Models of Computation

1: Basics, Languages

Basics, terminology

- **Alphabet**: a finite, non-empty set of symbols/letters.
- **Words** or **strings** over V: Finite sequences of the elements of an alphabet V.
- V^* : the **set of words** over V including the **empty word** (ε).
- $V^+ = V^* \setminus \{\varepsilon\}$: the **set of non-empty words** over V.
- The **length** of a word $u = t_1 \ldots t_n$ is the numer of letters u, denoted by $|u| = n$.
  - Length of the empty set ε is 0 ($|\varepsilon| = 0$).

- Example:
  Let $V = \{a, b\}$, then ab and *baaabb* are words over V.

Basics, terminology

- Let V be an alphabet and let u and v be words over V (i.e., $u,v \in V^*$). Then the word uv is the **concatenation** of u and v.

- $|uv| = |u| + |v|$.

- Example:
  Let $V = \{a, b\}$, $u = ab$ and $v = baabb$ words over V.
  Then $uv = abbaabb$.

Basics, terminology

Properties

- The concatenation is associative, but in general not commutative.
  - if $u, v \in V^*$, $u \neq v$, then uv differs from vu, unless V consists of only one letter (not commutative).
  - if $u, v, w \in V^*$, then $u(vw) = (uv)w$ (associative).
- V^* is **closed** for the operation of concatenation (i.e. for any $u, v \in V^*$, $uv \in V^*$ holds).
- The concatenation is an operation with **identity element**, or **neutral element**, the neutral element is ε (i.e., for any $u \in V^*$, $u = u\varepsilon = \varepsilon u$).

Basics, terminology

- Let i be a non-negative integer and u be a word over V ($u \in V^*$). The ***i*-th power** u^i of the word u is the concatenation of i instances of u.
- Convention: $u^0 = \varepsilon$.

- Example:
  Let $V = \{a, b\}$ and $u = abb$ be a word above V.
  Then $u^0 = \varepsilon$, $u^1 = \mathrm{abb}$, $u^2 = abbabb$, $u^3 = abbabbabb$, ...

Basics, terminology

- Let u and v be words over V. The words u and v are **equa**l, if as sequences of letters, they are equal element-by-element, i.e., $|u|=|v|$ and for all $i = 1,\dots,|u|$, the i-th letter of u and the i-th letter of v are equal.
- Let V be an alphabet and u and v be words over V.
  The word u is a **subword** (or **substring**) of v,
  if $v = xuy$, for some $x, y \in V^*$.
- A word u is a **proper subword** (or **proper substring**) of a word v if at least one of x or y is not empty, i.e. if $xy \neq \varepsilon$.
- If $x = \varepsilon$, then u is the **prefix** of v.
- If $y = \varepsilon$, then u is the **suffix** of v.

Basics, terminology

- Example:
  Let $V = \{a, b\}$ and $u = abb$.
  - Subwords of u: $\varepsilon, a, b, ab, bb, abb$.
  - Proper subwords of u: $\varepsilon, a, b, ab, bb$.
  - Prefixes of u: ε, a, ab, abb.
  - Suffixes of u: ε, b, bb, abb.

Basics, terminology

- Let u be a word over the alphabet V. The **reverse** (or **mirror**) word u^{-1} of u is the word obtained, s.t. the letters of u are written in reverse order.
- Let $u = a_1 \ldots a_n$, $a_i \in V$, $1 \leq i \leq n$. Then $u^{-1} = a_n \ldots a_1$.
- $(u^{-1})^{-1} = u.$
- $(u^{-1})^i = (u^i)^{-1}$ also holds, where $i = 1, 2, \ldots$

- Example:
  Let $V = \{a, b\}$ and $u = abba$ and $v = aabbba$
  Then $u^{-1} = abba$ (palindrome) and $v^{-1} = abbbaa$.

Basics, terminology

- Let V be an alphabet and L be an arbitrary subset of V^*. L is called a **language** over V.
- An **empty language** (a language that does not contain any words) is denoted by $\varnothing$.
- A language L over V is a **finite language** if it has a finite number of words. Otherwise, L is an **infinite language**.

- Example:
  Let $V = \{a, b\}$ be an alphabet.

  $L_1 = \{a, b, \varepsilon\}$.

