FA MINIMIZATIONS

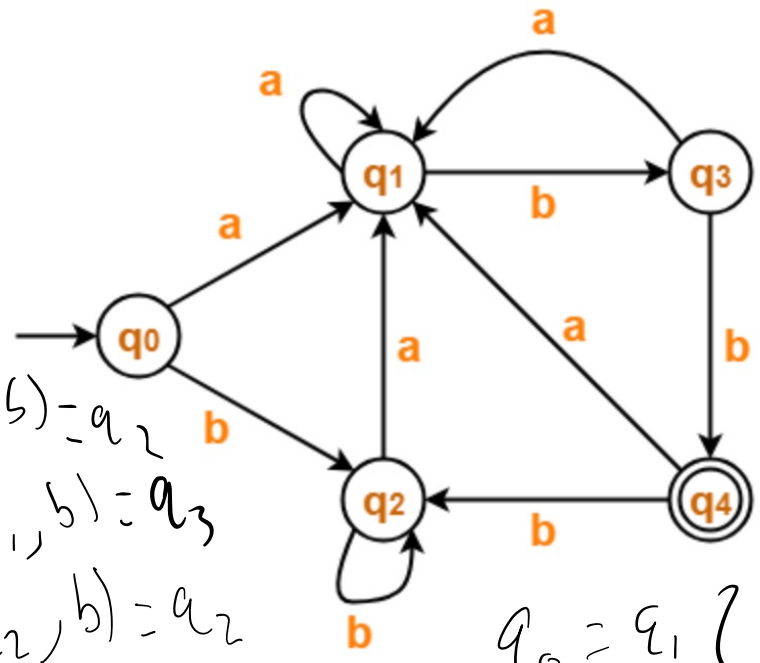
COMP 4200 – Formal Language

MINIMIZATION OF DFA

- Requirement → a minimal version of any DFA, which consist of minimum number of states possible.
- Lesser the number of states, higher the efficiency.
- But how?
  - Combine two states and make it as one state
  - How?
    - Condition: Two states can be combined only if two states are equivalent.
- Equivalent:
  - Two states are said to be equivalent if
    - $\delta(A,X) \rightarrow F \; and \; \delta(B,X) \rightarrow F \; or \; \delta(A,X) \nRightarrow F \; and \delta(B,X) \nRightarrow F$
    - X is the input string.
    - F is the final state.

EXAMPLE

	a	b
q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
q_4	q_1	q_2



$\delta(q_0, a) = q_1$ $\delta(q_0, b) = q_2$

$\delta(q_1, a) = q_1$ $\delta(q_1, b) = q_3$

$\delta(q_2, a) = q_1$ $\delta(q_2, b) = q_2$

$\delta(q_3, a) = q_1$ $\delta(q_3, b) = q_4$

0 equ. $\{q_0, q_1, q_2, q_3\}$ $\{q_4\}$

1 equ. $\{q_0, q_1, q_2\}$ $\{q_3\}$ $\{q_4\}$

2 equ. $\{q_0, q_2\}$ $\{q_1\}$ $\{q_3\}$ $\{q_4\}$

3 equ.

