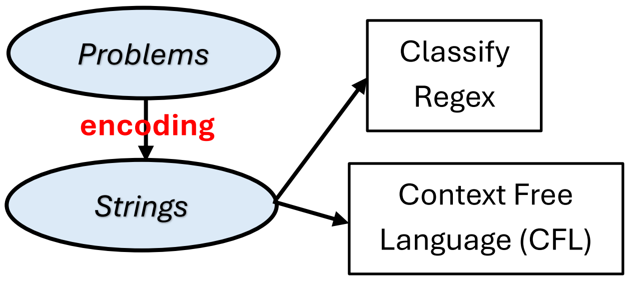
AUTOMATA

Key Points
- Language Recognition
- The power of encoding
- Everything is a string

Application
- Programming Language (PL)
- Natural Language Processing (NLP)
- Security/Cryptography (Regex)
- Games
- DNA (Genes)



Alphabet (Σ)
- Finite set of symbols

String (w)
- Finite series of symbols maybe in the alphabet.

Set
- Collection of objects with common characteristics.

Language
- Infinite/finite set of string from alphabet (Σ).

Theorem: All language is a subset of Σ*.

Sigma Kleene Star (Σ*)
- Set of all possible string from ε (Epsilon).

Σ^+
- Same with Σ* but has no ε.

Length (|w|)
- Number of characters in a string (w).

Function of Strings
1. |ababb| = 5
2. $\#_a$ (ababb) = 2
3. Concatenate
   w = abba, v = baba
   wv = abbababa
   vw = babaabba
   Theorem: w ε = w
4. Reverse
   v^R = abab
   Theorem: $(wv)^R = v^R w^R$
5. Replication
   w^2 = ww = abbaabba

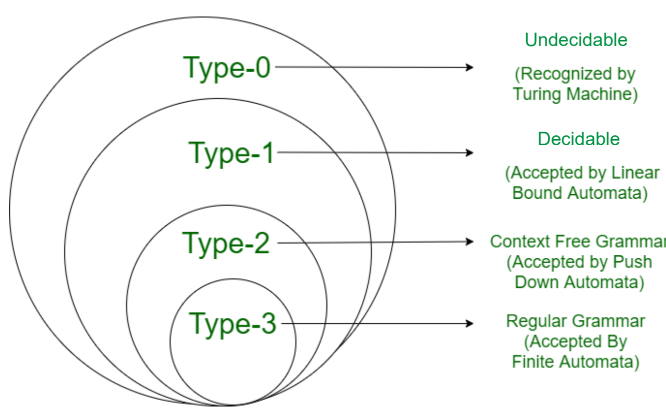
Relation on String
1. Substring
2. Prefix
3. Suffix

Define a Language
1. Listing
2. Set Builder
   Ø, {} = empty language
   {ε} ≠ Ø
   - $L_1 = L_2$ if and only if $L_1 \leq L_2$ and $L_1 \geq L_2$.

Operation on Language

U = Universal = sigma Kleene star
1. $L_1 \cap L_2$ – Intersection
2. $L_1 \cup L_2$ – Union
3. $L_1 \setminus L_2$ - Difference
4. L_1', L_2^c, $-L_1$ - Complement
5. $L_1 \oplus L_2$ – Symmetric Difference
6. L_1L_2 – Multiplication (concatenation)
7. L_1^R – Reversal
   L_1^R = {ab, bba, bbab, aaa}
   Theorem: $(L_1 L_2)^R = L_1^R L_2^R$
8. L_1^* - Kleene Star
9. $| L_1 |$ - Cardinality
   Theorem: | Σ* | = infinite countably
   # of language – countably infinite

Target
1. Define language
2. Is w in L?
   Decision Problem
   - Adding two integer
     $L\_1 = \{w | w < int > + < int > = < int > where\ int \in \mathbb{Z}\}$
     $9 + 3 = 12 \in L\_1$
     $7 + 6 = 10 \notin L\_1$
   - Sorting
     $L_1 = \{w_1, w_2 \mid w_1 \# w_2$, w_1 is the given non-sorted sequence of numbers in a form num1, num2, … where w_2 is the sorted $w_1\}$
     $2, 4, 3, \#, 2, 3, 4 \in L\_2$



Language Classes

Machine Hierarchy
1. Finite State Machine (Regular Language)
2. Push – Down Automata (Context Free Language)
3. Turing Machine

Why??
1. Computational Power
2. Efficient
3. Decidability
4. Clarity

Tractability Hierarchy

P – computes in polynomial time

NP – Nondeterministic, polynomial time

P-SPACE – computational within polynomial space (memory)

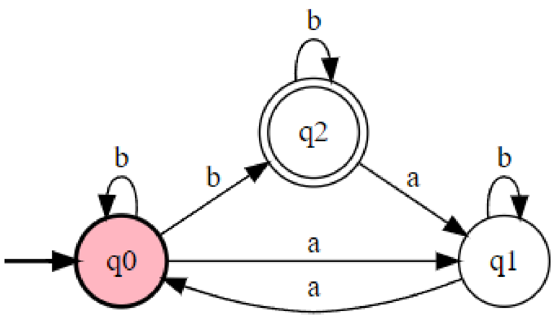
P ≤ NP ≤ P-SPACE

Computational
- Define problem as Language
- Define a program as static whose input is a string & output is Accept or Reject
- Accept: w ∈ L
- Reject: w ∉ L