  $L_2 = \{a^i b^i \mid i \geq 0\}$.

  $L_3 = \{uu^{-1} \mid u \in V^*\}$.

  $L_4 = \{(a^n)^2 \mid n \geq 1\}$.

  $L_5 = \{u \mid u \in \{a, b\}^+ , N_a(u) = N_b(u)\}$, where $N_a(u)$ and $N_b(u)$ denote the number of occurrences of symbols a and b in u, respectively.

  L_1 is a finite language, the others are infinite.

Basics, terminology

- A **generative grammar** G is a 4-tuple (N, T, P, S), where
  - N and T are disjoint finite alphabets (i.e. $N \cap T = \emptyset$).
  - The elements of N are called **nonterminal** symbols.
  - The elements of T are called **terminal** symbols.
  - $S \in N$ is the **start symbol** (axiom).
  - P is a finite set of ordered (x,y) pairs, where $x,y \in (N \cup T)^*$ and x contains at least one non-terminal symbol.
  - The elements of P are called **rewriting rules** (**rules** for short) or **productions**. $x \to y$ can be used instead of (x,y), where $\to \notin (N \cup T)$.

Basics, terminology

- Example:
  - $G_1 = (\{S, A, B\}, \{a, b, c\}, \{S \to c, S \to AB, A \to aA, B \to \varepsilon, abb \to aSb\}, S)$ is not a generative grammar.

  - $G_2 = (\{S, A, B, C\}, \{a, b, c\}, \{S \to a, S \to AB, A \to Ab, B \to \varepsilon, aCA \to aSc\}, S)$ is a generative grammar.

Basics, terminology

- Let $G = (N, T, P, S)$ be a generative grammar and let $u, v \in (N \cup T)^*$. The word v can be **derived directly** or **in one step** from u in G, denoted as $u \Rightarrow_G v$,
  if $u = u_1 x u_2$ and $v = u_1 y u_2$, where $u_1, u_2 \in (N \cup T)^*$ and $x \rightarrow y \in P$.

- Let $G = (N, T, P, S)$ be a generative grammar and $u, v \in (N \cup T)^*$. The word v can be **derived** from u in G , denoted as $u \Rightarrow^*_G v$,
  - if $u = v$, or
  - there exists a word $z \in (N \cup T)^*$, for which $u \Rightarrow^*_G z$ and $z \Rightarrow_G y$.
  - $\Rightarrow^*$ is the reflexive, transitive closure of $\Rightarrow$.
  - $\Rightarrow^+$ is the transitive closure of $\Rightarrow$.

Basics, terminology

- Let $G = (N, T, P, S)$ be a generative grammar and $u, v \in (N \cup T)^*$.
  The word v can be **derived in *k* steps** from u in G, $k \geq 1$, if there exists a sequence of words $u_1, \ldots, u_{k+1} \in (N \cup T)^*$, s.t. $u = u_1$, $v = u_{k+1}$, and $u_i \Rightarrow_G u_{i+1}$, $1 \leq i \leq k$.
- A word v can be **derived** from a word u in G if either $u = v$, or there is a number $k \geq 1$, s.t. v can be derived from u in k steps.

Basics, terminology

- Let $G = (N, T, P, S)$ be an arbitrary generative grammar. The **generated language** $L(G)$ by the grammar G is: $L(G) = \{w \mid S \Rightarrow^*_G w, w \in T^*\}$

- This means that $L(G)$ consists of words that are in T^* and can be derived from S by grammar G.

Basics, terminology

- Example:
  Let $G = (N, T, P, S)$ be a generative grammar, where
  $N = \{S, A, B\}$, $T = \{a, b\}$ and
  $P = \{S \rightarrow aSb, S \rightarrow ab, S \rightarrow ba\}$.
  Then $L(G) = \{a^n abb^n , a^n bab^n \mid n \geq 0\}$.

- Example:
  Let $G = (N, T , P, S)$ be a generative grammar, where
  $N = \{S, X , Y\}$, $T = \{a, b, c\}$ and
  $P = \{S \rightarrow abc, S \rightarrow aXbc, Xb \rightarrow bX , Xc \rightarrow Ybcc, bY \rightarrow Yb,$
  $aY \rightarrow aaX , aY \rightarrow aa\}$.
  Then $L(G) = \{a^n b^n c^n \mid n \geq 1\}$.

