Damascus University

Automata theory and formal languages

Closure Properties & Regular Languages

Dr. Mohammad Ahmad

Closure properties

Example

- The language L of strings that end in 101 is regular

$$(0+1)^*101$$

- How about the language $\overline{L}$ of strings that do not end in 101?

Example

- **Hint:** w does not end in 101 if and only if it ends in:

$$000, 001, 010, 011, 100, 110 \text{ or } 111$$

  or it has length 0, 1, or 2

- So $\overline{L}$ can be described by the regular expression

$$(0+1)^*(000+001+010+010+100+110+111)$$
$$+ \varepsilon + (0 + 1) + (0 + 1)(0 + 1)$$

Complement

- The complement $\overline{L}$ of a language L is the set of all strings that are not in L

- Examples ($\Sigma = \{0, 1\}$)
  - L_1 = all strings that end in 101
  - $\overline{L_1}$ = all strings that do not end in 101
    = all strings end in $000, \ldots, 111$ or have length 0, 1, or 2

  - $L_2 = 1^* = \{\varepsilon, 1, 11, 111, \ldots\}$
  - $\overline{L_2}$ = all strings that contain at least one 0
    $= (0 + 1)^*0(0 + 1)^*$

Example

- The language L of strings that contain 101 is regular

$$(0+1)^*101(0+1)^*$$

- How about the language $\overline{L}$ of strings that do not contain 101?

You can write a regular expression,
but it is a lot of work!

Closure under complement

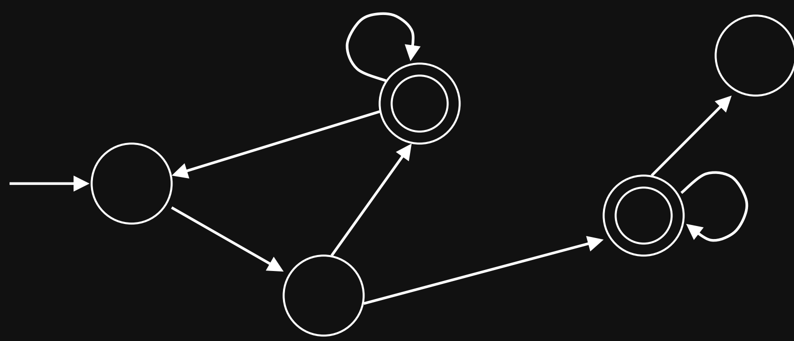
If L is a regular language, so is $\overline{L}$.

- To argue this, we can use any of the equivalent definitions for regular languages:

- The DFA definition will be most convenient
  - We assume L has a DFA, and show $\overline{L}$ also has a DFA