$q_0 = q_1$ } $q_0 = q_2$

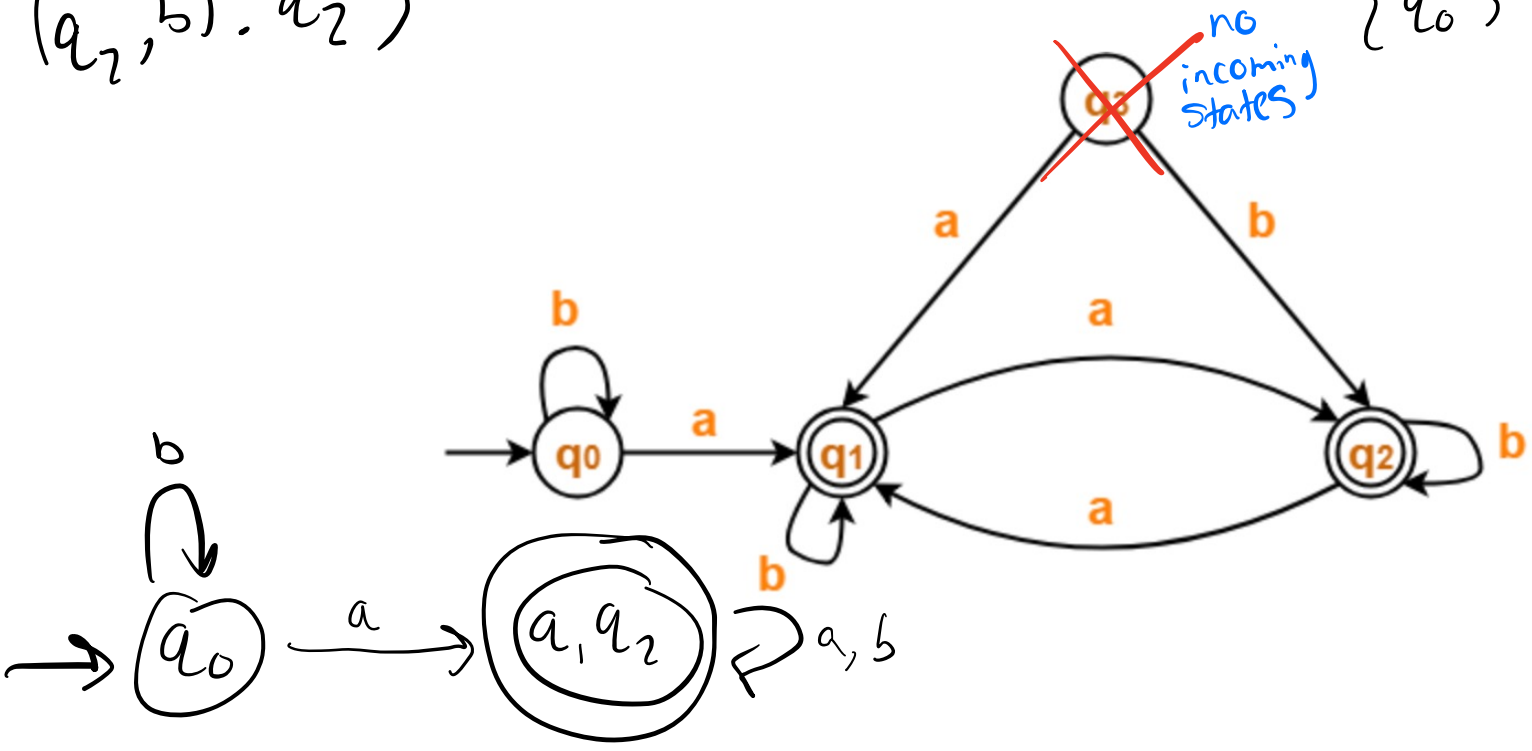
$q_1 = q_2$ }

$\delta(q_1, a) = q_2$
$\delta(q_2, a) = q_1$
$\delta(q_1, b) = q_1$
$\delta(q_2, b) = q_2$

