammar – Definition I

Definition ([6]) :
i) An indexed grammar is a quintuple $G = (N, T, I, S, P)$ where

- N, T and S are specified as in a context-free grammar,
- I is a finite set of finite sets of productions of the form $A \rightarrow w$ with $A \in N$ and $w \in V_G^*$, and
- P is a finite set of productions of the form $A \rightarrow \alpha$ with $A \in N$ and $\alpha \in (NI^* \cup T)^*$.

The elements of I are called indexes.

Indexed Grammar – Definition II

ii) For $x, y \in (NI^* \cup T)^*$, we say that x directly derives y, written as $x \Longrightarrow y$, if either

$$\begin{aligned}
&x = x_1 A\beta x_2 \text{ for some } x_1, x_2 \in (NI^* \cup T)^*,\ A \in N,\ \beta \in I^*,\\
&A \rightarrow X_1\beta_1 X_2\beta_2 \dots X_k\beta_k \in P,\\
&y = x_1 X_1\gamma_1 X_2\gamma_2 \dots X_k\gamma_k x_2\\
&\qquad\qquad \text{with } \gamma_i = \beta_i\beta \text{ for } X_i \in N \text{ and } \gamma_i = \lambda \text{ for } X_i \in T,\ 1 \le i \le k,
\end{aligned}$$

or

$$\begin{aligned}
&x = x_1 Ai\beta x_2 \text{ for some } x_1, x_2 \in (NI^* \cup T)^*,\ A \in N,\ i \in I,\ \beta \in I^*,\\
&A \rightarrow X_1 X_2 \dots X_k \in i,\\
&y = x_1 X_1\gamma_1 X_2\gamma_2 \dots X_k\gamma_k x_2\\
&\qquad\qquad \text{with } \gamma_i = \beta \text{ for } X_i \in N \text{ and } \gamma_i = \lambda \text{ for } X_i \in T,\ 1 \le i \le k.
\end{aligned}$$

Indexed Grammar – Definition III and a Result

$\Longrightarrow^*$ denotes the reflexive and transitive closure of $\Longrightarrow$.

iii) The language $L(G)$ generated by G is defined as

$$L(G) = \{w \mid w \in T^*,\ S \Longrightarrow^* w\}$$

λI	–	family of all languages generated by indexed grammars
I	–	family of all languages generated by indexed grammars without erasing rules

Theorem :
$CF \subset I = \lambda I \subseteq CS$.

Indexed Grammar – Examples

$G = (\{S, A, B\}, \{a, b, c, d\}, \{f\}, S, P)$

$f = \{B \to bB, B \to b\},$
$P = \{S \to aSf, S \to A, A \to cAd, A \to B\}$

$L(G) = \{a^n c^m b^n d^m \mid n \geq 1, m \geq 1\}$

$G' = (\{S, A\}, \{a, b\}, \{f_a, f_b, h\}, S, P)$

$f_a = \{B \to Ba\}, \quad f_b = \{B \to Bb\}, \quad h = \{B \to \lambda\},$
$P = \{S \to Ah, A \to aAf_a, A \to bAf_b, A \to B\}$

$L(G') = \{ww \mid w \in \{a, b\}^*\}$

Hierarchy of Languages Obtained by Regulated Rewriting

Theorem : The following equalities are valid:

$RE = \lambda M_{ac} = \lambda rC_{ac} = \lambda P_{ac} = \lambda RC = \lambda C = \lambda sC$,

