Alphabet: finite set of symbols.

Ex: Binary, $\Sigma = \{0,1\}$

Decimal Numbers, $\Delta = \{0,1,\dots,9\} \cup \{+ - .\}$

English Language, $\Gamma = \{a-z\} \cup \{A-Z\} \cup \{\sqcup\} \cup \{\text{punctuation symbols}\}$

space/blank

String: finite sequence of symbols from Σ.

Ex: $x = 011001$ (Σ)

$y = -2345.1$ (Δ)

$z = abracadabra$ (Γ)

- length, $|z| = 11$.
- empty string ϵ with $|\epsilon| = 0$.

$\Sigma^* \leftarrow$ set of all strings over Σ

supports concatenation operation

Ex: $\Sigma = \{a,b,c\}$

$\therefore \Sigma^* = \{\epsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, \dots\}$

If $x = a_1 a_2 \dots a_m$, $y = b_1 b_2 \dots b_n$ } $xy = a_1 \dots a_m b_1 \dots b_n$.

$a^0 = \epsilon$, $a^n = aaa\dots a$ (n times)

$\epsilon^n = \epsilon$, $a\epsilon^n = \epsilon^n a = a$

Concatenation on Σ^* → associative, → identity (ϵ), → closed.

Not commutative, No inverse

Language: $\mathcal{L} \subseteq \Sigma^*$ (set of strings)

→ Σ^* is countable

→ $\#\mathcal{L} = \mathcal{P}(\Sigma^*)$ is uncountable.

~~Computers cannot solve all problems.~~

Representation: → finite → enumerate, → infinite → cannot enum.

→ Require finite description ← Compute can solve Countable Prob

Computational problems { Given $L \subseteq \Sigma^*$ and $w \in \Sigma^*$, determine $w \in L$? (decision Prob)

1. English: set of strings of even length
2. Math: $\{w \in \{0,1\}^* \mid w \text{ is multiple of } 3\}$
3. Recursive: palindrome $- S_n \dots S_2 S_1 c S_1 S_2 \dots S_n$, factorial $- n * f(n-1)$
4. GRAMMARS
5. MACHINES → FSM, PDA, TM — Automata

⇒ [Next Lectures]

Prefix (Suffix) of strings:

$w \in \Sigma^*$, $u \in \Sigma^*$ is Prefix (suffix) of w if $w = uv$ (vu), for some $v \in \Sigma^*$

Operation over Strings : → Power of string, $w^n = \begin{cases} \epsilon & , n = 0 \\ w w^{n-1} & , n \geq 1 \end{cases}$

→ Concatenation (defined)

disjunction $(x+y)$.

Ex: $(01)^3 = 010101$

$0^3 1^3 = 000111$

$01^3 = 0111$

Operation over Languages : $L \subseteq \Sigma^*$ → sets

→ All set operations

$A \cup B$, $A \cap B$, $A - B$, $\bar{A} = \Sigma^* \setminus A$

→ Concatenation $AB = \{uv \mid u \in A, v \in B\}$

Ex: $A = \{a, ab\}$
$B = \{\epsilon, b\}$
$\therefore AB = \{a, ab, abb\}$

→ Powers : $A \subseteq \Sigma^*$

$A^n = \begin{cases} \{\epsilon\} & , n = 0 \\ AA^{n-1} & , n \geq 1 \end{cases}$

Ex: $A = \{a, ab\}$
$A^0 = \{\epsilon\}$
$A^1 = \{a, ab\}$
$A^2 = \{aa, aab, aba, abab\}$
- - -

→ Identities:

① $(A^*)^* = A^*$

② $\emptyset^* = \{\epsilon\}$

③ $A^* A^* = A^*$

④ $A^* = \{\epsilon\} \cup A^* A = \{\epsilon\} \cup A A^*$

Prove Yourself

Kleene Star (Asterics)

$A^* = \bigcup_{n \geq 0} A^n$, $A^+ = \bigcup_{n \geq 1} A^n$

$\Sigma^* = \bigcup_{n \geq 0} \{0,1\}^n$

▷ $\boxed{L} \subseteq \Sigma^*$, $w \in \Sigma^*$ → $w \in L$?