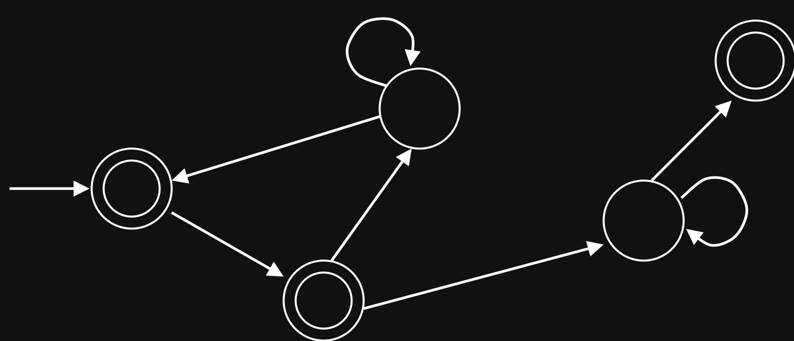
Arguing closure under complement

- Suppose L is regular, then it has a DFA M



accepts L

- Now consider the DFA M' with the accepting and rejecting states of M reversed

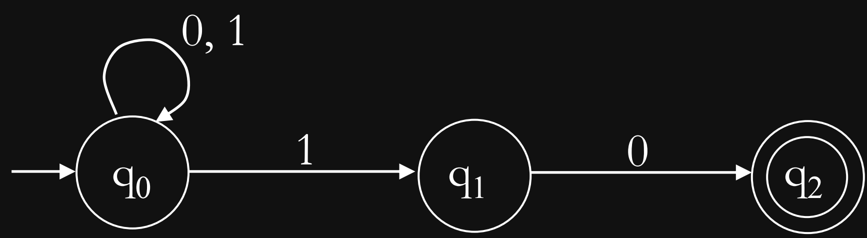


accepts strings not in L

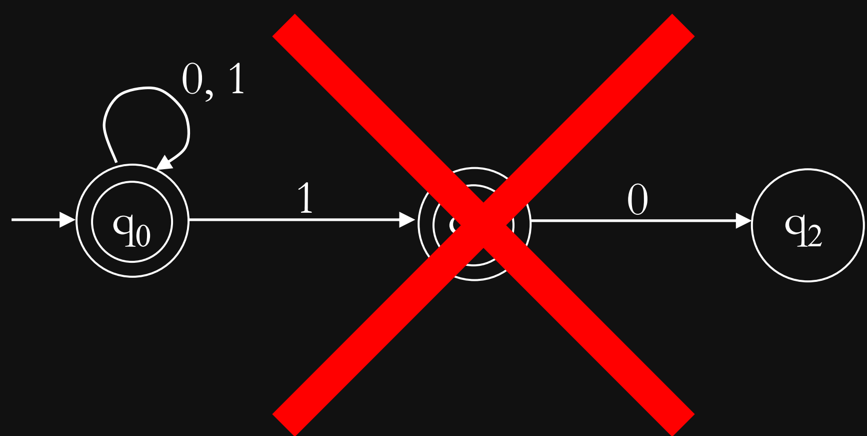
this is exactly $\overline{L}$

Food for thought

- Can we do the same thing with an NFA? **NO!**



$(0+1)^*10$



$(0+1)^*$

Intersection

- The intersection $L \cap L'$ is the set of strings that are in both L and L'

- Examples:

$L = (0 + 1)^*11$ $\quad L' = 1^*$ $\quad L \cap L' = 1^*11$

$L = (0 + 1)^*10$ $\quad L' = 1^*$ $\quad L \cap L' = \emptyset$

- If L, L' are regular, is $L \cap L'$ also regular?

Closure under intersection

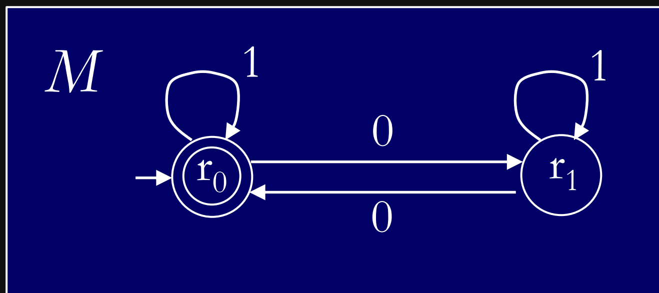
If L and L' are regular languages, so is $L \cap L'$.

- To argue this, we can use any of the equivalent definitions for regular languages:

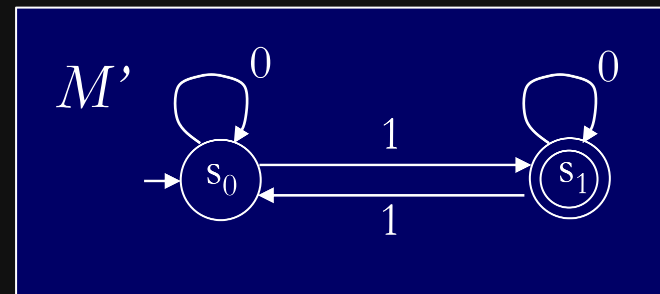
Suppose L and L' have DFAs, call them M and M'

Goal: Construct a DFA (or NFA) for $L \cap L'$

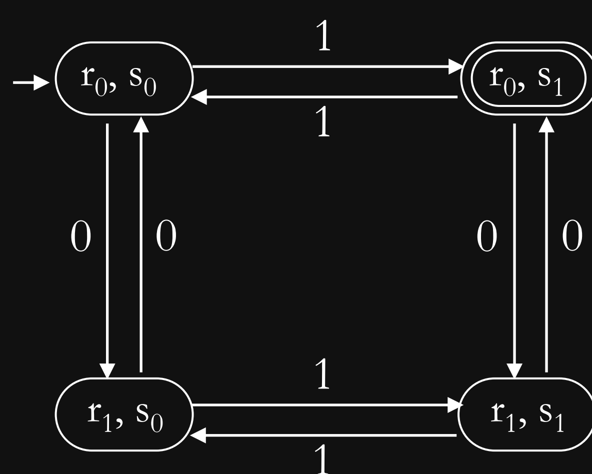
An example



L = even number of 0s



L' = odd number of 1s



$L \cap L'$ = even number of 0s and odd number of 1s

Closure under intersection

	M and M'	DFA for $L \cap L'$
states	$Q = \{r_1, \ldots, r_n\}$ $Q' = \{s_1, \ldots, s_{n'}\}$	$Q \times Q' = \{(r_1, s_1), (r_1, s_2), \ldots,$ $(r_2, s_1), \ldots, (r_n, s_{n'})\}$
start state	r_i for M s_j for M'	(r_i, s_j)
accepting states	F for M F' for M'	$F \times F' = \{(r_i, s_j): r_i \quad F, s_j \quad F'\}$

Whenever M is in state r_i and M' is in state s_j, the DFA for $L \cap L'$ will be in state (r_i, s_j)