EXAMPLE

0 eqv.
$\{q_0\}$ $\{q_1, q_2\}$

1 eqv.
$\{q_0\}$ $\{q_1, q_2\}$

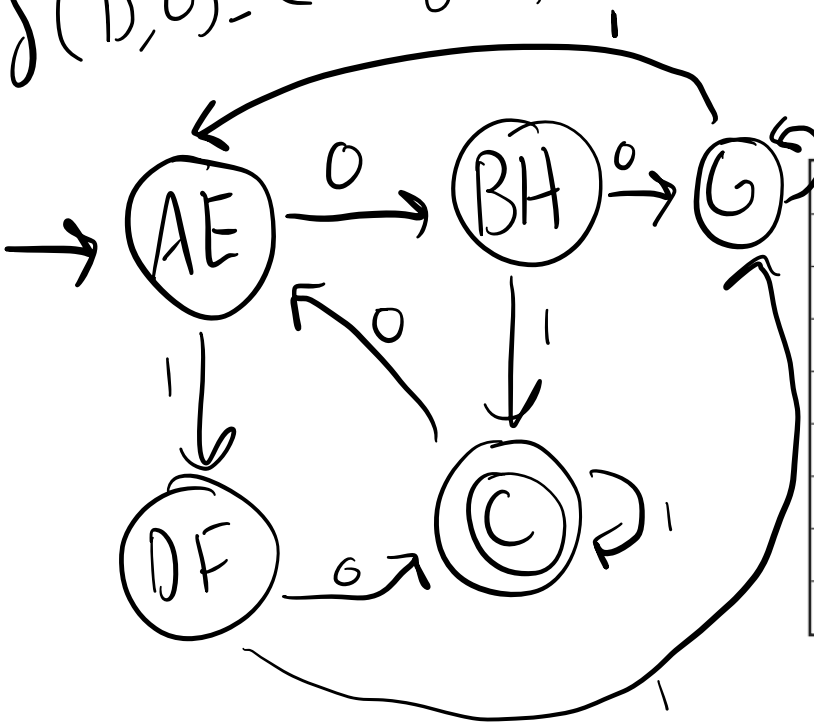


EXAMPLE

$\delta(A,0)=B$ $\delta(A,1)=F$

$\delta(B,0)=G$ $\delta(B,1)=C$

$\delta(D,0)=C$ $\delta(D,1)=G$



	0	1
→A	B	F
✗ B	G	C
Ⓒ	A	C
✗ D	C	G
E	H	F
✗ F	C	G
G	G	E
✗ H	G	C

0 eqv.

{A,B,D,E,F,G,H} {C}

1 eqv.

{A,E,G} {C}

{B,H}

{D,F}

2 eqv.

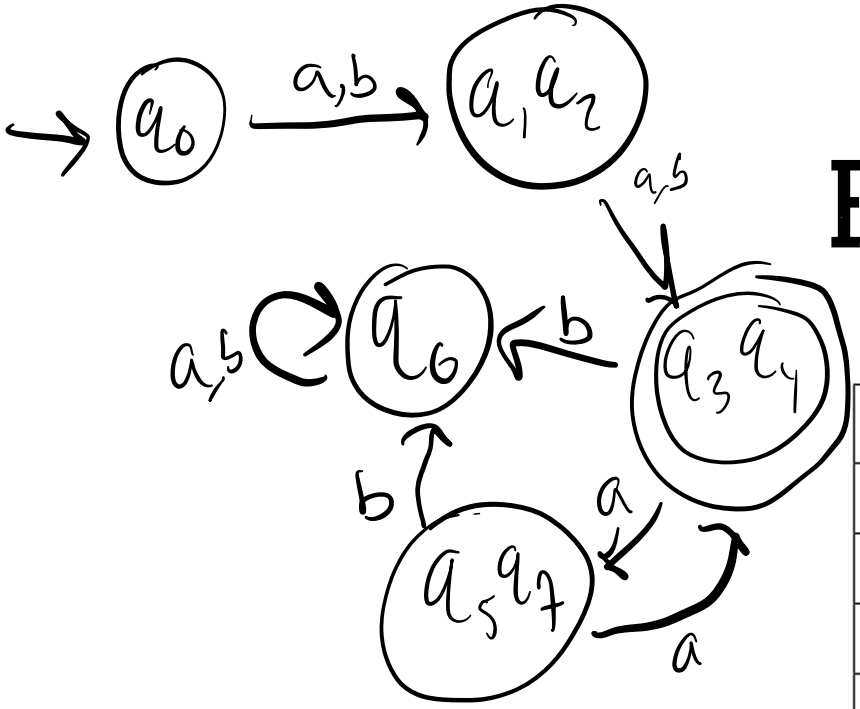
{A,E} {G} {C}

{B,H} {D,F}

3 eqv.

