Language Class:

- Regular Language
- Context Free Language
- Recursive and Recursively Enumerable Language

Language Classes have 2 important properties:

A. Decision Properties
B. Closure Properties

Properties of Regular Language:

A. Decision Properties:

1. Emptiness Properties
2. Finiteness Properties
3. Equivalence Properties
4. Membership Properties

B. Closure Properties:

1. Union
2. Concatenation
3. Kleene Closure
4. Reversal
5. Complement
6. Intersection

Problems are classified into 2 groups:

1. **Decidable** – the problems under this class has an algorithm to solve.
2. **Undecidable** – the problems under this class has no known algorithm to solve.

A. Decision Properties:

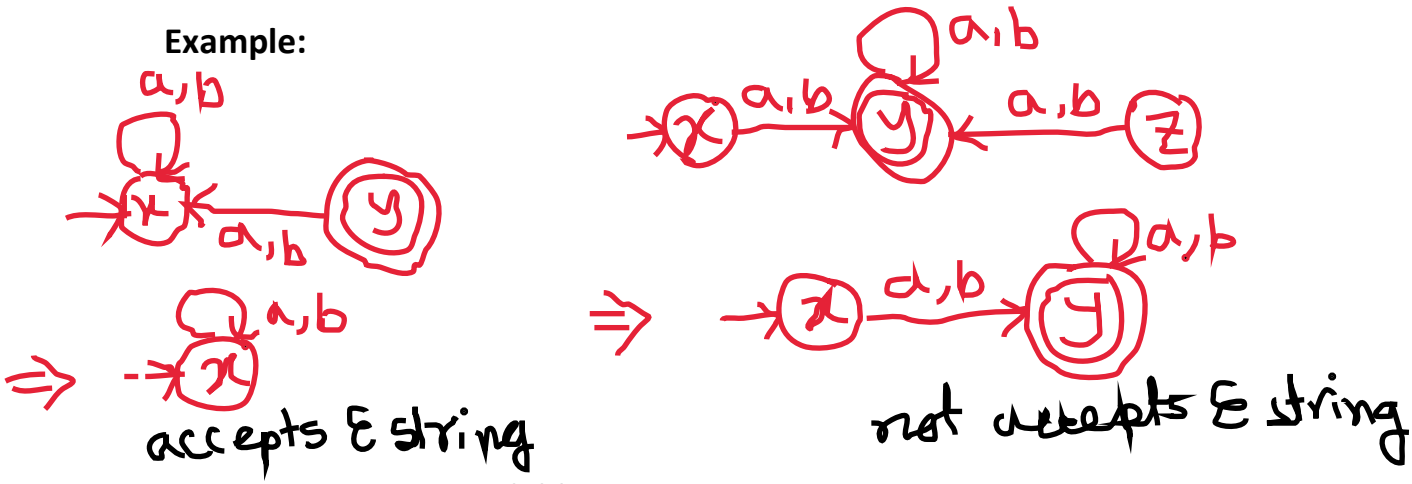
1. Emptiness Properties: - Decidable

"Finite Automata (FA) accepts empty language or not?"

Algorithm:

1) Eliminate inaccessible state(s)
2) If there exists at least 1 accepting sate in the resultant FA, then it accepts a non-empty language otherwise empty

Example:



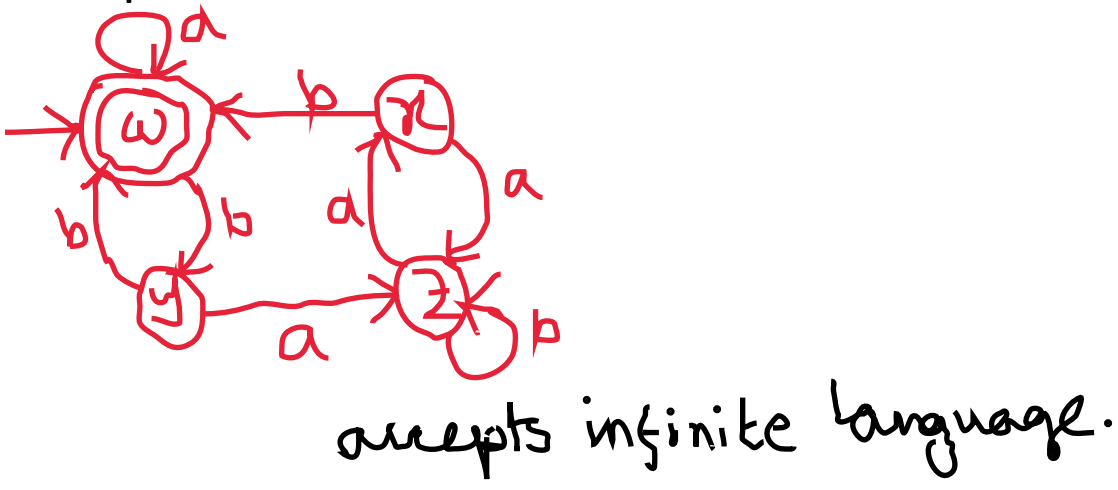
2. Finiteness Properties: - Decidable

"Checking whether the language is accepted by the given FA is finite or not"

Algorithm:

1) Eliminate all inaccessible state(s)
2) Eliminate the state(s) from which final state(s) is/are not reachable
3) In the resultant FA, if there exists any loop (cycle), the FA accepts the infinite language otherwise finite.

Example:



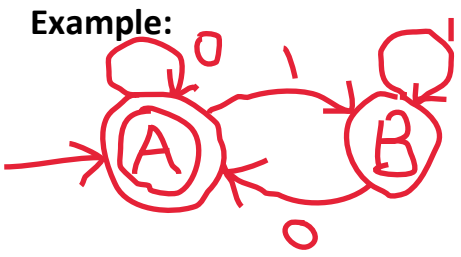
3. Equivalence Properties: - Decidable

"Given 2 FA accept same language?"

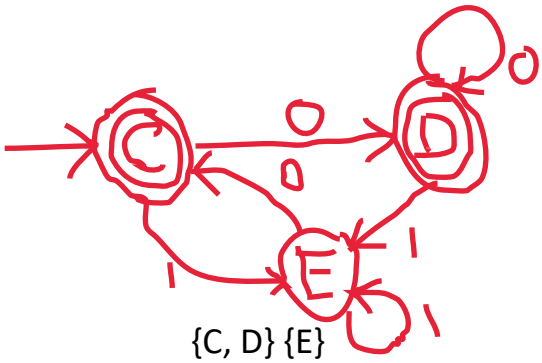
Algorithm:

1) Contract the "Comparison Table"
2) If the "State(s)" follow the homogeneous transition in "Comparison Table" then the FAs are accepting same language otherwise not

Example:



{A} {B}



{C, D} {E}

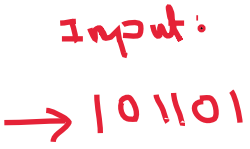
	0	1
{A, C}	{A, D}	{B, E}
{A, D}	{A, D}	{B, E}
{B, E}	{A, C}	{B, E}

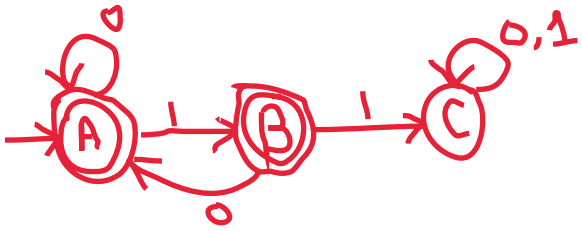
4. Membership Properties: - Decidable

"If a string is accepted by a given FA or not"

Algorithm:

1) Traverse through the FA of every symbols of the input
2) If the transition lies in accepting state after consuming all input symbols, then the string is accepted otherwise not.

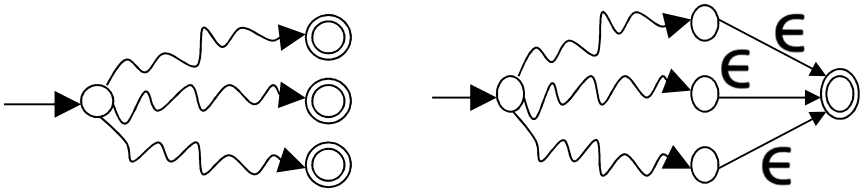


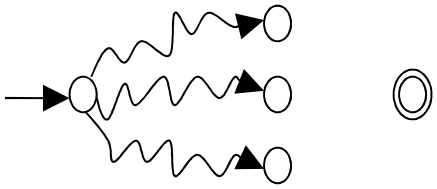


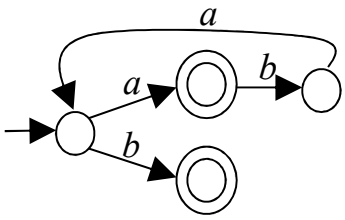
B. Closure Properties:

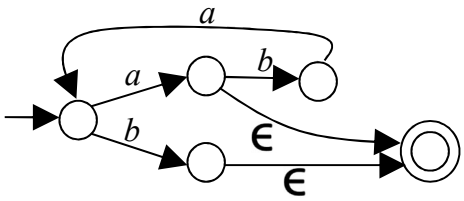
- Is the statement that implies certain operation to the language class produces another language in the same language class.