Closure under intersection

	M and M'	DFA for $L \cap L$'
transitions	$r_i \xrightarrow{a} r_j$ in M $s_{i'} \xrightarrow{a} s_{j'}$ in M'	$r_i, s_{i'} \xrightarrow{a} r_j, s_{j'}$

Reversal

- The reversal w^R of a string w is w written backwards

$$w = \text{cave} \qquad\qquad w^R = \text{evac}$$

- The reversal L^R of a language L is the language obtained by reversing all its strings

$$L = \{\text{cat}, \text{dog}\} \qquad\qquad L^R = \{\text{tac}, \text{god}\}$$

Reversal of regular languages

- $L =$ all strings that end in 101 is regular

$$(0+1)^*101$$

- How about L^R?
- This is the language of all strings beginning in 101
- Yes, because it is represented by

$$101(0+1)^*$$

Closure under reversal

If L is a regular language, so is L^R.

- How do we argue?

Arguing closure under reversal

- Take a regular expression E for L
- We will show how to reverse E
- A regular expression can be of the following types:
  - The special symbols $\emptyset$ and ε
  - Alphabet symbols like a, b
  - The union, concatenation, or star of simpler expressions

Proof of closure under reversal

regular expression E	➡	reversal E^R
$\emptyset$		$\emptyset$
ε		ε
a (alphabet symbol)		a
$E_1 + E_2$		$E_1{}^R + E_2{}^R$
E_1E_2		$E_2{}^R E_1{}^R$
$E_1{}^*$		$(E_1{}^R)^*$

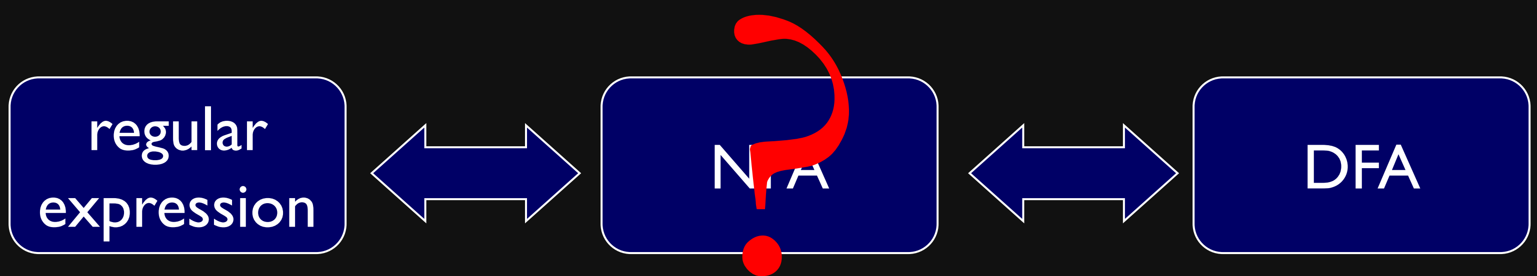
A question

$$L^{DUP} = \{ww\colon w \in L\}$$

Ex. $L = \{\text{cat}, \text{dog}\}$

$L^{DUP} = \{\text{catcat}, \text{dogdog}\}$

If L is regular, is L^{DUP} also regular?



A question

- Let’s try with regular expression:

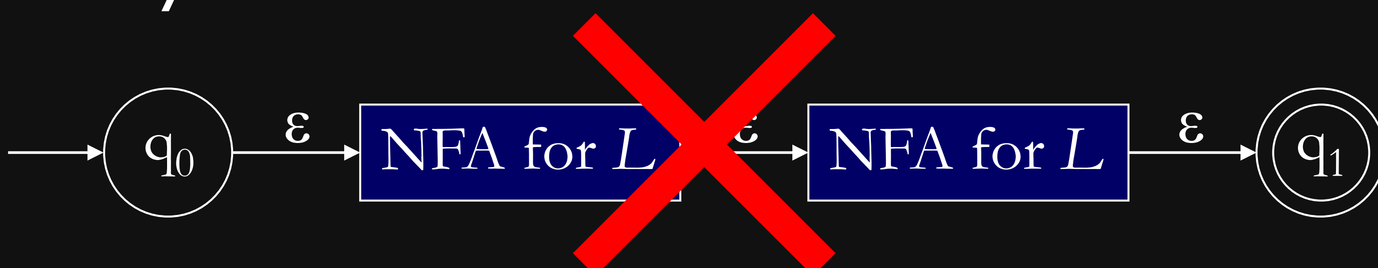
$$\cancel{L^{DUP} = LL}$$

$$L = \{\text{a, b}\}$$

$$L^{DUP} = \{\text{aa, bb}\}$$

$$LL = \{\text{aa, ab, ba, bb}\}$$

- Let’s try with NFA:



An example

$L = 0^*1$ is regular

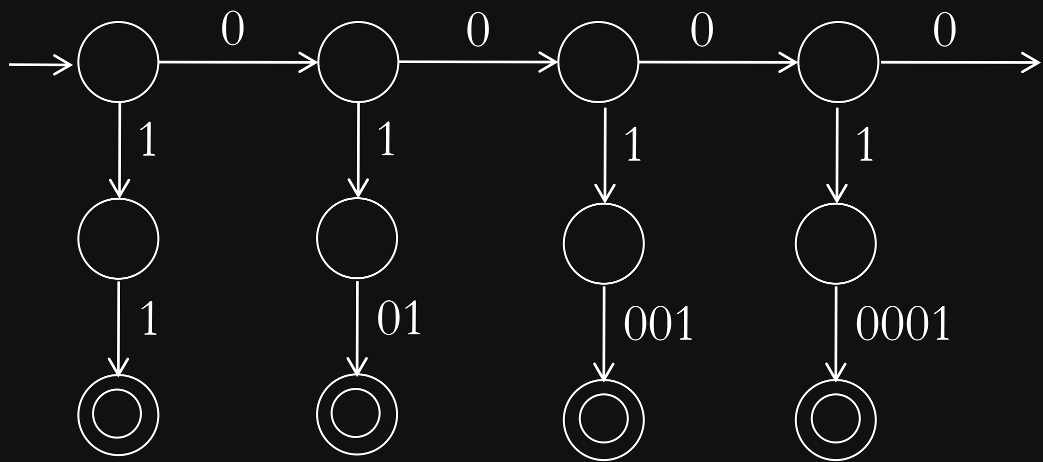
$$L \quad = \{1, 01, 001, 0001, ...\}$$

$$L^{DUP} = \{11, 0101, 001001, 00010001, ...\}$$

$$= \{0^n10^n1 : n \geq 0\}$$

- Let's try to design an NFA for L^{DUP}

An example



$$L^{DUP} = \{11, 0101, 001001, 00010001, \ldots\}$$

$$= \{0^n 1 0^n 1 \colon n \geq 0\}$$

For regular language L_1 and L_2

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1{}^*$

Reversal: $L_1{}^R$

Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

$\Bigg\}$ **Are regular Languages**

Regular Grammars

Regular Grammars

Grammars is generative description of a language

A regular grammar G is a quadruple (V, Σ, R, S), where:

- V is the rule alphabet (Variables), which contains non-terminals,
- Σ (the set of terminals).
- R (the set of rules) is a finite set of rules of the form:

  $$X \rightarrow Y,$$

- S (the start symbol) is a nonterminal.

Regular Grammars

In a regular grammar, all rules in R must:

- have a left hand side that is a single nonterminal
- have a right hand side that is:
  - ε, or
  - a single terminal, or
  - a terminals followed by a single nonterminal.

Legal: $S \rightarrow \mathtt{a}$, $S \rightarrow \varepsilon$, and $T \rightarrow \mathtt{a}S$

Not legal: $S \rightarrow \mathtt{a}S\mathtt{a}$ and $\mathtt{a}S\mathtt{a} \rightarrow T$

Regular Grammars

A regular grammar is any right-linear or left-linear grammar

Examples:

$$G_1$$

$$S \to abS$$

$$S \to a$$

$$G_2$$

$$S \to Aab$$

$$A \to Aab \mid B$$

$$B \to a$$

Observation

Regular grammars generate regular languages

Examples:

G_1

$$S \to abS$$

$$S \to a$$

$$L(G_1) = (ab)^* a$$

G_2

$$S \to Aab$$

$$A \to Aab \mid B$$

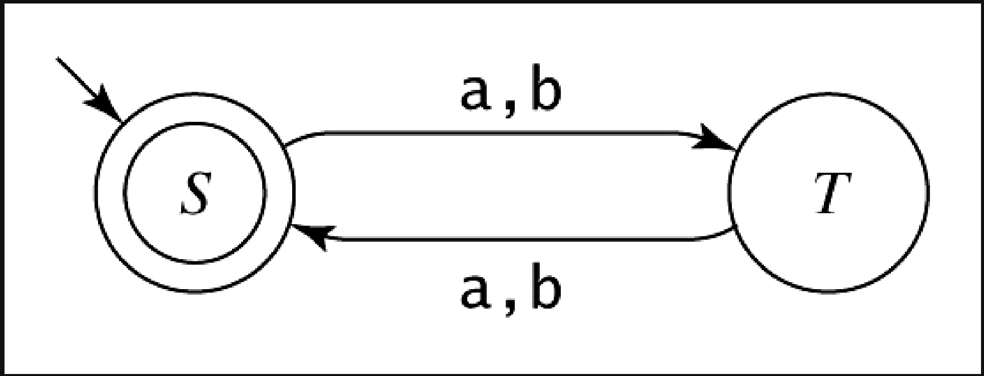
$$B \to a$$

$$L(G_2) = aab(ab)^*$$

Regular Grammar Example

$L = \{w \in \{\texttt{a}, \texttt{b}\}^* : |w| \text{ is even}\}$

FSM:

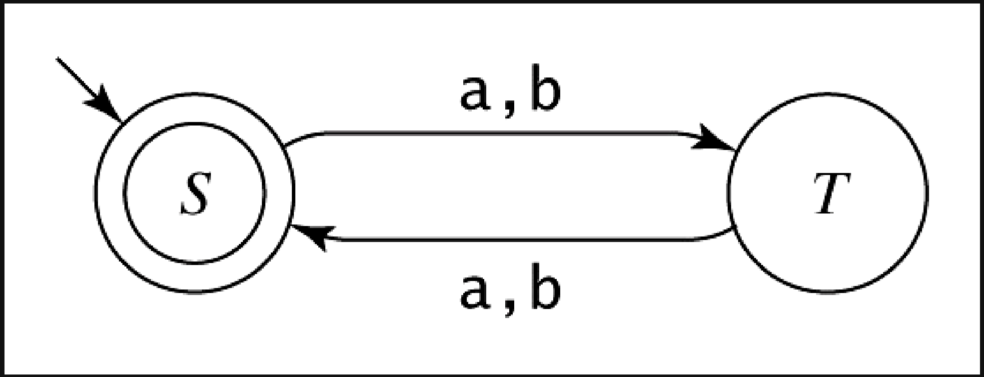


Regular Grammar Example

$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : |w| \text{ is even}\}$

Regular expression: $((\mathtt{aa}) \cup (\mathtt{ab}) \cup (\mathtt{ba}) \cup (\mathtt{bb}))^*$

FSM:



Regular grammar: G = (V, Σ, R, S), where

$V = \{S, T\}$, Σ = {a, b}, R = {

$S \rightarrow \varepsilon$

$S \rightarrow \mathtt{a}T$

$S \rightarrow \mathtt{b}T$

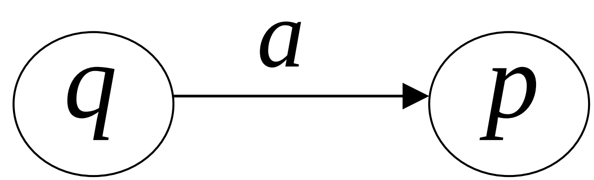
$T \rightarrow \mathtt{a}S$

$T \rightarrow \mathtt{b}S$

}

In General

For any transition:



Add production:

$$q \to ap$$

variable terminal variable

Regular Languages and Regular Grammars

Theorem: The class of languages that can be defined with regular grammars is exactly the regular languages.

Regular Languages and Regular Grammars

Regular grammar* $\rightarrow$ *FSM:

grammar to fsm (G = (V, Σ, R, S)) =

1. Create in M a separate state for each nonterminal in V.
2. Start state is the state corresponding to S .
3. If there are any rules in R of the form $X \rightarrow w$, for some $w \in \Sigma$, create a new state labeled #.
4. For each rule of the form $X \rightarrow w\ Y$, add a transition from X to Y labeled w.
5. For each rule of the form $X \rightarrow w$, add a transition from X to # labeled w.
6. For each rule of the form $X \rightarrow \varepsilon$, mark state X as accepting.
7. Mark state # as accepting.

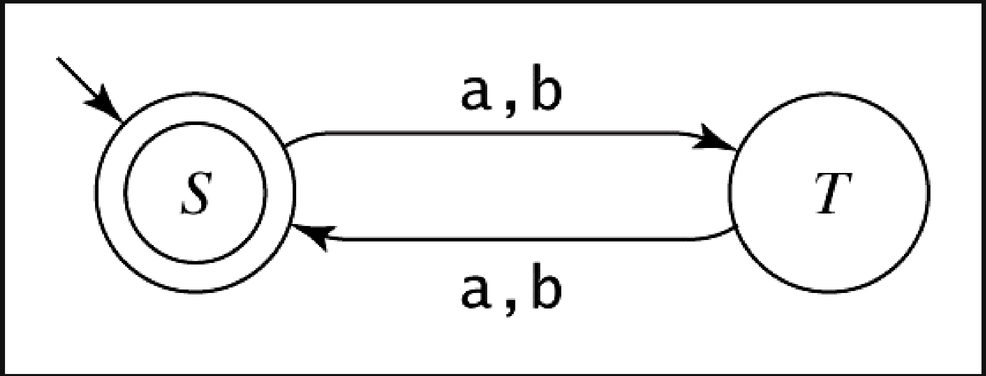
FSM* $\rightarrow$ *Regular grammar: Similarly.

