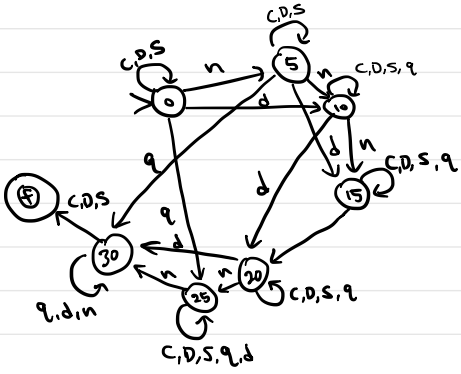
Use States: (total amount of money inserted)

Start state, final state

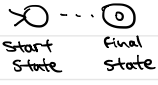


★ know how to do this.

← Finite Automaton

08/10

Deterministic Finite Automata (DFA)



Symbols : {n, d, q, C, D, S}

Alphabet Σ = {n, d, q, C, D, S}

We need:

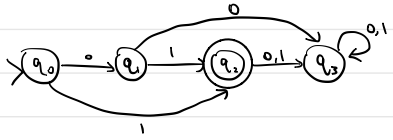
- A set of states Q
- Alphabet Σ
- Transition function δ (delta): $Q \times \Sigma \to Q$
- A Start State q_0 (only one)
- A set of final states F

$M = (Q, \Sigma, \delta, q_0, F)$

M: machine

☆Study this ex.

ex)



- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$

- δ

δ	0	1
q_0	q_1	q_2
q_1	q_3	q_2
q_2	q_3	q_3
q_3	q_3	q_3

transition table

- q_0
- $F = \{q_2\}$

ex) $w = 00$

instantaneous Configuration

$[q_0, 00] \vdash [q_1, 0] \vdash [q_3, \varepsilon]$

$\vdash$: yields; $[q_1, 0]$: read in one string at a time; ε: empty string

$\therefore$ No (Not in final state)

ex) $w = 01$

$[q_0, 01] \vdash [q_1, 1] \vdash [q_2, \varepsilon]$

$\therefore$ Yes (in final state)

*A string w is accepted by $M \iff [q_0, w] \vdash \vdash \vdash \vdash \vdash [q_f, \varepsilon]$ s.t. $q_f \in F$

$\iff [q_0, w] \overset{*}{\vdash} [q_f, \varepsilon]$ ☆

in our ex) Accepted Strings : 01, 1

↳ $L(M)$ is the language of M

☆ $\therefore L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$

Σ set of symbols

Σ^* all strings using symbols from Σ (all strings over Σ)

ex) Build a DFA that accepts those strings with two consequtive 1's

q_0 - last symbol not a 1

q_1 - last symbol is 1

q_2 - we have two consequtive 1's

i) $w = 0010$

$[q_0, 0010] \vdash [q_0, 010] \vdash [q_0, 10] \vdash [q_1, 0]$

$\vdash [q_0, \varepsilon]$

$\therefore 0010 \notin L(M)$

ii) 0110

$[q_0, 0110] \overset{*}{\vdash} [q_2, \varepsilon]$

$\therefore$ Yes, $0010 \in L(M)$

ex) $\Sigma = \{a, b\}$

accept those strings with even number of a's and odd number of b's

$q_{e,e}$ even a's, even b's

$q_{e,o}$ even a's, odd b's

$q_{o,e}$ odd a's, even b's

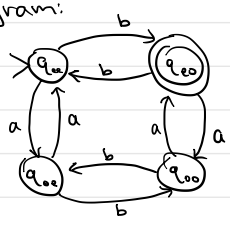
$q_{o,o}$ odd a's, odd b's

- $Q = \{q_{ee}, q_{eo}, q_{oe}, q_{oo}\}$

δ	a	b
q_{ee}	q_{oe}	q_{eo}
q_{eo}	q_{oo}	q_{ee}
q_{oe}	q_{ee}	q_{oo}
q_{oo}	q_{eo}	q_{oe}

- Start State: q_{ee} (0 # of a's & b's)
- $F = \{q_{eo}\}$

Diagram:



ex i) abaabab

$[q_{ee}, abaabab] \vdash [q_{oe}, baabab] \vdash [q_{oo}, aabab] \vdash [q_{eo}, abab]$
$\vdash [q_{oo}, bab] \vdash [q_{oe}, ab] \vdash [q_{ee}, b] \vdash [q_{eo}, \varepsilon]$
∴ yes

Definition:

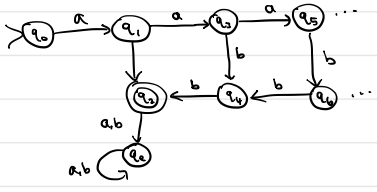
A language L (set of strings) is regular $\leftrightarrow \exists$ a DFA M s.t. $L = L(M)$

Non-Regular? Are there languages that are not regular?

$a \cdots a \; b \cdots b$

number of a's = number of b's

$L_1 = \{ab, aabb, aaabbb, \cdots\}$

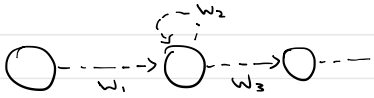


*DFA ← finite (finite set of states)

Assume that we can build a DFA M that accepts L_1. M must have a finite number of states (say n)

Let $x \in L_1$ s.t. length of $w > n$

the computation $[\,] \vdash \cdots$ must go through more than n states at least one state will be repeated.