{A,E}, {G} {C}

{B,H} {D,F}



EXAMPLE

State	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_4	q_3
q_2	q_4	q_3
● q_3	q_5	q_6
● q_4	q_7	q_6
q_5	q_3	q_6
q_6	q_6	q_6
q_7	q_4	q_6

0 eqv.

$\{q_0, q_1, q_2, q_5, q_6, q_7\}$

$\{q_3, q_4\}$

1 eqv.

$\{q_0, q_6\}$ $\{q_3, q_4\}$

$\{q_1, q_2\}$

$\{q_5, q_7\}$

2 eqv.

$\{q_0\}$ $\{q_6\}$ $\{q_3, q_4\}$ $\{q_1, q_2\}$

$\{q_5, q_7\}$

3 eqv.

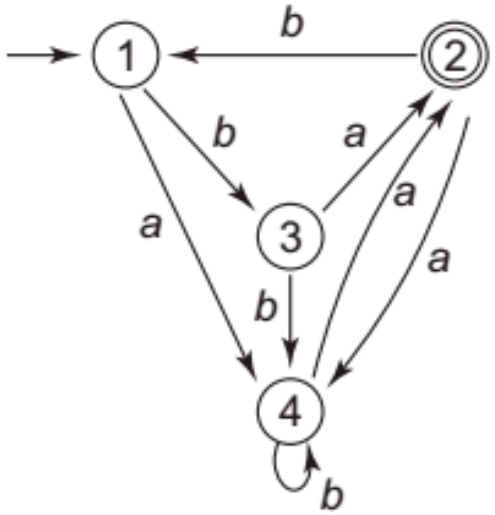
$\{q_0\}$ $\{q_6\}$ $\{q_3, q_4\}$ $\{q_1, q_2\}$

$\{q_5, q_7\}$

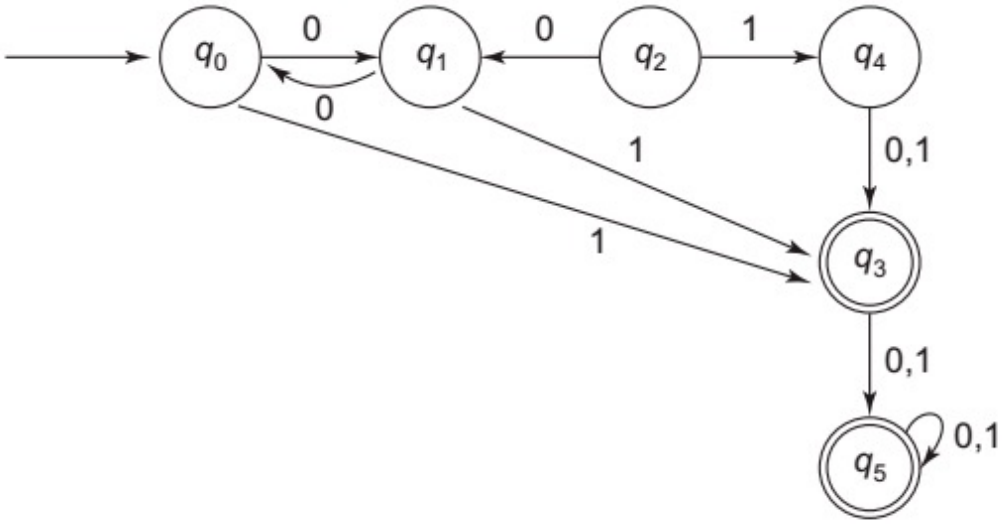
EXAMPLE

Input symbols and states	Next state	
	a	b
→*1	3	2
2	4	1
3	5	4
4	4	4
5	3	2

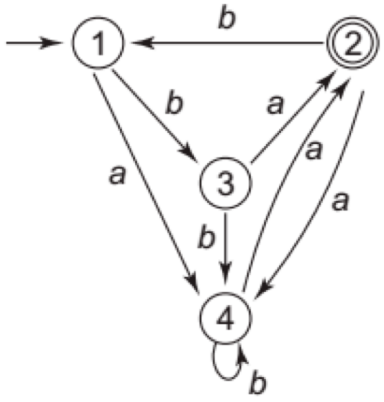
EXAMPLES TO TRY!



