

# Models of Computation

2: Programmed grammars, matrix grammars,  
random context grammars, Lindenmayer-systems

# Controlled context-free grammars

- **Question:** Is it possible to generate non-context-free languages with context-free grammars by specifying conditions on the applicability of production rules.
- **Answer: yes, e.g.**
  - Programmed grammars
  - Matrix grammars
  - Random context grammars

# Programmed Grammars

- A **context-free programmed grammar** is a 4-tuple  $G = (N, T, P, S)$ , where
  - $N$  and  $T$  are disjoint finite alphabets,
  - $S \in N$  is the start symbol (axiom),
  - $P$  is a finite set of ordered triples of the form  $r = (p, \sigma, \varphi)$ ,  
where  $p$  is a context-free rule,  $\sigma, \varphi \subseteq P$ ,
  - $\sigma$  is the **success field** of  $r$ ,  $\varphi$  is the **failure field** of  $r$ .
  - If  $r = (p, \sigma, \emptyset)$ , for all rules  $r \in P$ , then the grammar  $G$  is **without appearance checking**, otherwise, **with appearance checking**.

# Programmed Grammars

- Let  $G = (N, T, P, S)$  be a programmed context-free grammar
- If  $u, v \in (N \cup T)^*$  are two consecutive sentences (strings) in a derivation (the  $i-1$ -st. and  $i$ -th, where  $i \geq 0$ ) and the  $i$ -th applied rule is  $r_i = (A \rightarrow w, \sigma, \varphi)$ , then exactly one of the following hold
  - if  $u = xAy$ , for some  $x, y \in (N \cup T)^*$ , then  $v = xwy$ , and the  $i+1$ -st rule  $r_{i+1}$  applied in the derivation (if exists)  $r_{i+1} \in \sigma$ . (I.e. the next applied rule must be from the success set.)
  - if  $u$  does not contain  $A$ , then  $v = u$ , and the  $i+1$ -st rule  $r_{i+1}$  applied in the derivation (if exists) is  $r_{i+1} \in \varphi$ . (I.e. the next applied rule must be from the failure set.)
- Notation:  $u \Rightarrow v$

# Programmed Grammars

- Let  $G = (N, T, P, S)$  a programmed context-free grammar.  
The **language  $L(G)$  generated by  $G$**  is:  
$$L(G) := \{ w \in T^* \mid S \Rightarrow^* w \},$$
  
where  $\Rightarrow^*$  is the reflexive, transitive closure of the relation  $\Rightarrow$ .

# Programmed Grammar

- Example:

Let  $G = (N, T, P, S)$  be a programmed grammar, where  $N = \{S, A\}$ ,  $T = \{a\}$ , and  $P = \{r_1, r_2, r_3\}$ , where

- $r_1 = (S \rightarrow AA, \{r_1\}, \{r_2\})$ ,
- $r_2 = (A \rightarrow S, \{r_2\}, \{r_1, r_3\})$ , and
- $r_3 = (S \rightarrow a, \{r_3\}, \emptyset)$ .

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

# Programmed Grammar

Let  $G = (\{S, A\}, \{a\}, S, \{r_1, r_2, r_3\})$ , where

$$r_1 = (S \rightarrow AA, \{r_1\}, \{r_2\})$$

$$r_2 = (A \rightarrow S, \{r_2\}, \{r_1, r_3\})$$

$$r_3 = (S \rightarrow a, \{r_3\}, \emptyset)$$

The derivation for the string *aaaa* is as follows:

$$\begin{aligned} S &\Rightarrow_{r_1} AA \Rightarrow_{r_1} AA \Rightarrow_{r_2} SA \Rightarrow_{r_2} SS \Rightarrow_{r_2} SS \\ &\Rightarrow_{r_1} AAS \Rightarrow_{r_1} AAAA \Rightarrow_{r_1} AAAA \\ &\Rightarrow_{r_2} SAAA \Rightarrow_{r_2} SSAA \Rightarrow_{r_2} SSSA \Rightarrow_{r_2} SSSS \Rightarrow_{r_2} SSSS \\ &\Rightarrow_{r_3} aSSS \Rightarrow_{r_3} aaSS \Rightarrow_{r_3} aaaS \Rightarrow_{r_3} aaaa \Rightarrow_{r_3} aaaa \end{aligned}$$