Example 1 - Even Length Strings

$S \to \varepsilon$
$S \to \mathtt{a}T$
$S \to \mathtt{b}T$
$T \to \mathtt{a}S$
$T \to \mathtt{b}S$

Example 1 - Even Length Strings

$S \to \varepsilon$
$S \to \mathtt{a}T$
$S \to \mathtt{b}T$
$T \to \mathtt{a}S$
$T \to \mathtt{b}S$



Strings that End with aaaa

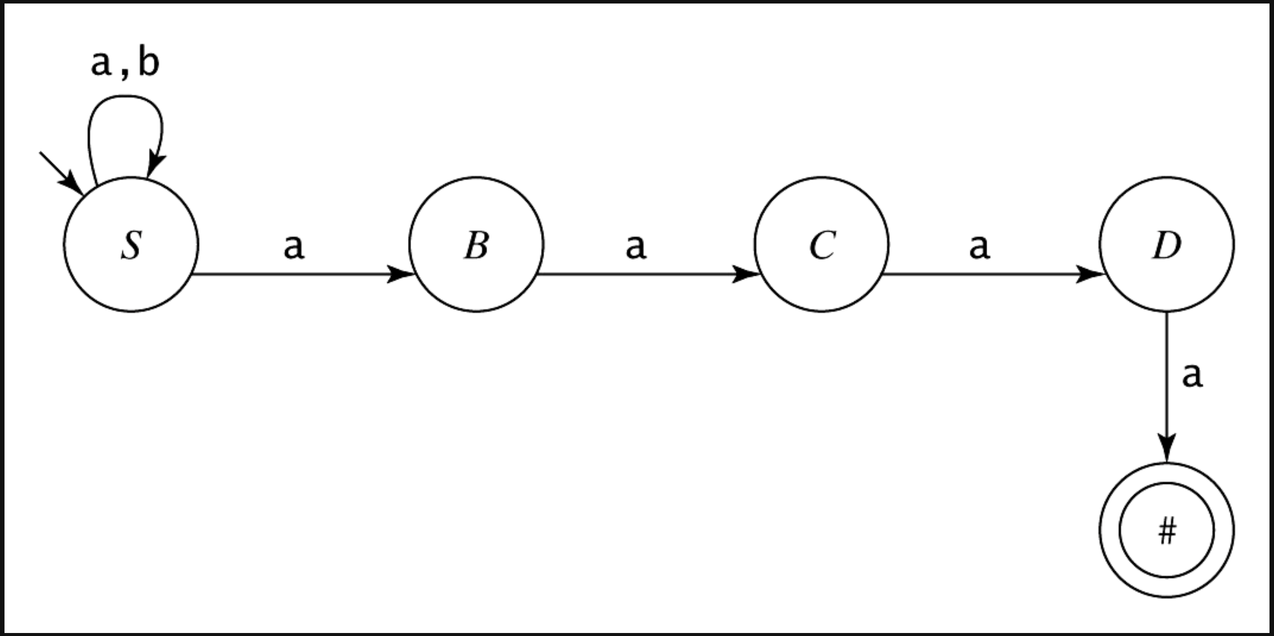
$L = \{w \in \{\texttt{a}, \texttt{b}\}^* : w$ ends with the pattern $\texttt{aaaa}\}$.

$$S \rightarrow \texttt{a}S$$
$$S \rightarrow \texttt{b}S$$
$$S \rightarrow \texttt{a}B$$
$$B \rightarrow \texttt{a}C$$
$$C \rightarrow \texttt{a}D$$
$$D \rightarrow \texttt{a}$$

Strings that End with aaaa

$L = \{w \in \{\texttt{a}, \texttt{b}\}^* : w$ ends with the pattern $\texttt{aaaa}\}$.

$S \rightarrow \texttt{a}S$
$S \rightarrow \texttt{b}S$
$S \rightarrow \texttt{a}B$
$B \rightarrow \texttt{a}C$
$C \rightarrow \texttt{a}D$
$D \rightarrow \texttt{a}$



Example 2 – One Character Missing

$S \rightarrow \varepsilon$
$S \rightarrow \mathtt{a}B$
$S \rightarrow \mathtt{a}C$
$S \rightarrow \mathtt{b}A$
$S \rightarrow \mathtt{b}C$
$S \rightarrow \mathtt{c}A$
$S \rightarrow \mathtt{c}B$

$A \rightarrow \mathtt{b}A$
$A \rightarrow \mathtt{c}A$
$A \rightarrow \varepsilon$
$B \rightarrow \mathtt{a}B$
$B \rightarrow \mathtt{c}B$
$B \rightarrow \varepsilon$