{Model whatever being accepted/Reject}

Memory (External)

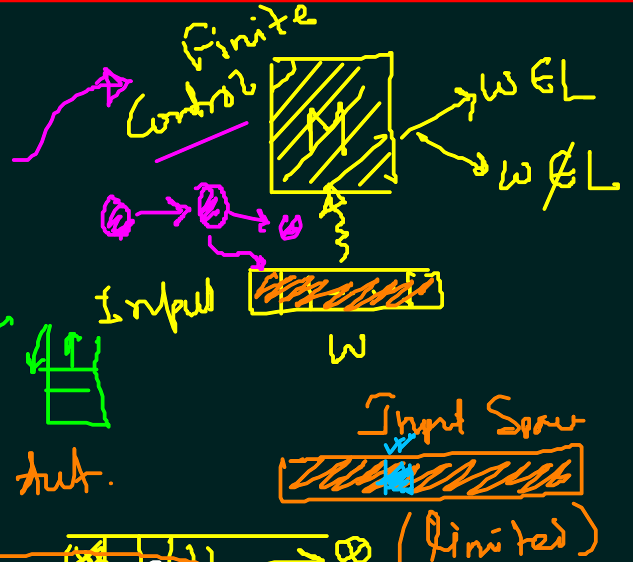
① Regular Grammars (Linear) → Finite Automata (No ext. Memory)

② Context-free Grammar → Push Down Automata (1 stack memory)

③ Context-Sensitive Grammar → Linear Bounded Aut.

④ Unrestricted grammar → Turing Machine

→ Recursive Language nor RE



Regular Languages : [Finite State Machines] (automata)

DFA vs. NFA

$\langle \underline{Q}, \underline{\Sigma}, \underline{q_0}, \delta, F \rangle$

$F \subseteq Q$

$\delta: Q \times \Sigma \to Q$ (DFA)

$\delta: Q \times \Sigma \to \mathcal{P}(Q)$ (NFA)