As can be seen from the derivation and the rules, each time  $r_1$  and  $r_2$  succeed, they feed back to themselves, which forces each rule to continue to rewrite the string over and over until it can do so no more. Upon failing, the derivation can switch to a different rule. In the case of  $r_1$ , that means rewriting all *Ss* as *AAs*, then switching to  $r_2$ . In the case of  $r_2$ , it means rewriting all *As* as *Ss*, then switching either to  $r_1$ , which will lead to doubling the number of *Ss* produced, or to  $r_3$  which converts the *Ss* to *as* then halts the derivation. Each cycle through  $r_1$  then  $r_2$  therefore either doubles the initial number of *Ss*, or converts the *Ss* to *as*. The trivial case of generating *a*, in case it is difficult to see, simply involves vacuously applying  $r_1$ , thus jumping straight to  $r_2$  which also vacuously applies, then jumping to  $r_3$  which produces *a*.

Source: [https://en.wikipedia.org/wiki/Controlled\\_grammar](https://en.wikipedia.org/wiki/Controlled_grammar)

# Matrix Grammar

- A **context-free matrix grammar with appearance checking** is a 5-tuple  $G = (N, T, M, S, \mathcal{F})$ , where
  - $N$  and  $T$  are disjoint finite alphabets,
  - $S \in N$  is the start symbol (axiom),
  - $M = \{m_1, m_2, \dots, m_n\}$ ,  $n \geq 1$ , is a finite set of sequences  $m_i = (p_{i1}, \dots, p_{ik(i)})$ ,  $k(i) \geq 1$ ,  $1 \leq i \leq n$ , where each  $p_{ij}$ ,  $1 \leq i \leq n$ ,  $1 \leq j \leq k(i)$ , is a context-free rule, and
  - $\mathcal{F} \subseteq \{p_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq k(i)\}$  is a subset of rules of sequences in  $M$ .
- The elements of  $M$  are called matrices.

# Matrix Grammar

- A matrix grammar  $G = (N, T, M, S, \mathcal{F})$  is **without appearance checking**, if and only if  $\mathcal{F} = \emptyset$ .

# Matrix Grammar

- Let  $G = (N, T, M, S, \mathcal{F})$  be matrix grammar and  $w, w' \in (N \cup T)^*$ . Then  $w'$  can be derived from  $w$  according to a matrix  $m_i : (A_{i1} \rightarrow v_{i1}, \dots, A_{ik(i)} \rightarrow v_{ik(i)}) \in M, 1 \leq i \leq n, k(i) \geq 1$ , (denoted as:  $w \Rightarrow_{mi} w'$ ), if and only if there exist words  $w_{i1}, \dots, w_{ik(i)+1} \in (N \cup T)^*$ , s.t.  $w = w_{i1}$ ,  $w' = w_{ik(i)+1}$  and for all  $i$  and  $j$ , where  $1 \leq i \leq n, 1 \leq j \leq k(i)$ ,
  - either  $w_{ij} = w'_{ij} A_{ij} w''_{ij}$  and  $w_{ij+1} = w'_{ij} v_{ij} w''_{ij}$ ,
  - or  $A_{ij}$  does not appear  $w_{ij}$  and  $w_{ij} = w_{ij+1}$ , and  $A_{ij} \rightarrow v_{ij} \in \mathcal{F}$ .

# Matrix Grammar

- Let  $G = (N, T, M, S, \mathcal{F})$  be a matrix grammar.  
The **language**  $L(G)$  **generated by**  $G$  is:  
$$L(G) = \{w \in T^* \mid S \Rightarrow_{m_{j_1}} y_1 \Rightarrow_{m_{j_2}} y_2 \Rightarrow_{m_{j_3}} \dots \Rightarrow_{m_{j_s}} w, 1 \leq j_i \leq r, 1 \leq i \leq s\}.$$
- Example: Let  $G = (N, T, M, S, \emptyset)$  be a matrix grammar without appearance checking, where  $N = \{S, A, B\}$ ,  $T = \{a, b\}$ , and  $M = \{m_1, m_2, m_3, m_4, m_5\}$ , where  
 $m_1 = (S \rightarrow AB),$   
 $m_2 = (A \rightarrow bA, B \rightarrow bB),$   
 $m_3 = (A \rightarrow b, B \rightarrow b),$   
 $m_4 = (A \rightarrow aA, B \rightarrow aB),$  and  
 $m_5 = (A \rightarrow a, B \rightarrow a).$   
Then  $L(G) = \{ww \mid w \in \{a, b\}^+\}.$