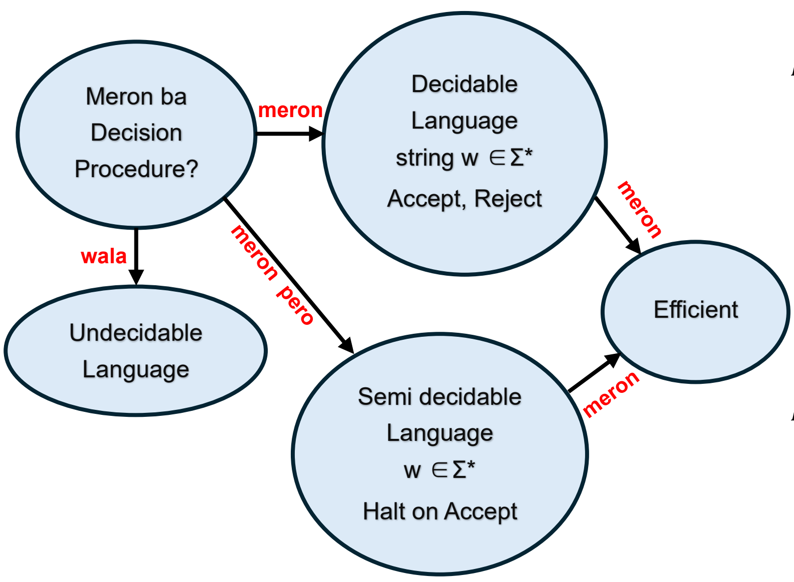
Key Ideas

1. Decision Procedure

   Ex.
   - is w in L?
   - is L = Ø
   - is $L_1 = L_2$
   - is L finite
2. Nondeterminism
   - given a string but don't know where it belongs.
   - given a string but don't know which of the two.



3. Function on Language



Finite State Machine (FSM)

- Deterministic Finite State Machine (DFSM)
  - A DFSM is a quintuple (K, Σ, δ, S ,A)

  K – is the set of state
  Σ – alphabet
  δ – transition matrix
  S – start state
  A – set of final Accept state

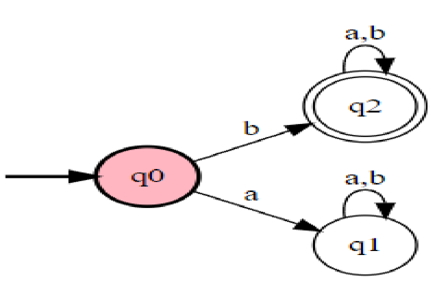
Configuration
- Start config (S, w)
- Yields in one step

Ex.

(1, abbaababb)
$\vdash_m$ (2, bbaababb) …
(1, abbaababb) $\vdash_m^*$ (3, bb)

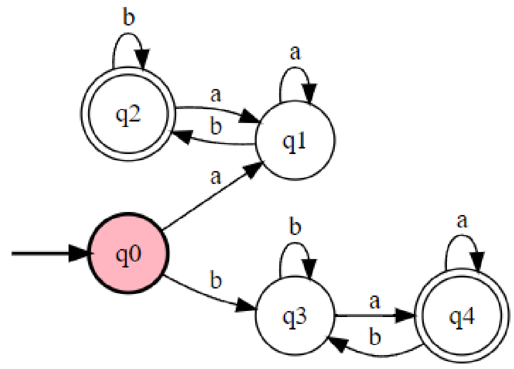
Computation:

$(S, w) \vdash_m^* (q_0, \varepsilon)$
$w \in L(m)$ if $q \in A$.
$w \notin L(m)$ if $q \notin A$.

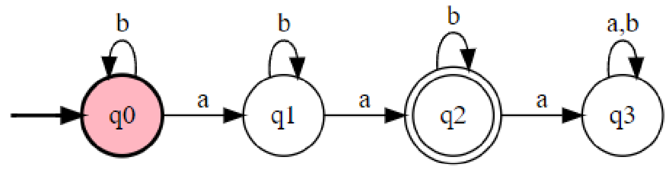


L → FSM
- Its okay even is has multiple states but prefer small no. of states.

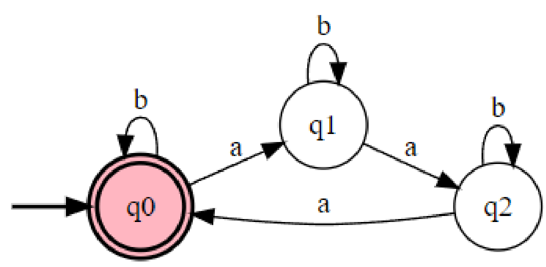
$L = \{w \ / \ w \ start \ \& \ ends \ in \ different \ letter\}$
$\Sigma = \{a, b\}$