$CS = C = sC$,

$\lambda M = \lambda rC = \lambda P$,

$M_{ac} = rC_{ac} = P_{ac} = RC$,

$M = rC = P$,

$uV = \lambda uV = mV = \lambda mV$,

$aV = \lambda aV$,

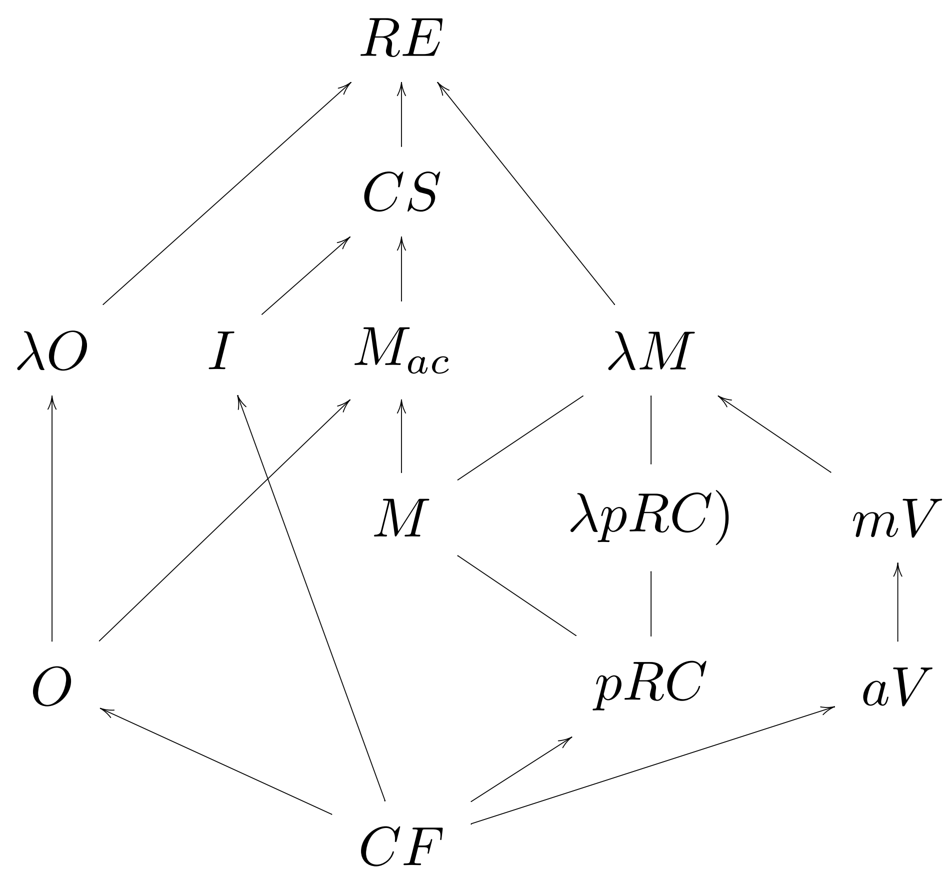
$\lambda O = \lambda fRC$,

$O = fRC$,

$I = \lambda I$

Theorem :
The opposite diagram holds.

If two families are connected by a line (an arrow), then the upper family includes (includes properly) the lower family; if two families are not connected then they are not necessarily incomparable.



Closure Properties

Theorem : The following table holds.

operation	M_{ac}	λM	M	uV	aV	I	λO	O	λpRC	pRC
union	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
intersection	?	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
complement	?	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
intersection by reg. sets	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
concatenation	$+$	$+$	$+$	$+$	$-$	$+$	$+$	$+$	$+$	$+$
Kleene-closure	$+$	?	?	$-$	$-$	$+$	$+$	$+$	$+$	$+$

operation	M_{ac}	λM	M	uV	aV	I	λO	O	λpRC	pRC
λ-free morphisms	+	+	+	+	+	+	+	+	+	+
(arbitrary) morphisms	–	+	–	+	+	+	+	?	+	?
inverse morphisms	+	+	+	+	+	+	+	+	+	+
λ-free gsm-mappings	+	+	+	+	+	+	+	+	+	+
gsm-mappings	–	+	–	+	+	+	+	?	+	?
derivative	+	+	+	+	+	+	+	+	+	+
quotient by reg. sets	–	+	–	+	+	+	+	+	+	?

Decision Results I

Theorem :

Let X be a family of grammars generating one of the families

$$\{M_{ac}, M, RC, O, \lambda M, \lambda RC, \lambda O, I, uV, aV\}.$$

Then the equivalence problem

Instance: grammars $G_1 \in X$ and $G_2 \in X$,
Answer: "Yes" if and only if $L(G_1) = L(G_2)$

and the problem

Instance: grammar $G \in X$
Answer: "Yes" if and only if G generates a context-free language.

are undecidable.

Decision Results II

Theorem : The following table holds.

grammar family	membership problem	emptiness problem	finiteness problem
I	NP-complete	$+$	$+$
λO	?	?	?
O	$+$, NP-hard	?	?
M_{ac}	$+$, NP-hard	-	-
λM	$+$	$+$, NP-hard	$+$, NP-hard
M	$+$	$+$, NP-hard	$+$, NP-hard
uV	$\in$LOGCFL	$+$, NP-hard	$+$, NP-hard
RC	$+$	$+$, NP-hard	$+$, NP-hard
aV	$\mathrm{DTIME}(n^4)$	$+$	$+$

Reachability Problem for Vector Addition System

An n-dimensional vector addition system is a couple (x_0, K) where $x_0 \in \mathbf{N}^n$ and K is a finite subset of $\mathbf{Z}^n$.