# Random context grammar

- A **random context grammar** is a 4-tuple  $G = (N, T, P, S)$ , where  
 $N$  and  $T$  are disjoint finite alphabets,  
 $S \in N$  is the start symbol (axiom),  
 $P$  is a finite set of ordered triples of the form  $(p, Q, R)$ ,  
where  $p$  is a context-free rule,  $Q, R \subseteq N$ .

# Random context grammar

- Let  $G = (N, T, P, S)$  be a random context grammar. The word  $y$  can be **derived** from  $x$ ,  $x, y \in (N \cup T)^*$ , (notation:  $x \Rightarrow y$ ), if
  - $x = x'Ax''$ ,  $y = x'wx''$ , for some words  $x', x'' \in (N \cup T)^*$  and
  - For all  $(A \rightarrow w, Q, R) \in P$ , if all symbols of  $Q$  appear in  $x'x''$ , and no symbol of  $R$  appear in  $x'x''$ .
- Remark:  
 $Q$  is called the **permitting context** of  $(A \rightarrow w, Q, R)$  and  
 $R$  is called the **forbidding context** of  $(A \rightarrow w, Q, R)$ .

# Random context grammar

- Let  $G = (N, T, P, S)$  be a random context grammar.  
The language  $L(G)$  **generated** by the grammar  $G$  is:  
 $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}.$
- Example:  
Let  $G = (N, T, P, S)$  be a random context grammar, where  
 $N = \{S, X, Y, A\}$ ,  $T = \{a\}$ , and  $P = \{r_1, r_2, r_3, r_4, r_5\}$ , where  
 $r_1 = (S \rightarrow XX, \emptyset, \{Y, A\})$ ,  
 $r_2 = (X \rightarrow Y, \emptyset, \{S\})$ ,  
 $r_3 = (Y \rightarrow S, \emptyset, \{X\})$ ,  
 $r_4 = (S \rightarrow A, \emptyset, \{X, Y\})$ ,  
 $r_5 = (A \rightarrow a, \emptyset, \{S\})$ .  
Then  $L(G) = \{a^{2^n} \mid n \geq 0\}.$

# Random context grammar

- A random context grammar generating the language  $\{a^{2^n} \mid n \geq 0\}$ .

Let  $G = (\{S, X, Y, A\}, \{a\}, S, \{r_1, r_2, r_3, r_4, r_5\})$ , where

$$r_1 = (S \rightarrow XX, \emptyset, \{Y, A\})$$

$$r_2 = (X \rightarrow Y, \emptyset, \{S\})$$

$$r_3 = (Y \rightarrow S, \emptyset, \{X\})$$

$$r_4 = (S \rightarrow A, \emptyset, \{X, Y\})$$

$$r_5 = (A \rightarrow a, \emptyset, \{S\})$$

Consider now the production for  $aaaa$ :

$$\begin{aligned} S &\Rightarrow_{r_1} XX \Rightarrow_{r_2} YX \Rightarrow_{r_2} YY \Rightarrow_{r_3} SY \Rightarrow_{r_3} SS \\ &\Rightarrow_{r_1} XXS \Rightarrow_{r_1} XXXX \Rightarrow_{r_2} YXXX \Rightarrow_{r_2} YYXX \Rightarrow_{r_2} YYYX \Rightarrow_{r_2} YYYY \\ &\Rightarrow_{r_3} SYYY \Rightarrow_{r_3} SSYY \Rightarrow_{r_3} SSSY \Rightarrow_{r_3} SSSS \\ &\Rightarrow_{r_4} ASSS \Rightarrow_{r_4} AASS \Rightarrow_{r_4} AAAS \Rightarrow_{r_4} AAAA \\ &\Rightarrow_{r_5} aAAA \Rightarrow_{r_5} aaAA \Rightarrow_{r_5} aaaA \Rightarrow_{r_5} aaaa \end{aligned}$$