Basics, terminology

- Each grammar generates a language, but the same language can be generated by several different grammars.

- Two grammars are **equivalent** if they generate the same language.
- Two languages are **weakly equivalent**, if they differ only in the empty word.

Chomsky hierarchy

- Let $G = (N, T, P, S)$ be a generative grammar. G is generative grammar is of i-type, $i = 0, 1, 2, 3$, if the rule set P satisfies the following:
  - $i = 0$: no restriction.
  - $i = 1$: All rules of P have the form $u_1 A u_2 \rightarrow u_1 v u_2$, where $u_1, u_2, v \in (N \cup T)^*$, $A \in N$, and $v \neq \varepsilon$, except for a rule $S \rightarrow \varepsilon$, when such a rule exists in P.
    If P contains the rule $S \rightarrow \varepsilon$, then S does not occur on the right side of any rule.
  - $i = 2$: All rules of P are of the form $A \rightarrow v$, where $A \in N$ and $v \in (N \cup T)^*$.
  - $i = 3$: All rules of P are of the form either $A \rightarrow uB$ or $A \rightarrow u$, where $A, B \in N$ and $u \in T^*$.

Chomsky hierarchy

- A language L is of **type** i, where $i = 0, 1, 2, 3$, if it can be generated by a type i grammar.
- $\mathcal{L}_i$, $i = 0, 1, 2, 3$, denotes the class (family) of type i languages.

Chomsky hierarchy

- Type 0 grammars are called **phrase-structured** grammars.
- Type 1 grammars are **context-sensitive** grammars, since some occurrence of the nonterminal A can only be substituted with the word v in the presence of contexts u_1 and u_2.
- Type 2 grammars are **context-free** grammars, because the substitution of a nonterminal A with v is allowed in any context.
- Type 3 grammars are **regular** or **finite state** grammars.

- The classes of languages of type 0,1,2,3 are called **recursively enumerable**, **context-sensitive**, **context-free**, and **regular**, respectively.

Chomsky hierarchy

Linguistic background

”The cunning fox hastily ate the leaping frog.”

- $S \rightarrow A + B$ (S: sentence, A: noun phrase, B: verb phrase)
- $A \rightarrow C + D + E$ (C : article, D: adjective, E : noun)
- $B \rightarrow G + B$ (G : adverb)
- $B \rightarrow F + A$ (F : verb)
- $C \rightarrow$ the
- $D \rightarrow$ cunning
- $E \rightarrow$ fox
- $G \rightarrow$ hastily
- $F \rightarrow$ ate
- $D \rightarrow$ leaping
- $E \rightarrow$ frog

Chomsky hierarchy

Linguistic background

- + (space) – terminal symbol
- cunning ← → leaping , fox ← → frog (they are interchangeable, but the meanings are different)
- Sentence is syntactically correct

- It is not possible to describe the complete syntax of natural languages

Chomsky hierarchy

- It is oblivious that $\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_0$ and $\mathcal{L}_1 \subseteq \mathcal{L}_0$.
- It can also be shown that (Chomsky's hierarchy) following hold: $\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$.
- The inclusion relation between language class $\mathcal{L}_2$ and $\mathcal{L}_1$ is not oblivious from the definition of the corresponding grammars. However, $\mathcal{L}_1$ can be also generated by so called length-non-decreasing grammars. For all rules $p \to q$ of a length-non-decreasing grammar, $|p| \leq |q|$ is fulfilled, except $S \to \varepsilon$. If $S \to \varepsilon \in P$, then S does not occur in the right side of any rule of P.

Operations on Languages

- Let V be an alphabet and L_1, L_2 be languages over V (that is, $L_1 \subseteq V^*$ and $L_2 \subseteq V^*$)
  - **union:** $L_1 \cup L_2 = \{u \mid u \in L_1 \text{ or } u \in L_2\}$.
  - **intersection:** $L_1 \cap L_2 = \{u \mid u \in L_1 \text{ and } u \in L_2\}$.
  - **difference:** $L_1 - L_2 = \{u \mid u \in L_1 \text{ and } u \notin L_2\}$.