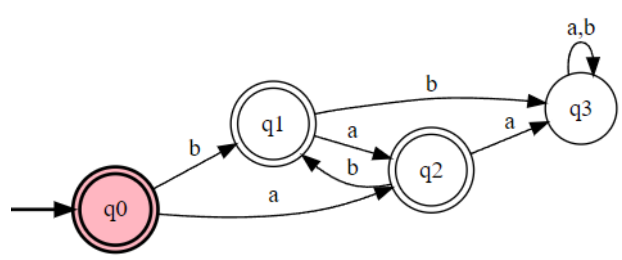
$L = \{w \ / \ \#_a(w) = 2\}$



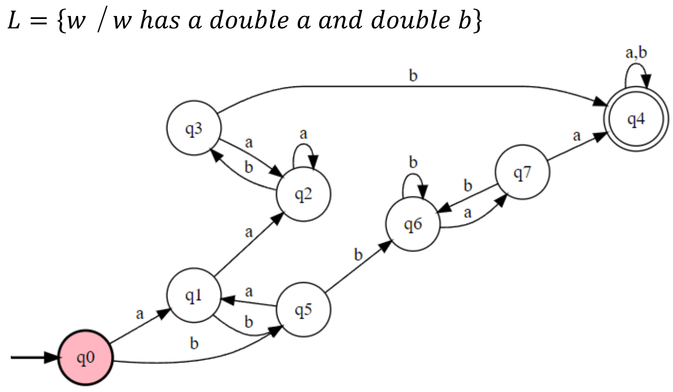
$L = \{w \ / \ \#_a(w) \ is \ divisible \ by \ 3\}$



$L = \{w \ / \ w \ is \ an \ alternating\}$



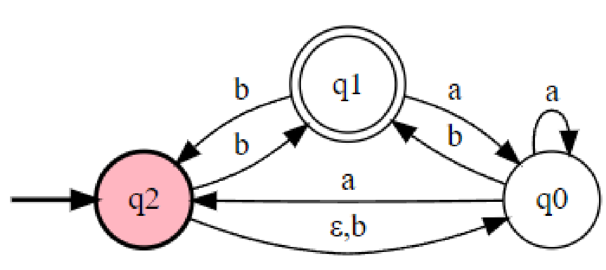
$L = \{w \ / \ w \ has \ a \ double \ a \ and \ double \ b\}$



Theorem: Any FSM has Regular Language

Non-Deterministic Finite State Machine (NDFSM)
- Is a quintuple (K, Σ, Δ, S ,A)

  Δ - transition function
- Has epsilon (ε) transition



Transition table:

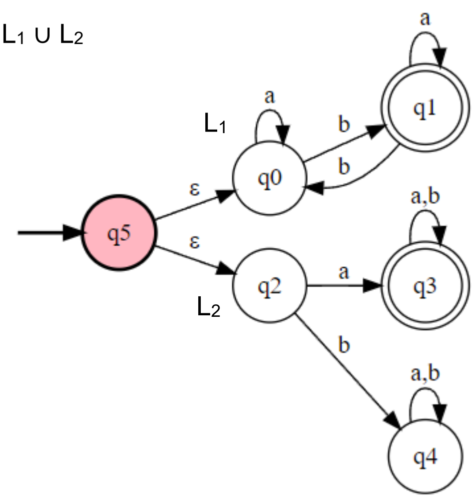
Δ	ε	a	b
q_0	**q_1**		**q_1, q_2**
q_1		**q_0, q_1**	**q_2**
q_2		**q_1**	**q_0**

Difference with DFSM:
1. ε-transition
2. state can have no outgoing
3. state can multiple outgoing

Definition: M accepts w, if one of its computation ends in accept state, reject, w if none of its computation ends in accept state
Theorem: DFSM = NDFSM

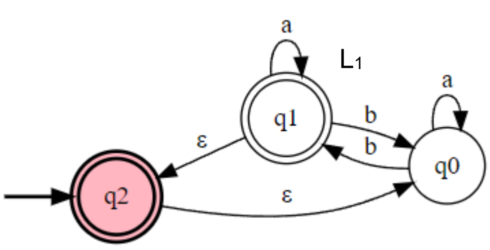
Operation on Language
NDFSM

$L_1 \cup L_2$

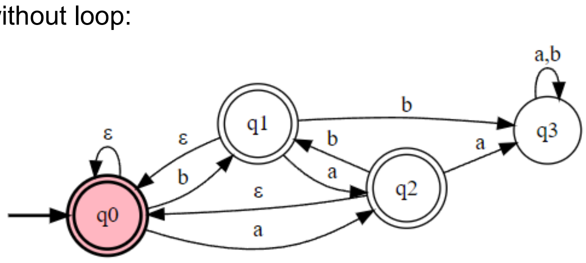


L*
- the new initial state would also be the final state.
- Kleene star always has an ε (epsilon).
- It is not necessary to create a new initial state if the existing initial state doesn't have a loop.