Source: [https://en.wikipedia.org/wiki/Controlled\\_grammar](https://en.wikipedia.org/wiki/Controlled_grammar)

# Language families

- $\mathcal{L}(\text{PR}_{\text{ac}})$  denotes the **class of programmed grammars with  $\varepsilon$ -free rules with appearance checking**.
- $\mathcal{L}(\text{PR}_{\text{ac}}^{\varepsilon})$  denotes the class of arbitrary programmed grammars with appearance checking.
- If the grammar is without appearance checking, the index ac is omitted.
- $\mathcal{L}(\text{MAT}_{\text{ac}})$ ,  $\mathcal{L}(\text{MAT}_{\text{ac}}^{\varepsilon})$  are the **classes of matrix grammars** with and without  $\varepsilon$ -free rules with appearance checking, respectively.
- $\mathcal{L}(\text{RC}_{\text{ac}})$ ,  $\mathcal{L}(\text{RC}_{\text{ac}}^{\varepsilon})$  are the **classes of random context grammars** with and without  $\varepsilon$ -free rules with appearance checking, respectively.
- **Theorem 1** [Dassow, Paun, 2012]:  
The following relations hold:  
 $\mathcal{L}_2 \subset \mathcal{L}(\text{PR}_{\text{ac}}) = \mathcal{L}(\text{MAT}_{\text{ac}}) = \mathcal{L}(\text{RC}_{\text{ac}}) \subset \mathcal{L}_1$  and  
 $\mathcal{L}(\text{PR}_{\text{ac}}^{\varepsilon}) = \mathcal{L}(\text{MAT}_{\text{ac}}^{\varepsilon}) = \mathcal{L}(\text{RC}_{\text{ac}}^{\varepsilon}) = \mathcal{L}_0$ .

# L-system

- A **0L-system (non-interacting Lindenmayer system, or L-system)** is a triple  $G = (V, P, w)$ , where
  - $V$  is a finite alphabet,
  - $P$  is a finite set of context-free rewriting rules (or production rules), and
  - $w \in V^+$  is the start state (or axiom or initiator).
  - For every  $a \in V$ , there exists a rule  $a \rightarrow x \in P$  (We say  $P$  is complete).
- Remark: For any symbol  $a \in V$ , which does not appear on the left hand side of a production in  $P$ , the identity production  $a \rightarrow a$  is assumed; these symbols are called constants or terminals.

# L-rewriting

- For words  $z_1, z_2 \in V^*$ ,  $z_1$  can be **rewritten** to  $z_2$  regarding  $G$ , denoted by  $z_1 \Rightarrow z_2$ , if  $z_1 = a_1 a_2 \dots a_r$ ,  $z_2 = x_1 x_2 \dots x_r$ , for some  $a_i \rightarrow x_i \in P$ ,  $1 \leq i \leq r$ .
- Remark: As many rules as possible are applied simultaneously. This differentiates an L-system from a language generated by a classical formal grammar.

# L-system, generated language

- Let  $G = (V, P, w)$  be a 0L-system. The language  $L(G)$  **generated** by  $G$  is:  
 $L(G) = \{z \in V^* \mid w \Rightarrow^* z\}$ , where  
 $\Rightarrow^*$  is the reflexive transitive closure of  $\Rightarrow$ .

# L-system, generated language

- Example: Let  $G = (V, P, w)$  be a 0L-system, where  
 $V = \{a\}$ ,  
 $P = \{a \rightarrow a^2\}$ , and  
 $w = a^3$ .  
Then  $L(G) = \{a^{3 \cdot 2^n} \mid i \geq 0\}$

# L-system, generated language

- Example (fractal, binary tree): Let  $G = (V, P, w)$  be a 0L-system, where  
 $V = \{0, 1, [, ]\}$ ,  
 $P = \{1 \rightarrow 11, 0 \rightarrow 1[0]0\}$ , and  
 $w = 0$ .

It produces the sequence:

$$w = w_0 = 0$$

$$w_1 = 1[0]0$$

$$w_2 = 11[1[0]0]1[0]0$$

$$w_3 = 1111[11[1[0]0]1[0]0]11[1[0]0]1[0]0$$

...