- Example:

  Let $V = \{a, b\}$ be an alphabet and $L_1 = \{a, b\}$ and $L_2 = \{\varepsilon, a, bbb\}$ languages over V. Then

  $L_1 \cup L_2 = \{\varepsilon, a, b, bbb\}$

  $L_1 \cap L_2 = \{a\}$

  $L_1 - L_2 = \{b\}$

Operations on Languages

- The **complement** of the language $L \subseteq V^*$ with respect to the alphabet V is the language $\overline{L} = V^* - L$.

- Example:
  Let $V = \{a\}$ be an alphabet and let $L = \{a^{4n} \mid n \geq 0\}$. Then $\overline{L} = V^* - \{a^{4n} \mid n \geq 0\}$.

Operations on Languages

- Let V be an alphabet and L_1, L_2 be languages over V (i.e. $L_1 \subseteq V^*$ and $L_2 \subseteq V^*$). The **concatenation** of L_1 and L_2 is $L_1L_2 = \{u_1u_2 \mid u_1 \in L_1, u_2 \in L_2\}$.

- Remark:
  The following equalities hold for every language L:
  $\varnothing L = L\varnothing = \varnothing$ and
  $\{\varepsilon\}L = L\{\varepsilon\} = L$.

Operations on Languages

- L^i denotes the ***i*-th iteration** of L (for the operation of concatenation), where $i \geq 1$. By convention, $L^0 = \{\varepsilon\}$.
- The **iterative closure** (or **Kleene closure**) of a language L is: $L^* = \bigcup_{i \geq 0} L^i$.
- The **positive closure** of L is: $L^+ = \bigcup_{i \geq 1} L^i$.

- Remark:
  Obviously, if $\varepsilon \in L$, then $L^+ = L^*$. Otherwise, $L^+ = L^* - \{\varepsilon\}$.

Operations on Languages

- Example (concatenation):
  Let $V = \{a, b\}$ and let
  $L_1 = \{a, b\}$, $L_2 = \{\varepsilon, a, bbb\}$,
  $L_3 = \{a^{4n}b^{4n} \mid n \geq 0\}$ and $L_4 = \{a^{7n}b^{7n} \mid n \geq 0\}$.
  Then
  - $L_1L_2 = \{a, b, aa, ba, abbb, bbbb\}$,
  - $L_3L_4 = \{a^{4n}b^{4n}a^{7m}b^{7m} \mid n \geq 0, m \geq 0\}$.

Operations on Languages

- Let V be an alphabet and $L \subseteq V^*$. Then the language $L^{-1} = \{u^{-1} \mid u \in L\}$ is the **mirror** (or **reversal**) of L.

- Remarks:
  - $(L^{-1})^{-1} = \mathrm{L}$,
  - $(L_1 \mathrm{L}_2 \ldots L_n)^{-1} = L_n^{-1} \ldots L_2^{-1} L_1^{-1}$,
  - $(L^i)^{-1} = (L^{-1})^i$, where $i \geq 0$, and
  - $(L^*)^{-1} = (L^{-1})^*$.

Operations on Languages

- Example (mirror, reversal):
  Let $V = \{a, b\}$ and $L = \{\varepsilon, a, abb\}$ be a language over V. Then
  $L^{-1} = \{\varepsilon, a, bba\}$.

Operations on Languages

- The **prefix of a language** $L \subseteq V^*$ is the language
  $\mathrm{PRE}(L) = \{ u \mid u \in V^*, uv \in L \text{ for some } v \in V^* \}$.

- Remark:
  By definition, $L \subseteq \mathrm{PRE}(L)$ for any language $L \in V^*$.

- The **suffix of a language** $L \subseteq V^*$ is the language
  $\mathrm{SUF}(L) = \{ u \mid u \in V^*, vu \in L \text{ for some } v \in V^* \}$.

Operations on Languages

- Let V_1 and V_2 be two alphabets. The mapping $h : V_1^* \rightarrow V_2^*$ is called a **homomorphism** if the following conditions hold:
  - for every word $u \in V_1^*$ there is exactly one word $v \in V_2^*$ for which $h(u) = v$.
  - $h(uv) = h(u)h(v)$, for all $u, v \in V_1^*$.