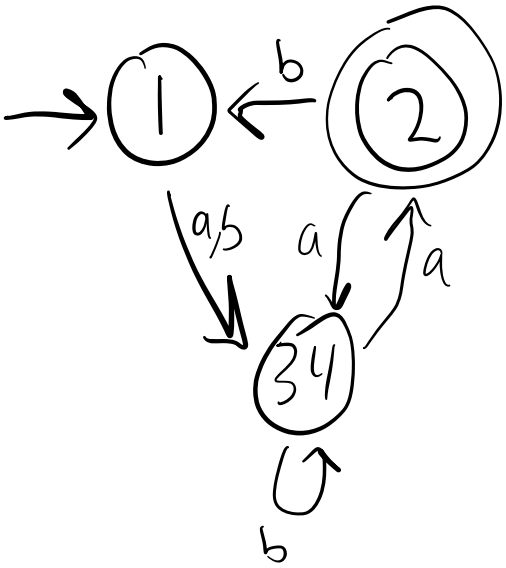
(1)



(2)



(1)



0 eqv.

{1,3,4} {2}

1 eqv.

{1} {2}

{3,4}

2 eqv.

{1} {2}

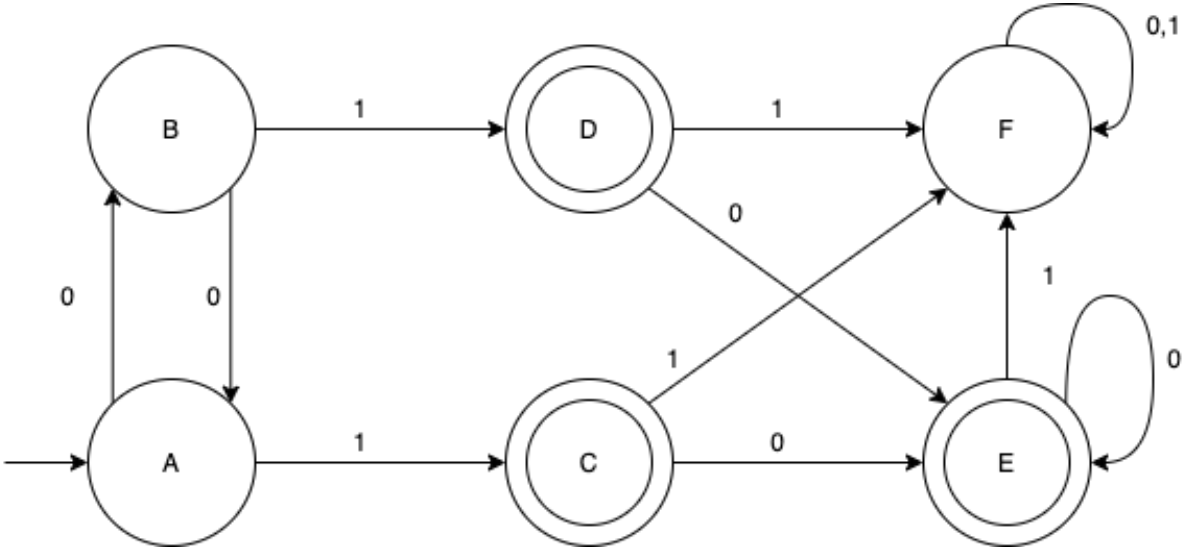
{3,4}

MYHILL NERODE THEOREM

- Steps:
  - Draw a table for all pairs of states (P,Q)
  - Mark all the points where P ∈ F and Q ∉ F
  - If there are any unmarked pairs (P,Q) such that $[\delta(P,X), \delta(Q,X)]$ is marked then mark [P,Q], where X is the input symbol.
  - Repeat until no marking can be made.
  - Combine all unmarked pairs and make them single state in the minimized DFA.