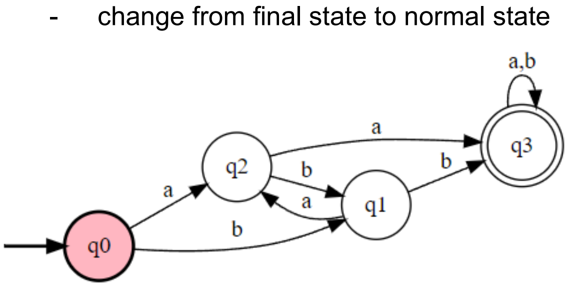
with loop:



without loop:

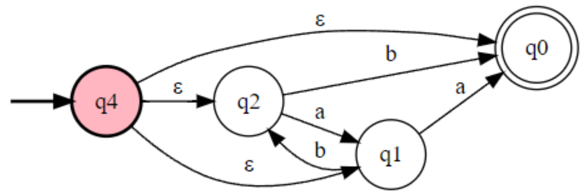


L'
- change from normal state to final state
- change from final state to normal state



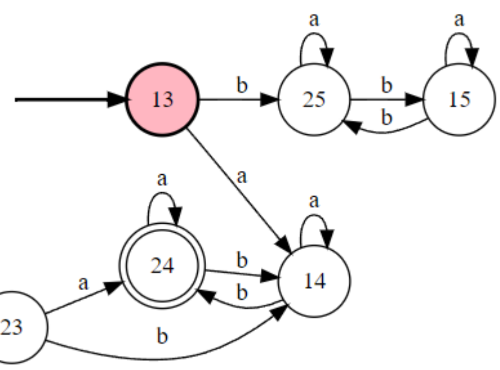
L_1^R
- change from final state to start state
- change from start state to final state
- reverse all of the arrows

Alternating:



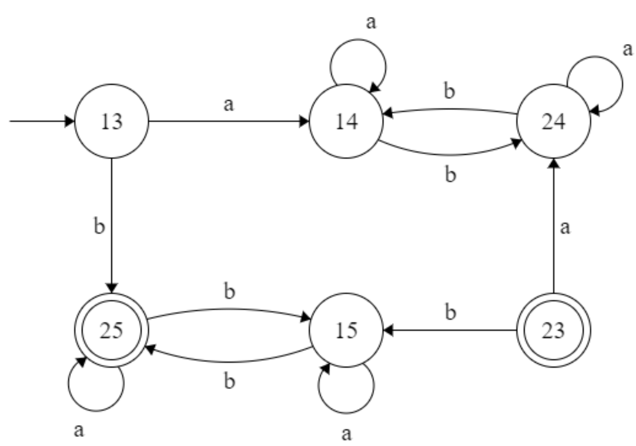
$L_1 \cap L_2$
- states of L_1 x states of L_2

δ	a	b
→**13** - initial	14	25
14	14	24
15	15	25
23	24	14
24 - final	24	14
25	25	15



$L_1 \setminus L_2 = L_1 \cap L_2'$

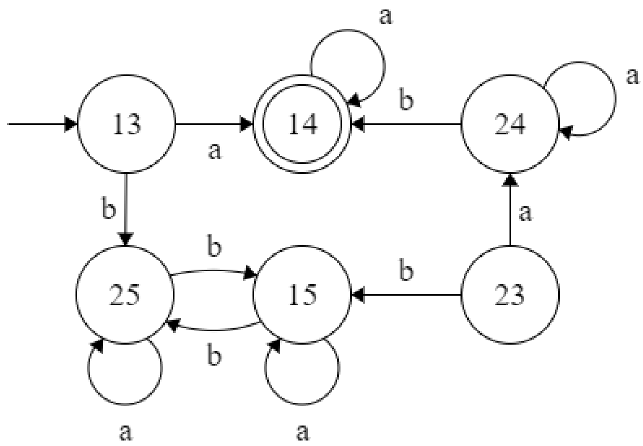
δ	a	b
→**13** - initial	14	25
14	14	24
15	15	25
23- final	24	15
24	24	14
25- final	25	15



$L_1 L_2 = (L_1 \setminus L_2) \cup (L_2 \setminus L_1)$

$(L_2 \setminus L_1)$

δ	a	b
→**13** - initial	14	25
14 - final	14	24
15	15	25
23	24	15
24	24	14
25	25	15



Simulators of FSM

- minimum FSM (≈L)

Theorem: There is a Unique Min DFSM for a regular language

Theorem: The ≈L is the lower bound of the # of state.

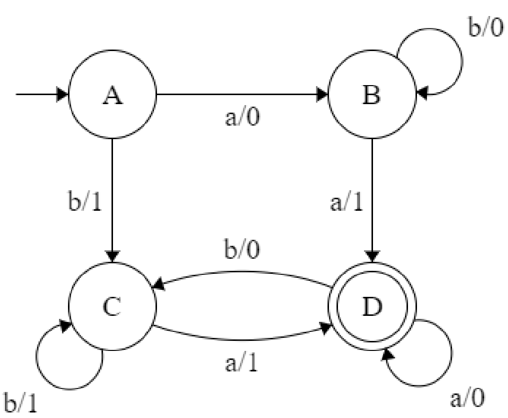
Theorem: Myhill Nerode Theorem – A language is Regular if and only if the number of equivalent FSM to ≈L is finite.