A vector $y \in \mathbf{N}^n$ is called reachable within (x_0, K) if and only if there are vectors $v_1, v_2, \ldots, v_t \in K$, $t \geq 1$, such that

$$x_0 + \sum_{i=1}^{j} v_i \in \mathbf{N}^n \text{ for } 1 \leq j \leq t \quad \text{and} \quad x_0 + \sum_{i=1}^{t} v_i = y.$$

The reachability problem

Instance: n-dimensional vector addition system (x_0, K), vector $y \in \mathbf{N}^n$
Answer: "Yes" if and only if y is reachable within (x_0, K)

is decidable (in exponential space).

3–Partition Problem

The 3-partition problem

Instance:	multiset $\{t_1, t_2, \ldots, t_{3m}\}$ of integers and integer t
Answer:	Yes, if there is partition $\{Q_1, Q_2, \ldots, Q_m\}$ of $\{t_1, t_2, \ldots, t_{3m}\}$ such that $\#(Q_i) = 3$ and $\sum_{s \in Q_i} s = t$ for $1 \leq i \leq m$.

is NP-complete.

Syntactic Complexity I

Definition :
i) For a grammar G, $Var(G)$ denotes the cardinality of its set of nonterminals.
ii) Let X be a family of languages and $\mathcal{G}(X)$ the corresponding set of grammars. For a language $L \in X$, we set

$$Var_X(L) = \min\{Var(G) \mid G \in \mathcal{G}(X),\ L(G) = L\}.$$

Theorem :
There is a sequence of context-free languages L_n, $n \geq 1$, such that

$$Var_{CF}(L_n) = n,$$
$$Var_M(L_n) \leq 3,\ Var_P(L_n) = 1,\ Var_{rC}(L_n) = 1,\ Var_{pRC}(L_n) \leq 8.$$

Syntactic Complexity II

Theorem :
i) For any recursively enumerable language L,

$$Var_{\lambda M_{ac}}(L) \leq 3 \quad \text{and} \quad Var_{\lambda P_{ac}}(L) \leq 3.$$

ii) $Var_{\lambda M_{ac}}(\{a^n b^n c^m d^m e^p f^p \mid n, m, p \geq 1\}) = 3$
iii) There is a sequence of recursively enumerable languages L_n, $n \geq 1$, such that

$$f(n) \leq Var_{\lambda RC}(L_n) \leq [\log_2 n] + 3 \quad \mathit{for}\ n \geq 1$$

where f is an unbounded function from **N** into **N**.

Finite Index – Definitions

G – grammar

$D = S = w_0 \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow \ldots \Longrightarrow w_n = w$ – derivation of w in G

$Ind(G, w, D) = \max\{\#_N(w_i) \mid 0 \leq 1 \leq n\}$

$Ind(G, w) = \min\{Ind(G, w, D) \mid D$ is a derivation of w in $G\}$

$Ind(G) = \sup\{Ind(G, w) \mid w \in L(G)\}$

$Ind_X(L) = \min\{Ind(G) \mid G \in \mathcal{G}(X), L = L(G)\}$

$X_{fin} = \{L \mid L \in X, Ind_X(L) < \infty\}$

Families of Languages of Finite Index

Theorem :

i) All the following language families are equal to M_{fin}

P_{fin}, $(P_{ac})_{fin}$, λP_{fin}, $(\lambda P_{ac})_{fin}$,
rC_{fin}, $(rC_{ac})_{fin}$, λrC_{fin}, $(\lambda rC_{ac})_{fin}$,
λM_{fin}, $(M_{ac})_{fin}$, $(\lambda M_{ac})_{fin}$, RC_{fin}, λRC_{fin},

ii) $O_{fin} \subseteq M_{fin} \subseteq C_{fin}$

iii) $pRC_{fin} \subseteq M_{fin} \subset M$

iv) $aV_{fin} \subset uV_{fin} \subseteq M_{fin}$

Theorem :

Each language in $\mathcal{L}_{fin}(M)$ is semilinear.