Transition table

	0	1
A*	B	C
B	A	D
C	E	F
D	E	F
E	E	F
F	F	F



$\delta(A,F)$: $\delta(A,0)=B$ $\delta(A,1)=C$
$\delta(F,0)=F$ $\delta(F,1)=F$

$\delta(F,B)$: $\delta(F,0)=F$ $\delta(F,1)=F$
$\delta(B,0)=A$ $\delta(B,1)=D$

	A	B	C	D	E	F
A	-	-	-	-	-	-
B	•	-	-	-	-	-
C	×	×	-	-	-	-
D	×	×	•	-	-	-
E	×	×	•	•	-	-
F	×	•	×	×	×	-

$\delta(B,A)$:
$\delta(B,0)=A$ $\delta(B,1)=D$
$\delta(A,0)=B$ $\delta(A,1)=C$

$\delta(C,D)$: $\delta(C,0)=E$ $\delta(C,1)=F$
$\delta(D,0)=E$ $\delta(D,1)=F$

$\delta(C,E)$: $\delta(C,0)=E$ $\delta(C,1)=F$
$\delta(E,0)=E$ $\delta(E,1)=F$

$\delta(D,E)$: $\delta(D,0)=E$ $\delta(D,1)=F$
$\delta(E,0)=E$ $\delta(E,1)=F$

	A	B	C	D	E	F
A	-	-	-	-	-	-
B		-	-	-	-	-
C	X		-	-	-	-
D				-	-	-
E					-	-
F						-

	A	B	C	D	E	F
A	-	-	-	-	-	-
B		-	-	-	-	-
C	X	X	-	-	-	-
D	X	X		-	-	-
E					-	-
F						-

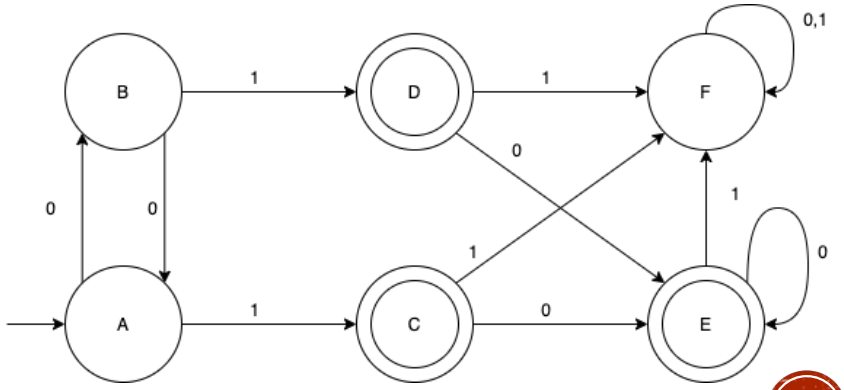
Result at end of step 2

	A	B	C	D	E	F
A	-	-	-	-	-	-
B		-	-	-	-	-
C	X	X	-	-	-	-
D	X	X		-	-	-
E	X	X			-	-
F			X	X	X	-

If there are any unmarked pairs (P,Q) such that $[\delta(P,X),\delta(Q,X)]$ is marked then mark [P,Q], where X is the input symbol.

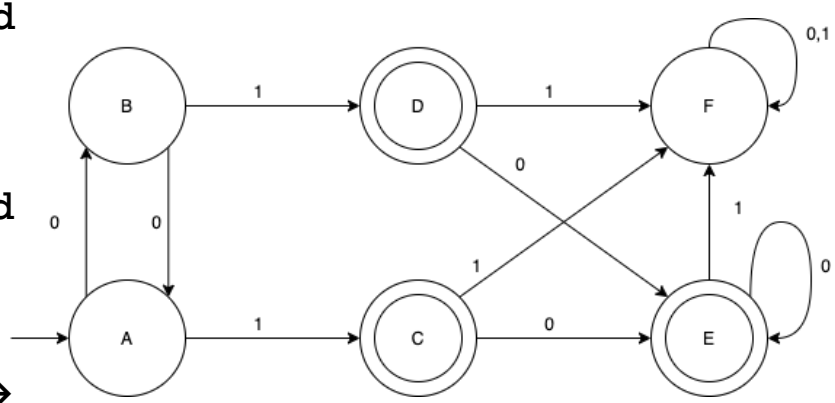
	A	B	C	D	E	F
A	-	-	-	-	-	-
B		-	-	-	-	-
C	X	X	-	-	-	-
D	X	X		-	-	-
E	X	X			-	-
F			X	X	X	-