[**]









Take two languages

Example

Regular language L_1

Regular language L_2

$L(M_1) = L_1$

$L(M_2) = L_2$

NFA M_1

NFA M_2

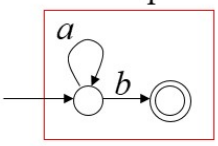
Single accepting state

Single accepting state

M_1

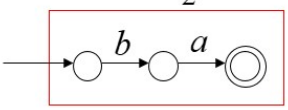
$n \geq 0$

$L_1 = \{a^n b\}$



M_2

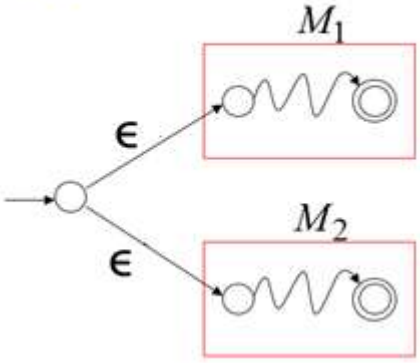
$L_2 = \{ba\}$

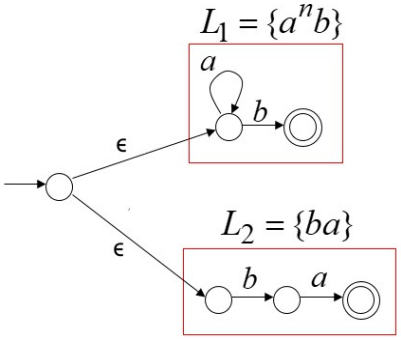


1. Union

NFA for $L_1 \cup L_2$

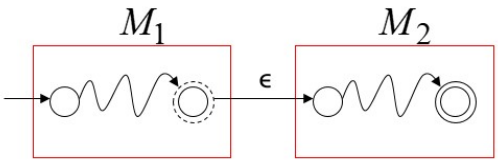
NFA for $L_1 \cup L_2 = \{a^n b\} \cup \{ba\}$