FSM: Transducer ($L_1 \Rightarrow L_2$)

- can have NO Accept State
- Output can be repeated

1. **Moore Machine**

- Is a seven tuple (K, Σ, O, δ, D, S ,A)
    - O – Output
    - D – Output Transition

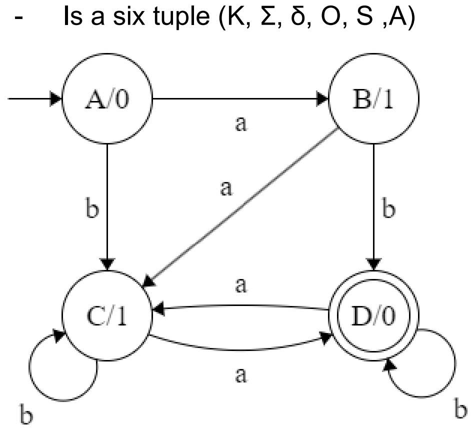


Output: string: babbaab

$\vdash_m$ (A, babbaab, **ε**)
$\vdash_m$ (C, abbaab, **1**)
$\vdash_m$ (D, bbaab, **11**)
$\vdash_m$ (C, baab, **110**)
$\vdash_m$ (C, aab, **1101**)
$\vdash_m$ (D, ab, **11011**)
$\vdash_m$ (D, b, **110110**)
$\vdash_m$ (C, **ε**, **1101100**)

2. **Mealy Machine**

- Is a six tuple (K, Σ, δ, O, S ,A)



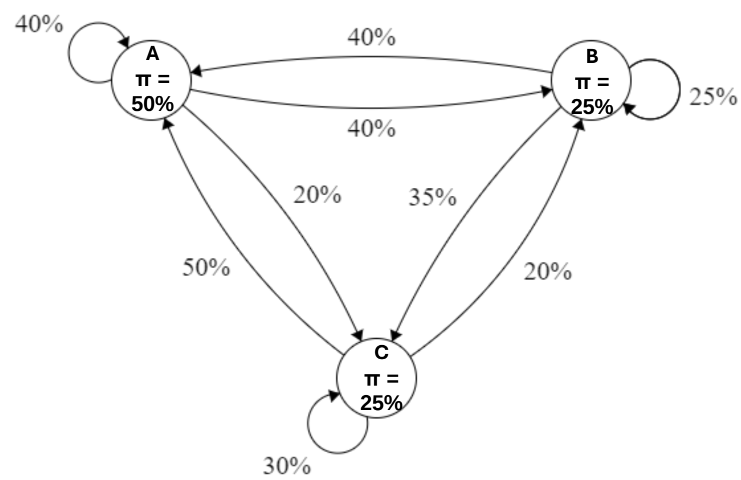
Output: string: abaabba

$\vdash_m$ (A, abaabba, **0**)
$\vdash_m$ (B, baabba, **01**)
$\vdash_m$ (D, aabba, **010**)
$\vdash_m$ (C, abba, **0101**)
$\vdash_m$ (D, bba, **01010**)
$\vdash_m$ (D, ba, **010100**)
$\vdash_m$ (D, a, **0101000**)
$\vdash_m$ (C, **ε**, **01010001**)

Stochastic FSM

1. **Markov Model (AI)**

- Is a tritupple (K, π, A)
    - π – vector probability matrix
    - A – transition probability matrix
- There's NO Start and Final State
- No. of state = n
- Start state depends on the value of random probability of π.
- Sum of probability is equal to 100.
- P (A|A) – supposed to create a table



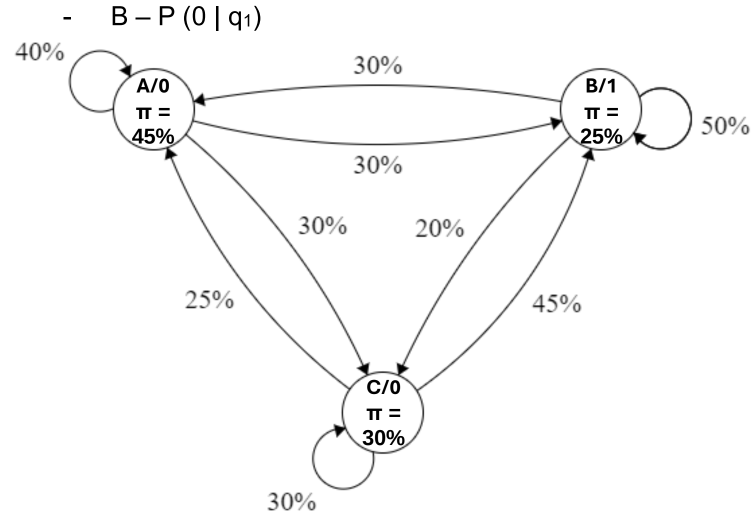
Output can be:

2 steps (A – (A)ttack, B – (S)tandby, C – (W)alk)

Steps	Probability
AA	(.5)(.4)
AS	(.5)(.4)
AW	(.5)(.2)
SA	(.25)(.4)
SS	(.25)(.25)
SW	(.25)(.35)
WA	(.25)(.5)
WS	(.25) (.2)
WW	(.25) (.3)

2. **Hidden Markov Model (HMM)**

- Is a five tuple (K, O, π, A, B)
- B – P (0 **|** q_1)



P (001 | B)

- AAB = (.45)(.4)(.3)
- ACB = (.45)(.3)(.45)
- CCB = (.3)(.3)(.45)
- CAB = (.3)(.25)(.3)

Problems:

- **Decoding problem** – it is hard to find a path, common algo used (Viterbi Algorithm)
- **Evaluation Problem** – How can we determine if the path is right, common algo used (Forward Algorithm)
- **Training Problem**

3. **Buchi Automata DA ≠ NDA**

- Input: “infinite string”
- It is a five tuple (K, Σ, △, S, A)
- Accepts when it enters in the accepts in infinite # of times.
- A language accept by Buchi Automata is called Buchi Language.