Ex: 00.....0 11 .....1 2.....2 + ∈

∈ ∈ ∈

+ 11....1 22....2

$L = \{0^* 1^* 2^*\}$

$\Sigma = \{0, 1, 2\}$

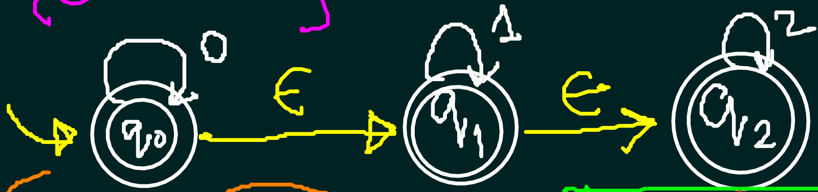
accept 00122 ✓

022 ✓

reject 1022 ✗

2100 ✗

2000 ✗

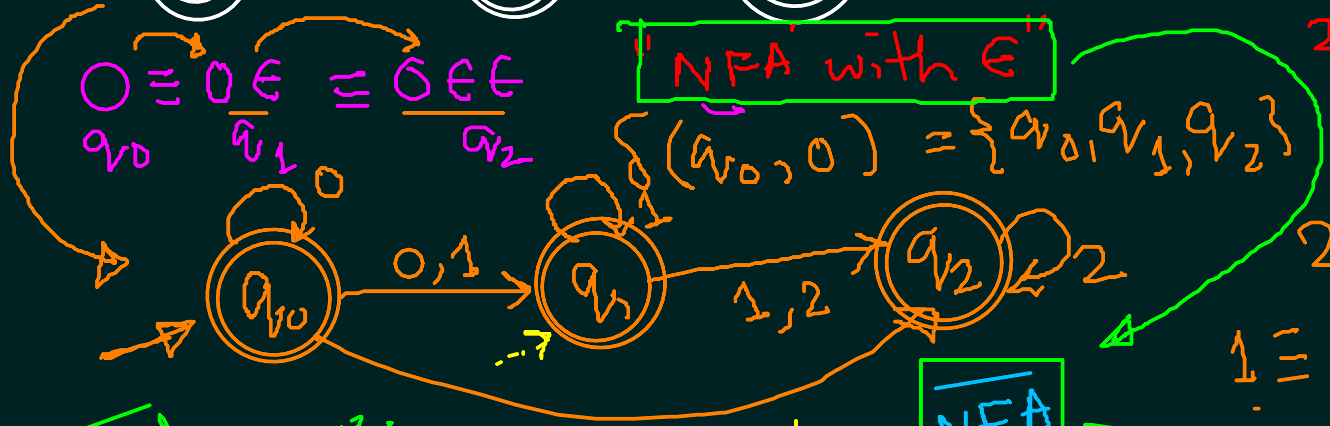


$0 \equiv 0\epsilon \equiv 0\epsilon\epsilon$

$q_0 \quad q_1 \quad q_2$

"NFA with ∈"

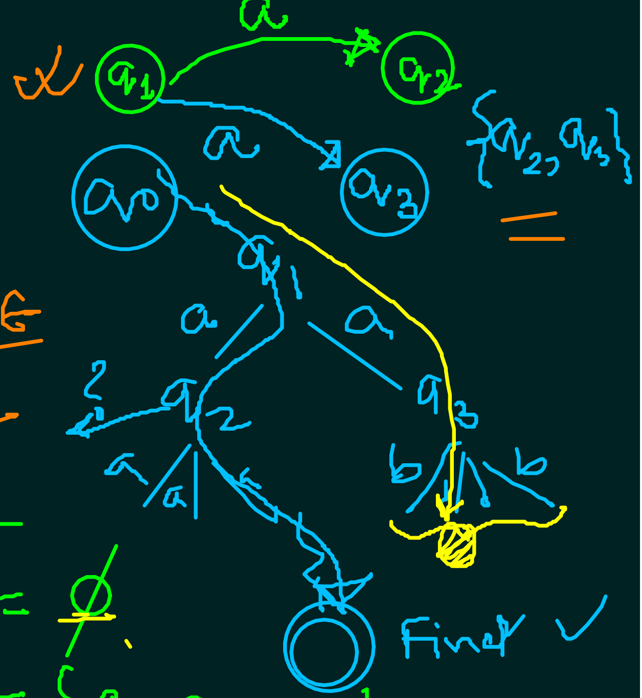
$\delta(q_0, 0) = \{q_0, q_1, q_2\}$



$2 \equiv \epsilon\epsilon 2$

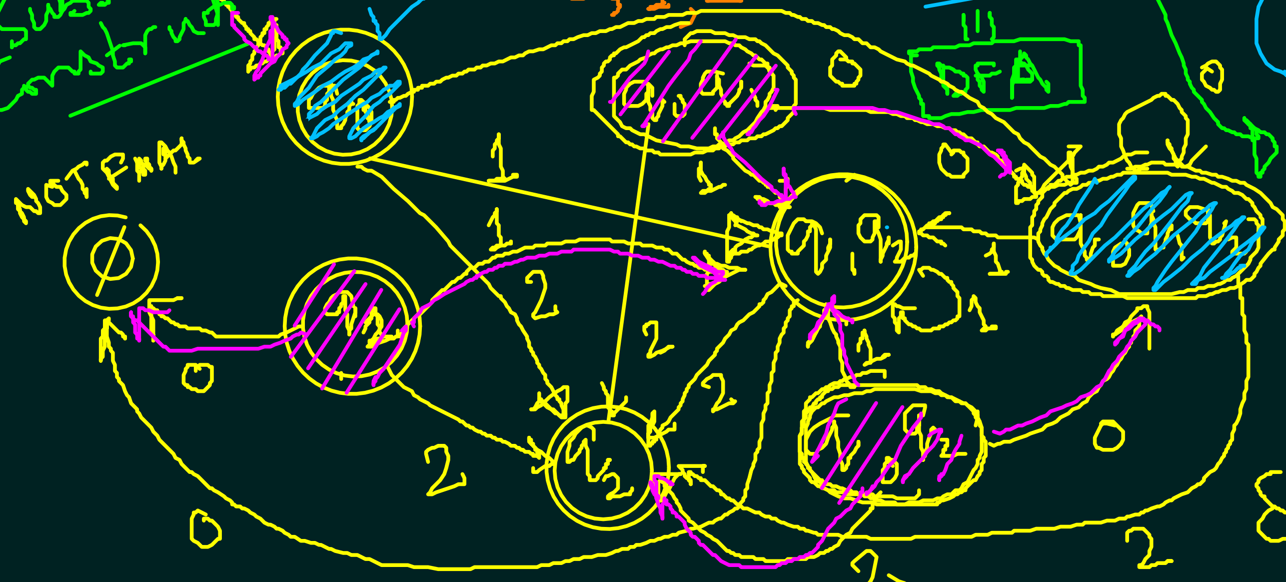
$1 \equiv \underset{q_1}{\underline{\epsilon 1}} \equiv \underset{q_2}{\underline{\epsilon 1 \epsilon}}$

