I3307 - Theory of Computation

* **Alphabet:** (Σ) a finite set of symbols
  Binary alphabet is $\{0, 1\}$
  English alphabet is $\{a, b, \ldots, z\}$

* **String:** finite sequence of symbols from an alphabet.
  Note: the empty string λ contains no symbols.
  - length of a string is its length as a sequence. $|abbab| = 5$, $|\lambda| = 0$
  - Σ^*: set of all possible strings in an alphabet.
  - Concatenation ($\circ$): $u = a_1 a_2 \ldots a_n$, $v = b_1 b_2 \ldots b_m$, $u \circ v = uv = a_1 a_2 \ldots a_n b_1 b_2 \ldots b_m$
    - $\lambda u = u \lambda = u$
    - in general, $uv \neq vu$
    - $|uv| = |u| + |v|$
  - Reverse: $u = a_1 a_2 \ldots a_n$, $u^R = a_n a_{n-1} \ldots a_1$
    - $(uv)^R = v^R u^R$
  - Power: $u^n = \underbrace{uuu\ldots u}_{n \text{ times}}$
    - $u^0 = \lambda$
    - u palindrome $\Leftrightarrow$ u^n palindrome

* **Language:** set of strings defined over an alphabet.
  - union: $L_1 \cup L_2 = \{w / w \in L_1 \text{ or } w \in L_2\}$ inclusive-or: can be either or both.
  - intersection: $L_1 \cap L_2 = \{w / w \in L_1 \text{ and } w \in L_2\}$
  - Complement: $(\overline{L}) = \{w / w \notin L\}$ unless specified, consider the complement with respect to Σ^*.
  - Concatenation: $L_1 \circ L_2 = \{w = xy \mid x \in L_1, y \in L_2\}$
  - Reverse: $L_1^R = \{w^R ; w \in L\}$
  - Power: $L^n = \underbrace{L \circ L \circ L \ldots \circ L}_{n \text{ times}}$
    - $L^n = L \circ L^{n-1}$
    - $L^0 = \lambda$

  Note: just like strings, we can denote $L \circ k$ as Lk for simplicity.

  - Kleene Closure: $L^* = \bigsqcup_{i \in \mathbb{N}} L^i = \{\lambda, \underbrace{\ldots, \ldots, \ldots}_{L^1}, \underbrace{\ldots \ldots}_{L^2} \ldots\}$
  - Positive Closure: $L^+ = L^* \setminus \{\lambda\}$
    - $L^* = L^* L^*$
    - $(L^*, \circ)$ is closed $\Leftarrow$ L^* is closed under concatenation. $x \in L^*$, $y \in L^* \Rightarrow x \circ y = xy \in L^*$

Reminder:
to prove something false: counter example
to prove it true: Method 1: taking general values (let $x \in$, $y \in$, ...)
Method 2: Induction
- prove true for base case
- assume true for k^i
- prove true for k^{i+1}

***Regular Expressions:** pattern that matches a set of strings.
It uses (*) the kleene star that indicates the pattern repeats 0 or more times.

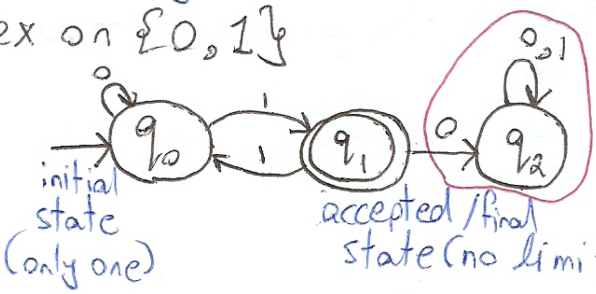
ex: $ab(a^*)$ matches ab, aba, abaaa, etc...

$L(r)$ is the language of the set of strings matching "r".

- $L(r_1+r_2) = L(r_1) \cup L(r_2)$ ex: $(a+b)$ matches a or b.
- $L(r_1 r_2) = L(r_1)L(r_2) = \{xy \mid x \in L(r_1) \text{ and } y \in L(r_2)\}$
- $L(r^*) = \{x_1 x_2 x_3 \ldots x_n \mid x_i \in L(r)\}$

Note: $L((a+b)^*)$ = any string in $\{a,b\}^*$ $\neq$ $L(a^*+b^*)$ = any string in $\{a\}^*$ or $\{b\}^*$

***Automata:** ex on $\{0,1\}$



dead end: a state that can never send us to an accepted state. Not adding a transition is equivalent of that transition being a dead end.

Deterministic Final Automata (DFA):
only one transition can leave a state for the same symbol.

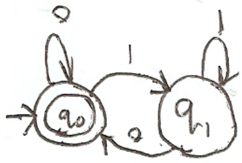
defined by $(Q, \Sigma, \delta, q_0, F)$

- Q → finite set of states
- Σ → input alphabet
- δ → transition function
- q_0 → initial state
- F → set of final states

ex:

δ	0	1
q_0	q_0	q_1
q_1	q_0	q_1

for



Regular Language: accepted by a certain DFA.

- L is regular $\Leftrightarrow$ $\bar{L}$ is regular.
- L_1 and L_2 are regular $\Rightarrow$ so is $(L_1 \cup L_2)$ and $(L_1 \cap L_2)$

Grammar: (V, Σ, R, S)

- V: set of variables
- Σ: set of terminals
- R: rules
- S: start variable

ex:
$S \to xS$
$S \to y$
$S \to yT$
$T \to z$

abbreviated:
$S \to xS \mid y \mid yT$
$T \to z$

$S \Rightarrow xS \Rightarrow xxS \Rightarrow xxxS \ldots$

at any point we can also replace S with y or yT which is yz then:

$S \Rightarrow xS \Rightarrow xy$
$\Rightarrow xyz$

$\therefore \{x, xy, xyz, xx, xxy, xxyz, xxx, xxxy, xxxyz, xxxx, \ldots\}$

Language defined by this grammar.

- A grammar is regular if the language defined by it is regular.

Nondeterministic Final Automata (NFA): multiple paths of the same symbol can leave each state.

[DFA is a type of NFA]

defined by $(Q, \Sigma, \Delta, q_0, F)$

Q: set of states
Σ: alphabet
Δ: transition function
q_0: initial state
F: set of final states

transition function $\Delta = Q \times \Sigma \rightarrow P(Q)$

$P(Q)$: powerset of Q: $\{\emptyset, \{q_1\}, \{q_2\}, \{q_1, q_2\}, \ldots\}$

NFAs can also have λ-transitions:

When traversing, taking a λ transition is optional

=> there are multiple ways to traverse an NFA with the same input string.

A string only has to match one path to a final state to be accepted by an NFA.