- $u = w_1w_2w_2w_3 \in L_1$
- $w_1w_2\textcircled{w_2}w_3$ must be in L_1
- $aaa\cdots abb\cdots b$
  - Cannot include a's and b's
  - or else $\longrightarrow$ $aaababababbb\cdots$

- w_2 only a's $\rightarrow$ more a's than b's
- w_2 only b's $\rightarrow$ more b's than a's

$\therefore L = \{a^n b^n \mid n \geq 1,\ a,b \in \Sigma\}$ is not regular.

Regular Operations:

- Union: $L_1 \cup L_2$
- Concatenation: L_1L_2 $\quad L_1L_2 = \{w_1w_2 \mid w_1 \in L_1 \wedge w_2 \in L_2\}$
- Kleene Star: L^* (concat on itself)

$L^* = \{\varepsilon\} \cup L \cup LL \cup (LL)L \cdots$

$= \{\varepsilon\} \cup L \cup L^2 \cup L^3 \cup L^4 \cdots$

*These are sets, so duplicates aren't allowed.

ex) $L_1 = \{aa, abb, aba\}$

$L_2 = \{bb, baa, bab\}$

1) $L_1 \cup L_2 = \{aa, abb, aba, bb, baa, bab\}$ * Check for any duplicates

2) Concat $= L_1L_2 = \{aabb, aabaa, aabab, abbbb, abbbaa, abbbab, ababb, ababaa, ababab\}$

* Get rid of any duplicates

3) Kleene Star:

$L_1^* = \{\varepsilon,$ $aa, abb, aba,$ $aaaa, aaabb, aaaba, abbaa, abbabb, abbaba, abaaa, abaabb,$ $abaaba, \cdots\}$

(L_1; L_1^2; L_1^3)

$aaaaaa, aaaaabb$

*Concat with L_1

Σ^* Set of all possible strings

Property:

Regular languages are closed under:

1) Union
2) Concatenation
3) Kleene Star

If L_1 and L_2 are regular $\rightarrow$ $L_1 \cup L_2$ is regular

$L_1 L_2$ is regular

L_1^* is regular

· Deterministic

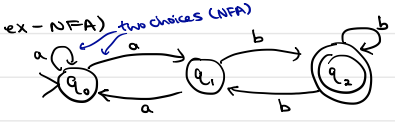
$q_i, a \rightarrow$ one choice q_j

· Non-Deterministic Finite Automata (NFA)

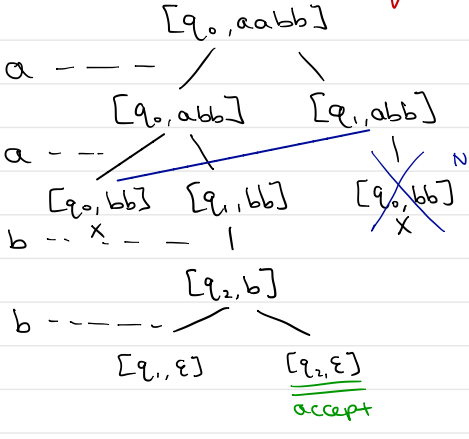
↳ Difference:

$\delta : Q \times \Sigma \rightarrow 2^Q$

ex - NFA)



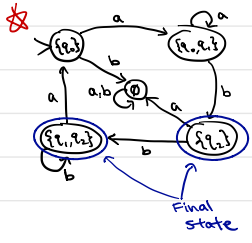
w = aabb



study this ex.

δ	a	b
q_0	$\{q_0, q_1\}$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_2\}$
$\{q_2\}$	$\emptyset$	$\{q_1, q_2\}$
$\{q_1, q_2\}$	$\{q_0\}$	$\{q_1, q_2\}$

→ Deterministic



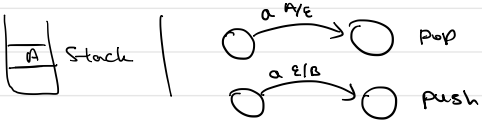
*has an accepting state of NFA

NFA ↔ DFA:

If L is accepted by a NFA → L is accepted by a DFA
→ L is regular.

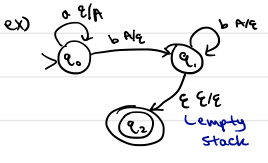
Notice: DFA is just a special case of NFA

Using stack: (DFA)



[q, ____, ____]
input Stack

ex)



push pop

[q₀, aaabbb, ε] ⊢

Push Down Automata (can accept Context free)

- More powerful
- Add little bit of memory → can do more complex operation like $a^n b^n$.

* Languages accepted by PDAs are called Context Free.

Grammar: Set of variables V
Set of terminals Σ
Start variable S
Set of productions P

* Grammar generate language, NOT accept languages.

ex) S → aSb
S → ab

Σ = {a, b}
V = {S}

→ S ⇒ aSb
⇒ aaSbb
⇒ aaaSbbb
⇒ aaaabbbb ⇒ $a^n b^n$

Grammar generates a language.

Language of a grammar:

$L(G) = \{w \in \Sigma^* \mid S^* \Rightarrow w\}$

ex)

variable

$S \rightarrow S+S$

$S \rightarrow S*S$

$S \rightarrow (S)$

$S \rightarrow x \mid y \mid z$ (or)

$(x+y)*z$

$S \Rightarrow S*S$

$\Rightarrow (S)*S \rightarrow (S+S)*S$

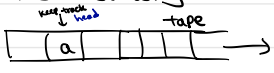
$\Rightarrow (x+y)*z$

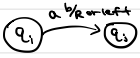
How Compiler works:

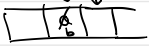
Phase 1: check basic structure variable → tokenize

Phase 2: Use grammar → Production

↳ More memory







$(q_2, a) \rightarrow (q_3, b, R)$

Church turing thesis

Turing Machine ≡ Computer with algorithms

↳ Model of Computation