Formal Language and Automata Theory (CS21004)

Soumyajit Dey
CSE, IIT Kharagpur

Table of Contents

- The slide is just a short summary
- Follow the discussion and the boardwork
- Solve problems (apart from those we dish out in class)

Table of Contents

Formal Languages : alphabet

$\Sigma \rightarrow$ Alphabet, a finite non-empty set of symbols
'Strings' :: any possible *concatenation* of symbols $\in \Sigma$
Some concepts related to strings ::

- concatenation ($\circ$) of strings : $x \circ y$ (ignore the operator in general) $\rightarrow$ a generalization of concatenation of symbols in Σ
- length '| |' of a string (inductive definition) :
  let x be a string (defined over Σ) and $a \in \Sigma$. Then
  $\forall a \in \Sigma, |a| = 1, |x \circ a| = |xa| = |x| + 1.$

Formal Languages : string

Σ^* : set of all strings obtained by concatenating zero or more symbols from Σ.

- empty string : choose no symbol from Σ, denoted by λ or ϵ or $\perp$
- $|\lambda| = 0$
- $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$
- Although Σ is a finite set, both Σ^+ and Σ^* are infinite sets.

Formal Languages : string

- String concatenation is associative
- A Monoid is an algebraic structure formed by a set with an associative binary operation and an identity for the operation.
- $\langle \Sigma^*, \circ, \lambda \rangle$ is a Monoid

Σ^*

- Example : $\Sigma = \{a, b\}$
- $\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \cdots\}$
- The enumeration (ordering) is as per 'dictionary order' with a difference.
- strings smaller in length are placed earlier
- two strings equal in length are in dictionary order
- $\Rightarrow$ Lexicographic ordering of strings

Σ^*, lexicographic ordering

- The relation thus defined is a partial order over Σ^*
- In fact, it is a 'total order'
- The enumeration of Σ^* following standard dictionary ordering would be 'unfair'
- you cannot physically exhaust strings starting with 'a' and go strings that start with 'b'
- $\Rightarrow$ more on this later on

String ops

Let $\Sigma = \{a, b\}$

- String reversal : $(abaab)^R = baaba$; Note $(\sigma^R)^R = \sigma$
- String x is a prefix of the string σ if $\exists y$ such that $x \circ y = \sigma$
- Prefixes of 'aab' $= \{a, aa, aab\}$
- String x is a suffix of the string σ if $\exists y$ such that $y \circ x = \sigma$
- Suffixes of 'aab' $= \{aab, ab, b\}$

Formal Language

Given Σ

- any formal language $\subseteq \Sigma^*$
- can be finite as well as infinite
- Ex (inf language) : $\{a^n b^n \mid n \geq 0\}$

Formal Language

Languages are sets, can apply all legal set operations

- $L_1 \cup L_2,\ \ L_1 \cap L_2$
- $\overline{L} = \Sigma^* \setminus L$

Can 'lift' operators on strings to languages

- $L^R = \{w^R \,|\, w \in L\}$
- $L_1 \circ L_2 = \{x \circ y \,|\, x \in L_1 \wedge y \in L_2\}$

Formal Language : Closure

- $L^0 = \{\lambda\}$
- $L^1 = L$
- $L^2 = \{xy \mid x, y \in L\}$
- $\vdots$
- Star closure of a language : $L^* = L^0 \cup L^1 \cup L^2 \cup \cdots$
- Positive closure of a language : $L^+ = L^1 \cup L^2 \cup \cdots$
- Ex : $L = \{a^n b^n \mid n \geq 0\}$, $L^2 = \{a^n b^n a^m b^m \mid n, m \geq 0\}$, $L^R = \{b^n a^n \mid n \geq 0\}$

Table of Contents

Grammar

Natural language (english say) has a set of rules : decides whether a sentence is well formed.