This string can be drawn as an image

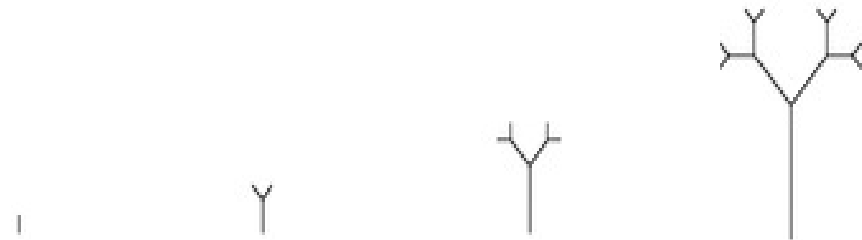
by interpreting the symbols as follows:

0: draw a line segment ending in a leaf

1: draw a line segment

[ : push position and angle, turn left 45 degrees

] : pop position and angle, turn right 45 degrees



# Family of languages generated by 0L-systems

- $\mathcal{L}(0L)$  denotes the family of languages generated by 0L-systems.

# E0L-systems

- An **E0L-system (Extended 0L-system)** is a 4-tuple  $G = (V, T, P, w)$ , where  $G = (V, P, w)$  is a 0L-system and  $T$  is an alphabet of terminal symbols.
- **Derivation**  $\Rightarrow_G$  (short  $\Rightarrow$ ) and  $\Rightarrow^*$  are defined similarly to 0L-systems.
- The **language**  $L(G)$  generated by  $G$  is:  
 $L(G) = \{z \in T^* \mid w \Rightarrow^* z\}.$
- $\mathcal{L}(\text{E0L})$  denotes the **family of languages** generated by E0L-systems.

# E0L-systems, generated language

- Example:  
Let  $G = (V, T, P, w)$  be an E0L-system, where  
 $V = \{a, b\}$ ,  
 $T = \{b\}$ ,  
 $P = \{a \rightarrow b, a \rightarrow bb, b \rightarrow b\}$ , and  
 $w = a$ .  
Then  $L(G) = \{b, bb\}$ .

# D0L-systems

- A **D0L-system (deterministic 0L-system)** is a 0L-system, if for every  $a \in V$  there is exactly one rule  $a \rightarrow x, x \in V^*$ .
- If the axiom is replaced by a finite language, then we obtain 0L-system (D0L-system) with a finite number of axioms, denoted as **F0L-system (FD0L-system)**.
- Remark: Since the set of production rules  $P$  of a D0L-system  $G = (V, P, w)$  defines a homomorphism  $h : V \rightarrow V^*$ , the notation  $G = (V, h, w)$  is also used.

# T0L-systems

- A **T0L-system** is an  $(n+2)$ -tuple  $G = (V, P_1, \dots, P_n, w)$ ,  $n \geq 1$ , where each  $G_i = (V, P_i, w)$ ,  $1 \leq i \leq n$ , is a 0L-system.
- The language  $L(G)$  **generated** by  $G$  is:  
$$L(G) = \{z \in V^* \mid w \Rightarrow_{G_1} w_1 \Rightarrow_{G_2} \dots \Rightarrow_{G_m} w_m = z, \\ 1 \leq i_j \leq n, 1 \leq j \leq m\}.$$
- $\mathcal{L}(\text{T0L})$  denotes the **family** of languages generated by T0L-systems

# T0L-systems

- Example: Let  $G = (V, P_1, P_2, w)$  be a T0L-system, where  
 $V = \{a\}$ ,  
 $P_1 = \{a \rightarrow a^2\}$ ,  $P_2 = \{a \rightarrow a^3\}$ , and  
 $w = a$ .
- Then  $L(G) = \{a^i \mid i = 2^m 3^n, m, n \geq 0\}$ .

# Literature

- Handbook of Formal Languages, G. Rozenberg, A. Salomaa, (eds.), Springer–Verlag, Berlin–Heidelberg, 1997.
- Gy. E. Révész, Introduction to Formal Languages, Dover Publications, Inc., New York, 2012.
- G. Rozenberg, A. Salomaa, The mathematical theory of L systems, Vol. 90., Academic Press, 1980.
- J. Dassow, Gh. Paun. Regulated rewriting in formal language theory, Springer Publishing Company, Inc., 2012.