- Remarks:
  - It follows from the above conitions that $h(\varepsilon) = \varepsilon$.
    Namely, for all $u \in V_1^*$ holds $h(u) = h(\varepsilon u) = h(u\varepsilon)$.
  - For all words $u = a_1a_2 \dots a_n$, $a_i \in V_1$, $1 \leq i \leq n$, it holds that $h(u) = h(a_1)h(a_2) \dots h(a_n)$.
    I.e. it is sufficient to define the mapping h on the elements of V_1, this is automatically extended to V_1^*.

Operations on Languages

- A homomorphism $h : V_1^* \rightarrow V_2^*$ is **ε-free** if for all $u \in V_1^+$, $h(u) \neq \varepsilon$.

- Let $h : V_1^* \rightarrow V_2^*$ be a homomorphism.
  The ***h*-homomorphic image** of a language $L \in V_1^*$ is the language $h(L) = \{w \in V_2^* \mid w = h(u), u \in L\}$

- Example (homomorphism):
  Let $V_1 = V_2 = \{a, b\}$ be two alphabets. Let $h : V_1^* \rightarrow V_2^*$ be a homomorphism, s.t. $h(a) = bbb$, $h(b) = ab$ and $L = \{a, abba\}$.
  Then $h(L) = \{bbb, bbbababbbb\}$.

Operations on Languages

- A homomorphism h is called an **isomorphism** if following holds:
  for any $u, v \in V_1^*$, if $h(u) = h(v)$, then $u = v$.

- Example (isomorphism – binary representation of decimal numbers):
  $V_1 = \{0, 1, 2, \ldots, 9\}$, $V_2 = \{0, 1\}$,
  $h(0) = 0000$, $h(1) = 0001$, . . . , $h(9) = 1001$

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, $\sigma, \varphi \subseteq P$,
  - σ is the **success field** of r, φ 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 $L(G) = \{a^{2^n} | n \geq 0\}$.

Programmed Grammar

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

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

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 *S*s as *AA*s, then switching to r_2. In the case of r_2, it means rewriting all *A*s as *S*s, then switching either to r_1, which will lead to doubling the number of *S*s produced, or to r_3 which converts the *S*s to *a*s then halts the derivation. Each cycle through r_1 then r_2 therefore either doubles the initial number of *S*s, or converts the *S*s to *a*s. 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

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),
  - $M = \{m_1, m_2, \ldots, m_n\}$, $n \geq 1$, is a finite set of sequences $m_i = (p_{i1}, \ldots, p_{ik(i)})$, $k(i) \geq 1$, $1 \leq i \leq n$, 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 $\mathcal{F} = \varnothing$.

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 $m_i : (A_{i1} \to v_{i1}, \ldots, A_{ik(i)} \to v_{ik(i)}) \in M$, $1 \le i \le n$, $k(i) \ge 1$, (denoted as: $w \Rightarrow_{mi} w'$), if and only if there exist words $w_{i1}, \ldots, w_{ik(i)+1} \in (N \cup T)^*$, s.t. $w = w_{i1}$, $w' = w_{ik(i)+1}$ and for all i and j, where $1 \le i \le n$, $1 \le j \le k(i)$,
  - 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 $A_{ij} \to v_{ij} \in \mathcal{F}$.

Matrix Grammar

- Let $G = (N, T, M, S, \mathcal{F})$ be a matrix grammar.
  The **language** $L(G)$ **generated by** G is:
  $L(G) = \{w \in T^* \mid S \Rightarrow_{m_{j1}} y_1 \Rightarrow_{m_{j2}} y_2 \Rightarrow_{m_{j3}} \ldots \Rightarrow_{m_{js}} w,\ 1 \leq j_i \leq r,\ 1 \leq i \leq s\}$.

- Example: Let $G = (N, T, M, S, \varnothing)$ be a matrix grammar without appearace checking, where $N = \{S, A, B\}$, $T = \{a, b\}$, and
  $M = \{m_1, m_2, m_3, m_4, m_5\}$, where
  $m_1 = (S \to AB)$,
  $m_2 = (A \to bA, B \to bB)$,
  $m_3 = (A \to b, B \to b)$,
  $m_4 = (A \to aA, B \to aB)$, and
  $m_5 = (A \to a, B \to a)$.
  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 appearin $x'x''$, 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
  $r_1 = (S \to XX, \varnothing, \{Y, A\})$,
  $r_2 = (X \to Y, \varnothing, \{S\})$,
  $r_3 = (Y \to S, \varnothing, \{X\})$,
  $r_4 = (S \to A, \varnothing, \{X, Y\})$,
  $r_5 = (A \to a, \varnothing, \{S\})$.
  Then $L(G) = \{a^{2^n} | n \geq 0\}$.