Theorem: L_1 is Regular, L_2 is Buchi Accepted, L_1L_2 is Buchi Acceptable.

Decision Procedure

1. Emptiness – if there’s no loop
2. Non-empty
3. Finite
4. Equivalent

A = B

Iff A ⊆ B & B ⊆ A

$L_1 = L_2$

Iff $L_1 \cap L_2' = \varnothing$,

$L_2 \cap L_1' = \varnothing$

Zugzwang – any moves that will result to a lose.

Regular Expression

- Simple pattern language
- Formal def: A regEx is a string that can be formed according to ff. rules:
  1. Ø is a RegEx
  2. ε is a RegEx
  3. ∀ x ∈ Σ, x is a RegEx
  4. Given two regex α & β then αβ is regex.
  5. Given two regex α & β then α ∪ β is regex.
  6. Given a regex α, α* is a regex.
  7. Given a regex α, α+ is a regex.
  8. Given a regex α, (α) is a regex.

a* = aa+

Order of Operation = (), * $^+$, αβ, +

L = {w|w starts in a}
= a (a + b)*

L = {w|w ends in b}
= (a+b)*b

L = {w|w has a substring abb}
= (a+b)*abb(a+b)*

L = {w|w starts & ends in same letter}
= a(a+b)*a + b(a+b)*b

L = {w| $\#_a$(w) is exactly 2}
= b* a b* a b*

L = {w| $\#_a$ (w) is even}
= (b*ab*a)*b*

L = {w|w is alternate}
= (ab)*(a+ε) + (ba)*(b+ε) or (a+ε)(ba)*(b+ε)

L = {w|w has a double a’s & double b’s}
= (a+b)*aa(a+b)*bb(a+b)* + (a+b)*bb(a+b)*aa(a+b)*

L = {w| $\#_a$ (w) ≥ 2}
= (a+b)* a (a+b)* a (a+b)*

L = {w| |w| is even}
= (ab+ba+bb+aa)*

L = {w|$\#_a$ (w) > 0 & $\#_b$ (w) >0}
= a^+

L = {w|w starts with some a’s & follow by some b’s}
= a*b*

Theorem: If you have FSM then you also have RegEx and vice versa.

Theorem: If generation = regex, if determining if an element = FSM

Applications:

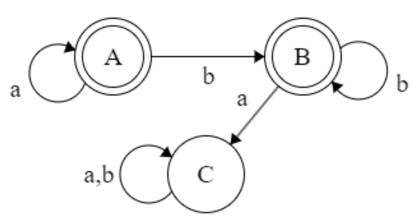
- Password
- Email checker

Manipulating or simplifying RegEx

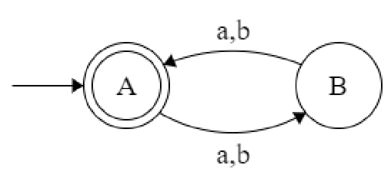
- ε is commutative
- associative in union
- Ø an identity for Union (ab + Ø) = ab
- Union is idempotent a + a = a
- Given any two set A and B, if B ⊆ A, then A ∪ B
- Concatenation is associative
- Concatenation is not commutative
- aε is still a
- Ø is zero a Ø = Ø
- Distribute concatenation over union
  - a(b+c) = ab + ac
- Ø* = ε
- ε* = ε
- (α*)* = α*
- a*a* = a*
- α* β* = α*
- (α + β)* = (α* β*)*