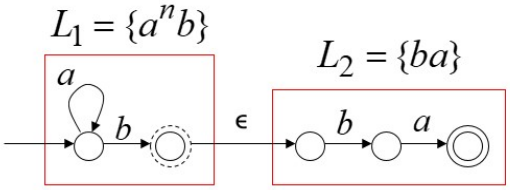


2. Concatenation

NFA for L_1L_2

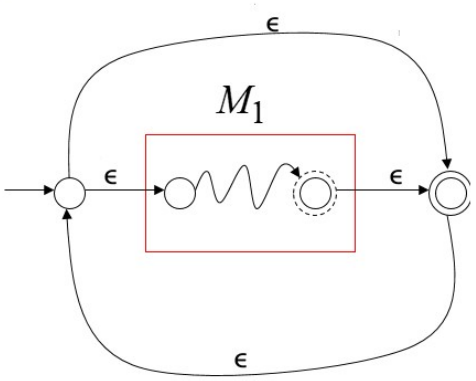


NFA for $L_1L_2 = \{a^n b\}\{ba\} = \{a^n bba\}$

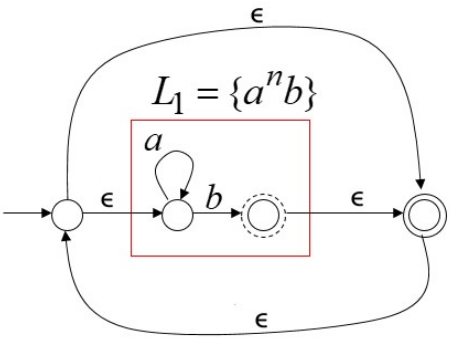


3. Kleene Closure

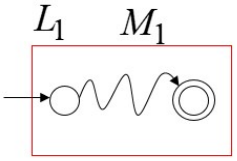
NFA for L_1*



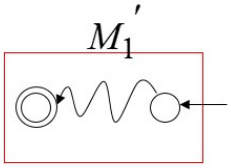
NFA for $L_1* = \{a^n b\}*$



4. Reversal

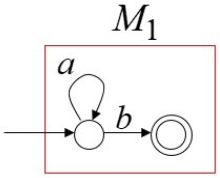


NFA for L_1^R

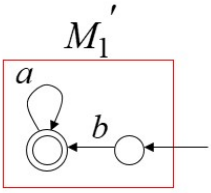


1. Reverse all transitions
2. Make initial state accepting state and vice versa

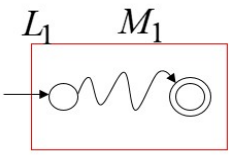
$L_1 = \{a^n b\}$

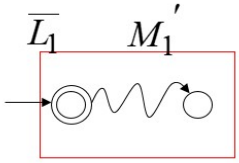


$L_1^R = \{ba^n\}$

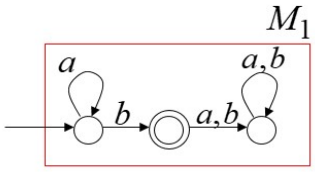


5. Complement



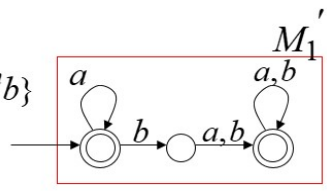


$L_1 = \{a^n b\}$



1. Take the DFA that accepts L_1
2. Make accepting states non-final, and vice-versa

$\overline{L_1} = \{a,b\}* - \{a^n b\}$



6. Intersection

L_1 regular

L_2 regular

We show $L_1 \cap L_2$ regular

$L_1 = \{a^n b\}$ regular

$L_2 = \{ab, ba\}$ regular

$L_1 \cap L_2 = \{ab\}$ regular

DeMorgan's Law: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

L_1 , L_2 regular

$\overline{L_1}$, $\overline{L_2}$ regular

$\overline{L_1} \cup \overline{L_2}$ regular

$\overline{\overline{L_1} \cup \overline{L_2}}$ regular

$L_1 \cap L_2$ regular

[Another way to prove Intersection Closure]