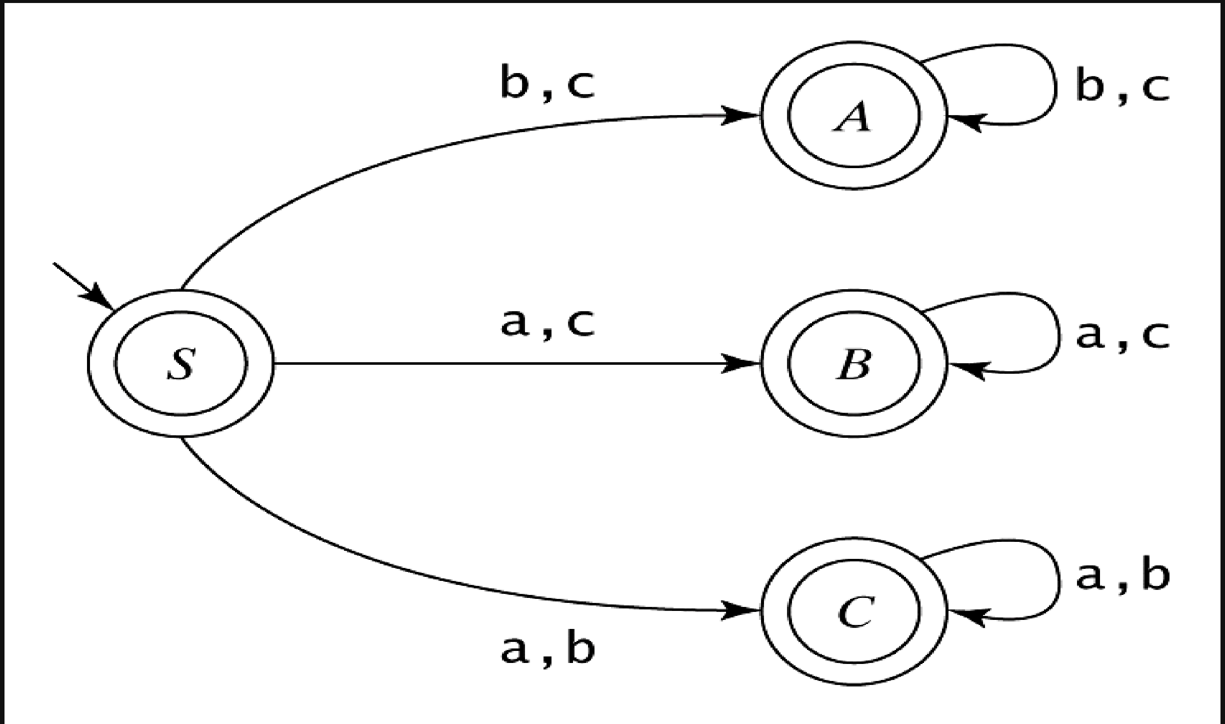
$C \rightarrow \mathtt{a}C$
$C \rightarrow \mathtt{b}C$
$C \rightarrow \varepsilon$

Example 2 – One Character Missing

$S \to \varepsilon$
$S \to \mathrm{a}B$
$S \to \mathrm{a}C$
$S \to \mathrm{b}A$
$S \to \mathrm{b}C$
$S \to \mathrm{c}A$
$S \to \mathrm{c}B$

$A \to \mathrm{b}A$
$A \to \mathrm{c}A$
$A \to \varepsilon$
$B \to \mathrm{a}B$
$B \to \mathrm{c}B$
$B \to \varepsilon$

$C \to \mathrm{a}C$
$C \to \mathrm{b}C$
$C \to \varepsilon$



Regular Languages and Regular Grammars

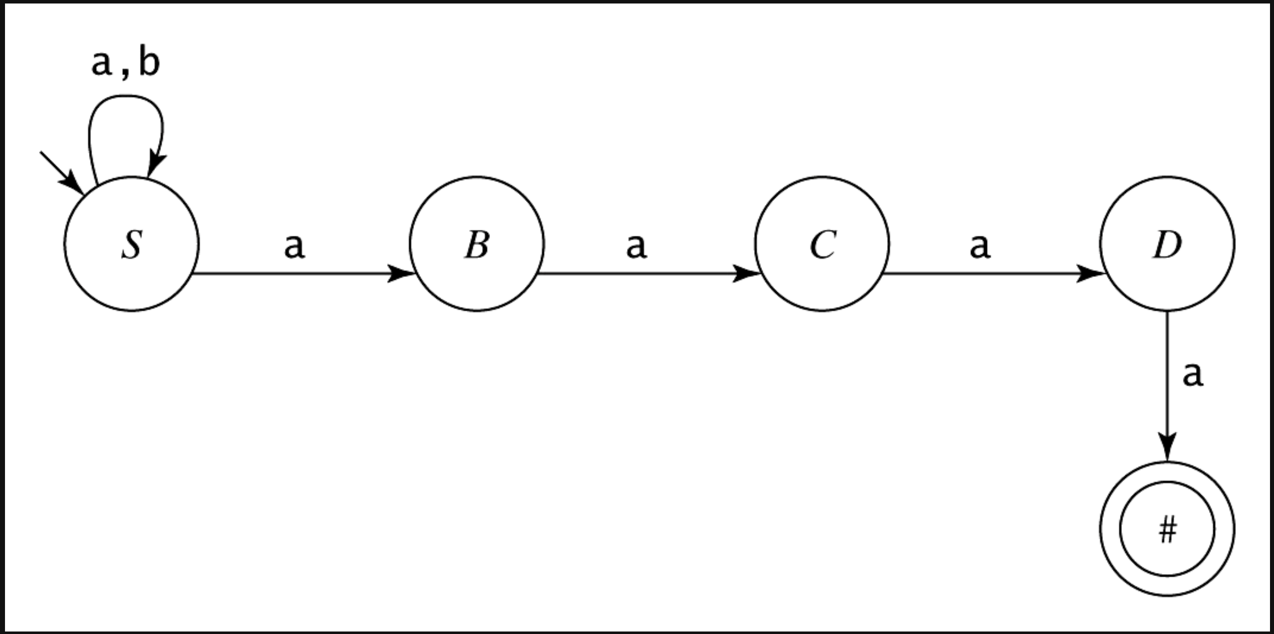
FSM* $\rightarrow$ *Regular grammar:

Show by construction that, for every FSM M there exists a regular grammar G such that $L(G) = L(M)$.

1. Make M deterministic → $M' = (Q, \Sigma, \delta, s, F)$.
   Construct $G = (V, \Sigma, R, S)$ from M'.
2. Create a nonterminal for each state in the M'.
   $V = Q$.
3. The start state becomes the starting nonterminal.
   $S = s$.
4. For each transition $\delta(T, a) = U$, make a rule of the form
   $T \rightarrow aU$.
5. For each accepting state T, make a rule of the form
   $T \rightarrow \varepsilon$.

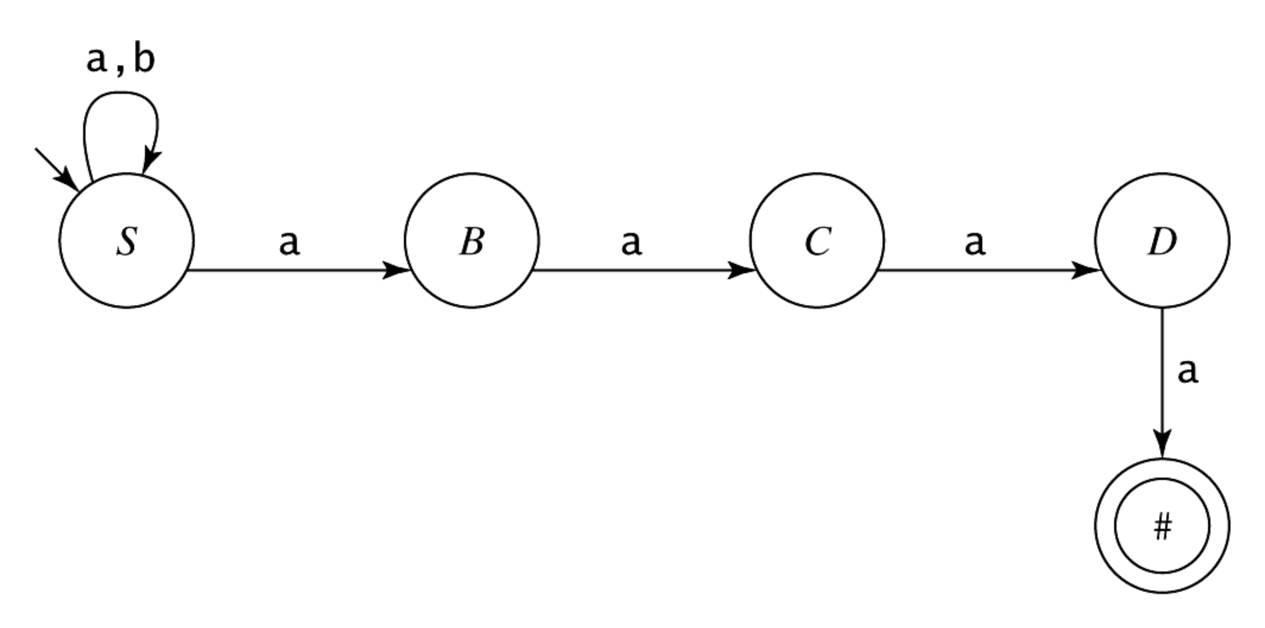
Strings that End with aaaa

$L = \{w \in \{\texttt{a}, \texttt{b}\}^* : w \text{ ends with the pattern } \texttt{aaaa}\}$.



Strings that End with aaaa

$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ ends with the pattern $\mathtt{aaaa}\}$.



$S \rightarrow \mathtt{a}S$

$S \rightarrow \mathtt{b}S$

$S \rightarrow \mathtt{a}B$

$B \rightarrow \mathtt{a}C$

$C \rightarrow \mathtt{a}D$

$D \rightarrow \mathtt{a}$