RegEx → FSM

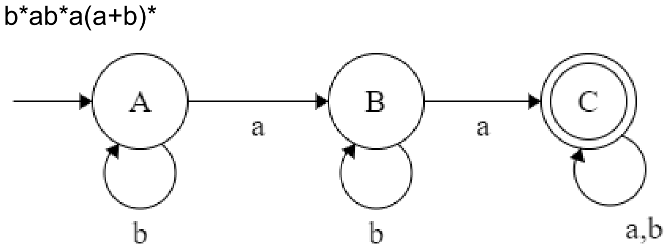
a*b*



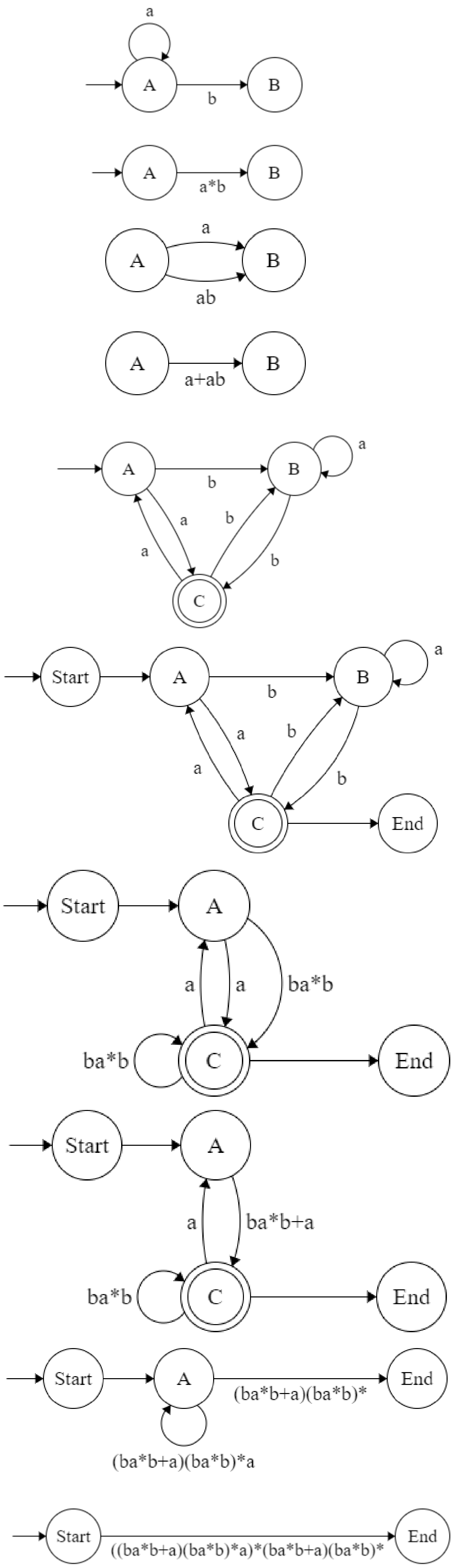
(ab+ba+bb+aa)*



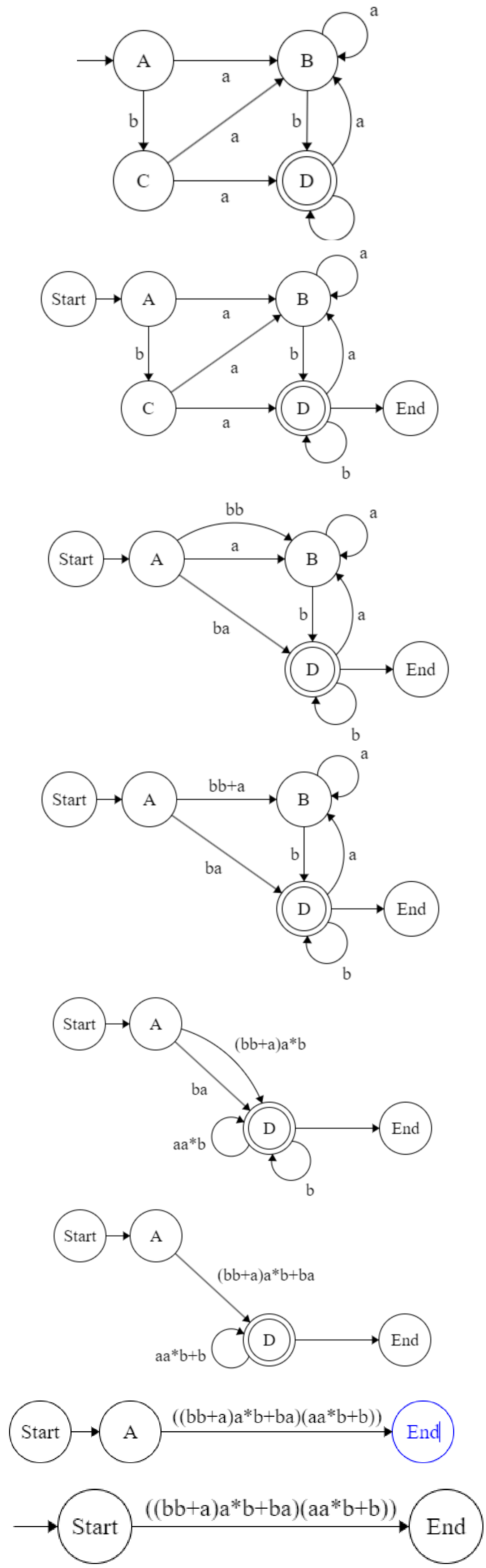
b*ab*a(a+b)*



FSM → RegEx



a
A
b
B
a*b
ab
a+ab
Start
C
End
ba*b
ba*b+a
(ba*b+a)(ba*b)*
(ba*b+a)(ba*b)*a
((ba*b+a)(ba*b)*a)*(ba*b+a)(ba*b)*



A
B
C
D
Start
End
bb
ba
bb+a
(bb+a)a*b
aa*b
(bb+a)a*b+ba
aa*b+b
((bb+a)a*b+ba)(aa*b+b))

Regular Grammar

- Rule based
- Is a four tuple (V, Σ, R, S)
- V – set of all the symbol
  - Nonterminal (capital letters Ex. A, B … S, Ø, …)
  - Terminal (Σ ∪ ε)
- Σ – terminal without ε = alphabet
- R – Set of all the rules X → Y
- S – is the nonterminal start

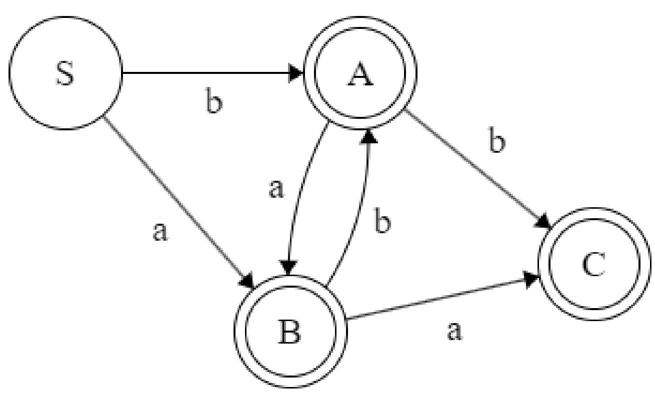
Rules of Rules of Grammar

1. Has a left hand side with a single nonterminal.
2. Has a right hand side with three possible:
   a. E
   b. Terminal from Σ (single)
   c. Single terminal followed aby single nonterminal