Machine M_1

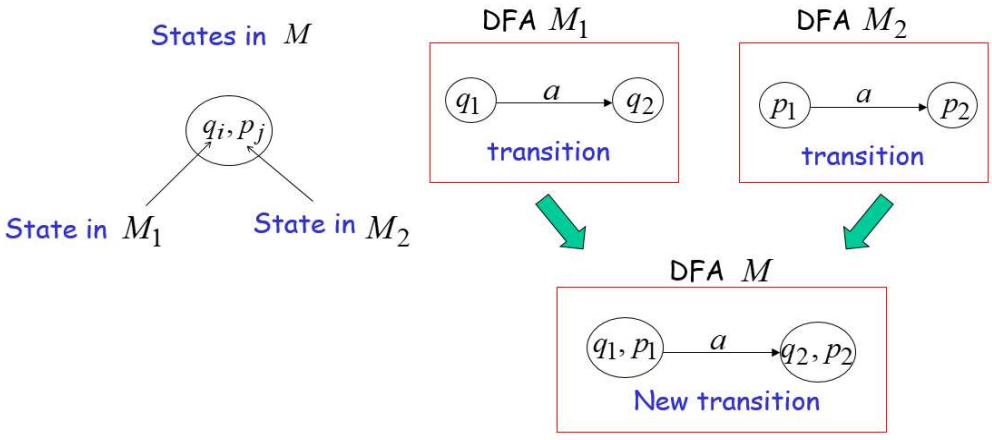
DFA for L_1

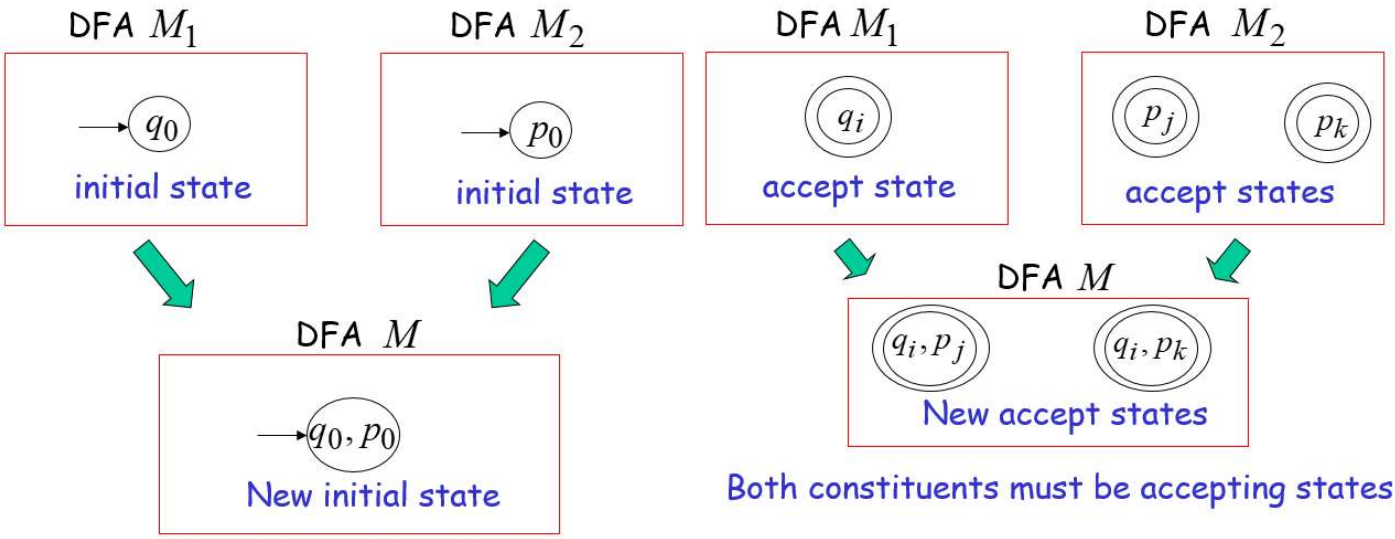
Machine M_2

DFA for L_2

Construct a new DFA M that accepts $L_1 \cap L_2$

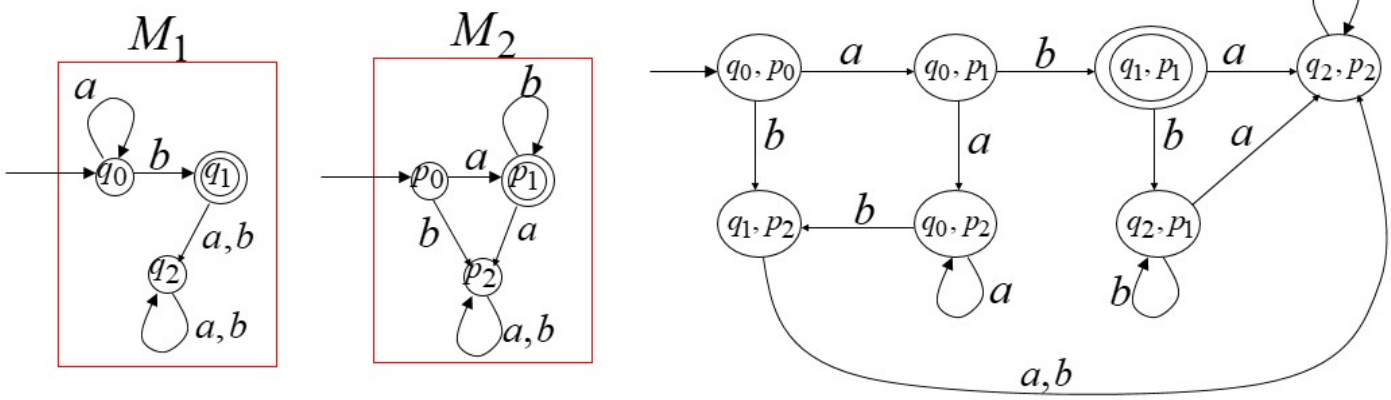
M simulates in parallel M_1 and M_2





$$L_1 = \{a^n b\}^{n \geq 0} \qquad L_2 = \{ab^m\}^{m \geq 0}$$

$$L = \{a^n b\} \cap \{ab^n\} = \{ab\}$$



M simulates in parallel M_1 and M_2

M accepts string w if and only if:

M_1 accepts string w

and M_2 accepts string w

$$L(M) = L(M_1) \cap L(M_2)$$