- $\langle \mathit{sentence} \rangle \rightarrow \langle \mathit{noun\_phrase} \rangle\ \langle \mathit{predicate} \rangle$
- $\langle \mathit{noun\_phrase} \rangle \rightarrow \langle \mathit{article} \rangle\ \langle \mathit{noun} \rangle$
- $\langle \mathit{predicate} \rangle \rightarrow \langle \mathit{verb} \rangle$

Grammar

A grammar G is a quadruple $G = \langle V, T, S, P \rangle$

- V : finite set of variables/nonterminals
- T : set of terminals
- $S \in V$: start symbol
- P : set of 'production rules'

‘production rules’

P is a set of production rules. Let $x \in (V \cup T)^+$, $y \in (V \cup T)^*$.

- A production rule is of the form $x \mapsto y$. Rules $\in P$
- Production rules apply on strings $\in (V \cup T)^+$
- String $w = uxv$ **derives** string $z = uyv$, written as

$$w \Rightarrow z$$

- w_1 derives w_n : $w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_n$ is written as $w_1 \overset{*}{\Rightarrow} w_n$

Grammar n Languages

- For grammar $G = \langle V, T, S, P \rangle$, the set $L(G) = \{w \in T^* \mid S \overset{*}{\Rightarrow} w\}$ is the language generated by G.
- For any string $w \in L(G)$, there exists a derivation $S \Rightarrow w_1 \Rightarrow \cdots \Rightarrow w_n \Rightarrow w$
- $S, w_1, \cdots, w_n \in (V \cup T)^+$ are **sentential forms** of the derivation (do not contain w)

Grammar n Languages

Let $G = \langle V = \{S\}, T = \{a, b\}, S, P = \{S \to aSb \,|\, \lambda\}\rangle$.
Note that,

- $L(G) = \{a^n b^n \,|\, n \geq 0\}$
- all sentential forms look like $w_i = a^i S b^i$
- all sentential forms are of odd length
- In order to generate $a^i b^i$, apply rule $S \to aSb$ i times followed by $S \to \lambda$

Grammar n Languages

- For $L = \{a^n b^{n+1} \mid n \geq 0\}$, production rules can be $P = \{S \rightarrow Ab, A \rightarrow aAb|\lambda\}$
- Prove that with

$$P = \{S \rightarrow SS|\lambda|aSb|bSa\}$$

$$L(G) = \{w \mid n_a(w) = n_b(w)\}.$$

$L = \{w \mid n_a(w) = n_b(w)\}$

Base case is obvious. Let $P = \{S \rightarrow SS|\lambda|aSb|bSa\}$ generate strings in L upto length 2n. Consider $\sigma \in \Sigma^*$ with $n_a(\sigma) = n_b(\sigma)$ and $|\sigma| = 2n + 2$. Possibilities :

1. $\sigma = a\sigma' b$
2. $\sigma = b\sigma' a$
3. $\sigma = a\sigma' a$
4. $\sigma = b\sigma' b$

In 1, 2, $n_a(\sigma') = n_b(\sigma')$ and $|\sigma'| = 2n$. Hence,
$S \Rightarrow aSb \stackrel{*}{\Rightarrow} a\sigma' b = \sigma$ ($S \stackrel{*}{\Rightarrow} \sigma'$ as per induction hypothesis)
$S \Rightarrow bSa \stackrel{*}{\Rightarrow} b\sigma' a = \sigma$ ($S \stackrel{*}{\Rightarrow} \sigma'$ as per induction hypothesis)
Case 3, 4 ??

$L = \{w \mid n_a(w) = n_b(w)\}$

Scan the string left to right, count $+1$ if faced with a, -1 if faced with b. If case 3, after first a, count $= +1$, before last a, count $=$ -1. Count must cross 0 in between.

- $\sigma = a\sigma' a \Rightarrow \exists \sigma', \sigma'' \in L$ such that $\sigma = \sigma'\sigma''$.
- In that case, $S \Rightarrow SS \overset{*}{\Rightarrow} \sigma'\sigma''$.
- same argument for case 4.