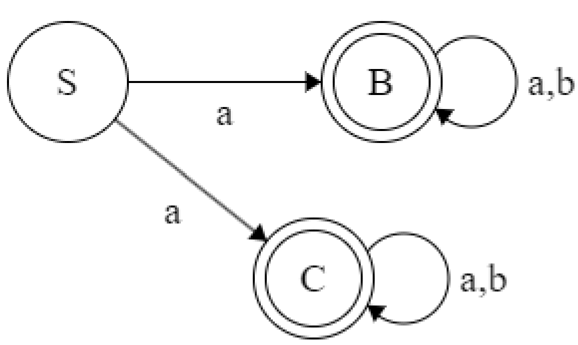
Note: If there is a violation within these rules will automatically not be a Regular Grammar.
Theorem: Any regular grammar has also a grammar.

RG → DFSM

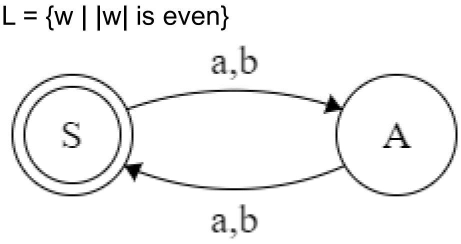
S → aB | bA
A → aB | ε
B → bA | ε



S → aB
A → aB | bB | ε



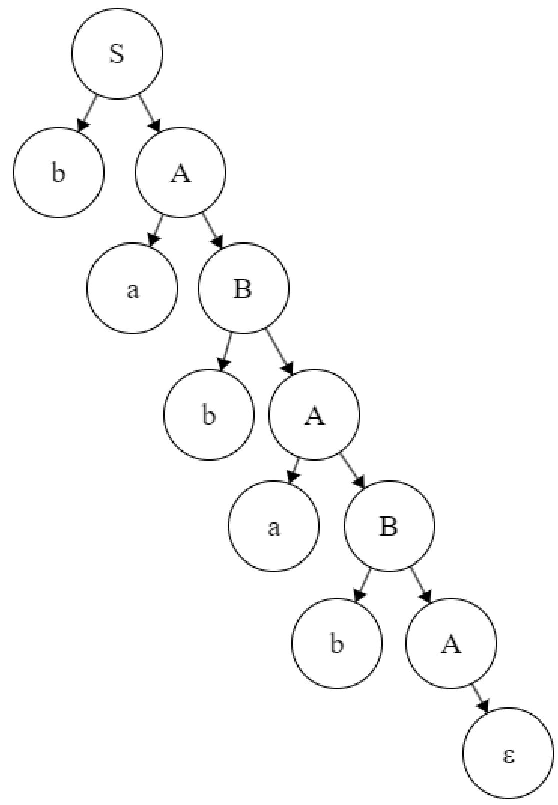
L = {w | |w| is even}



S → Aa | Ba | ε
A → As | Bs

Generating a string

- Forward computation / can also use backward computation
- For Alternating:
- S → bA → baB → babA → babaB → bababA → babab ε
- In tree



CHAPTER 8: Regular Grammar and non-Regular Grammar
Theorem:

1. Regular language is finite countably
2. Every finite language is regular
3. If a language has a DFSM, RG, RegEx then it has a regular language
4. Closure
   a. Close under Union
   b. Concatenation
   c. Kleene Star
5. Complement
   a. Reverse
   b. Intersection
   c. Letter Substitution
6. If the |w| ≥ |K|, then somewhere in the computation has a loop.

Pumping Lemma for Regular Language

- If L is a RL then ∃|K| ≥ |
- ∀ w ∈ L
- |w| ≥ |K|
- ∃ x, y, z
- w = xyz
- |xz| ≤ |K|
- y ≠ ε
- then ∃ q, xy^qz ∈ L

Ex. W = ababab
x = ab, y = ab, z = ab
$xy^qz = ab(ab)^qab \in L$

L = {a^nb^n, n ∈ $\mathbb{Z}^+$}
Assume:
|K| = 200
w = $a^{100}b^{100}$
x = a^{99}
y = a
z = b^{100}
$a^{99}(a)^qb^{100} \notin L$

Non - Regular

- balance parenthesis
- palindrome
- measures the number of the elements (letters)

Chapter 9: Decidability

- if it has algorithm then it is decidability, otherwise not.

Decision Procedure:

1. Membership “is w in L?”
    a. Regex, RG, FSM answers the membership
2. Emptiness | Totality
    a. Emptiness – L (M) = Ø
        i. There is no path to the final state.
    b. Totality L (M) = Σ^*
        i. All of the state is a final state.
3. Finiteness | Infiniteness
    a. Finiteness – there is no loop from start to finish
        i. The regex has no Kleene star
    b. There is a loop from start to finish
        i. It has Kleene star
4. Equivalent
    a. $A \cap B' = \varnothing$
    b. $B \cap A' = \varnothing$
5. Minimality (min DFSM)
    a. Number of states is the minimum.