(∈-closure)



Subset Construction

NFA ≡ DFA

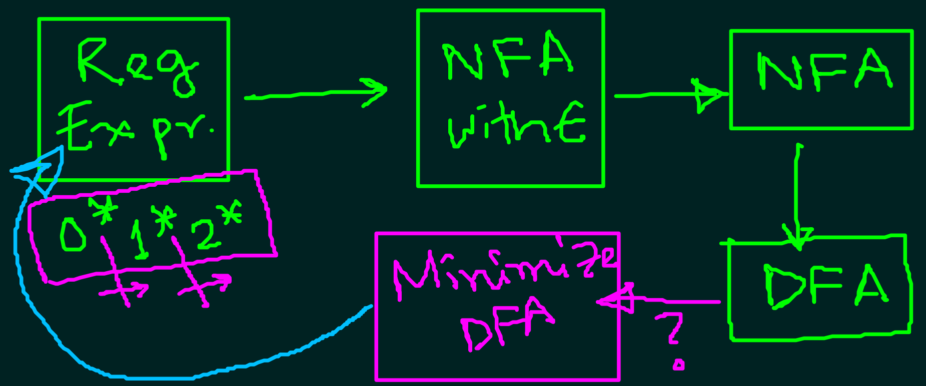


$\delta(q, a) = \emptyset$

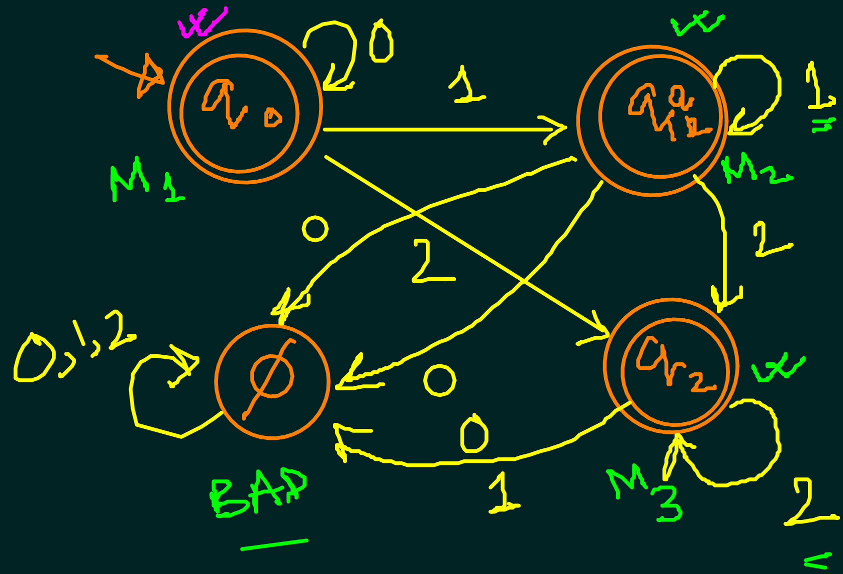
$= \{q_1, q_2 \ldots\}$

$= \{q'\}$

$\delta(\{q_0, q_1\}, a) = \underline{\delta(q_0, a)} \cup \underline{\delta(q_1, a)}$



- Remove Redundant States ✓
- Merge Equivalent States ✓



Minimize DFA $\equiv$ $0^*1^*2^*$