Random context grammar

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

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

$$r_1 = (S \to XX, \emptyset, \{Y, A\})$$
$$r_2 = (X \to Y, \emptyset, \{S\})$$
$$r_3 = (Y \to S, \emptyset, \{X\})$$
$$r_4 = (S \to A, \emptyset, \{X, Y\})$$
$$r_5 = (A \to a, \emptyset, \{S\})$$

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

Language families

- $\mathcal{L}(PR_{ac})$ denotes the **class of programmed grammars with ε-free rules with apearance checking**.
- $\mathcal{L}(PR^{\varepsilon}_{ac})$ denotes the class of arbitrary programmed grammars with apearance checking.
- If the grammar is without appearance checking, the index ac is omitted.
- $\mathcal{L}(MAT_{ac})$, $\mathcal{L}(MAT^{\varepsilon}_{ac})$ are the **classes of matrix grammars** with and without ε-free rules with apearance checking, respectively.
- $\mathcal{L}(RC_{ac})$, $\mathcal{L}(RC^{\varepsilon}_{ac})$ are the **classes of random context grammars** with and without ε-free rules with apearance checking, respectively.

- **Theorem 1** [Dassow, Paun, 2012]:
  The following relations hold:
  $\mathcal{L}_2 \subset \mathcal{L}(PR_{ac}) = \mathcal{L}(MAT_{ac}) = \mathcal{L}(RC_{ac}) \subset \mathcal{L}_1$ and
  $\mathcal{L}(PR^{\varepsilon}_{ac}) = \mathcal{L}(MAT^{\varepsilon}_{ac}) = \mathcal{L}(RC^{\varepsilon}_{ac}) = \mathcal{L}_0$.

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 $a \to x \in P$ (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 $a \to a$ 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 $z_1 = a_1 a_2 \ldots a_r$, $z_2 = x_1 x_2 \ldots x_r$, 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 $\Rightarrow$.

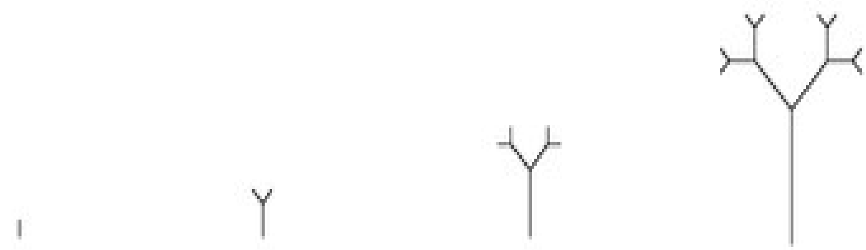
L-system, generated language

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

L-system, generated language

- Example (fractal, binary tree): Let $G = (V, P, w)$ be a 0L-system, where
  $V = \{0, 1, [,]\}$,
  $P = \{1 \to 11, 0 \to 1[0]0\}$, 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 (Extened 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:
  $L(G) = \{z \in T^* \mid w \Rightarrow^* z\}$.
- $\mathcal{L}$(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
  $V = \{a, b\}$,
  $T = \{b\}$,
  $P = \{a \to b, a \to bb, b \to b\}$, 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 $a \to x$, $x \in V^*$.

- 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 $h : V \to V^*$, the notation $G = (V, h, w)$ is also used.

T0L-systems

- A **T0L-system** is an $(n+2)$-tuple $G = (V, P_1, \ldots, P_n, w)$, $n \geq 1$, 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:
  $L(G) = \{z \in V^* \mid w \Rightarrow_{G_{i1}} w_1 \Rightarrow_{G_{i2}} \ldots \Rightarrow_{G_{im}} w_m = z,$
  $1 \leq i_j \leq n, 1 \leq j \leq m\}$.

- $\mathcal{L}$(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
  $V = \{a\}$,
  $P_1 = \{a \to a^2\}$, $P_2 = \{a \to a^3\}$, 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.