(B,A): $\delta(B,0)=A, \delta(B,1)=D, \delta(A,0)=B, \delta(A,1)=C$
(D,C): $\delta(D,0)=E, \delta(D,1)=\mathrm{F}, \delta(C,0)=E, \delta(C,1)=F$
(E,C): $\delta(E,0)=E, \delta(E,1)=\mathrm{F}, \delta(C,0)=E, \delta(C,1)=F$
(E,D): $\delta(E,0)=E, \delta(E,1)=\mathrm{F}, \delta(D,0)=E, \delta(D,1)=F$
(F,A): $\delta(F,0)=F, \delta(F,1)=\mathrm{F}, \delta(A,0)=B, \delta(A,1)=C$
(F,B): $\delta(F,0)=F, \delta(F,1)=\mathrm{F}, \delta(B,0)=A, \delta(B,1)=D$



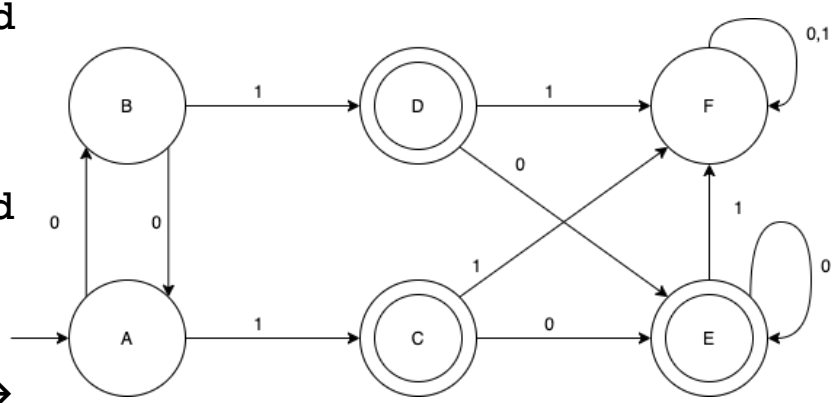
	A	B	C	D	E	F
A	-	-	-	-	-	-
B		-	-	-	-	-
C	X	X	-	-	-	-
D	X	X		-	-	-
E	X	X			-	-
F	X		X	X	X	-

(B,A): $\delta(B,0) = A, \delta(B,1) = D, \delta(A,0) = B, \delta(A,1) = C$ → is AB marked?
→ No, so don't mark
(D,C): $\delta(D,0) = E, \delta(D,1) = \mathrm{F}, \delta(C,0) = E, \delta(C,1) = F$ → EE not marked
→ don't mark
(E,C): $\delta(E,0) = E, \delta(E,1) = \mathrm{F}, \delta(C,0) = E, \delta(C,1) = F$ → EE not marked
→ don't mark
(E,D): $\delta(E,0) = E, \delta(E,1) = \mathrm{F}, \delta(D,0) = E, \delta(D,1) = F$ → EE not marked
→ don't mark
(F,A): $\delta(F,0) = F, \delta(F,1) = \mathrm{F}, \delta(A,0) = B, \delta(A,1) = C$ → FC is marked
→mark
(F,B): $\delta(F,0) = F, \delta(F,1) = \mathrm{F}, \delta(B,0) = A, \delta(B,1) = D$ → FA is marked → mark



	A	B	C	D	E	F
A	-	-	-	-	-	-
B		-	-	-	-	-
C	X	X	-	-	-	-
D	X	X		-	-	-
E	X	X			-	-
F	X	X	X	X	X	-

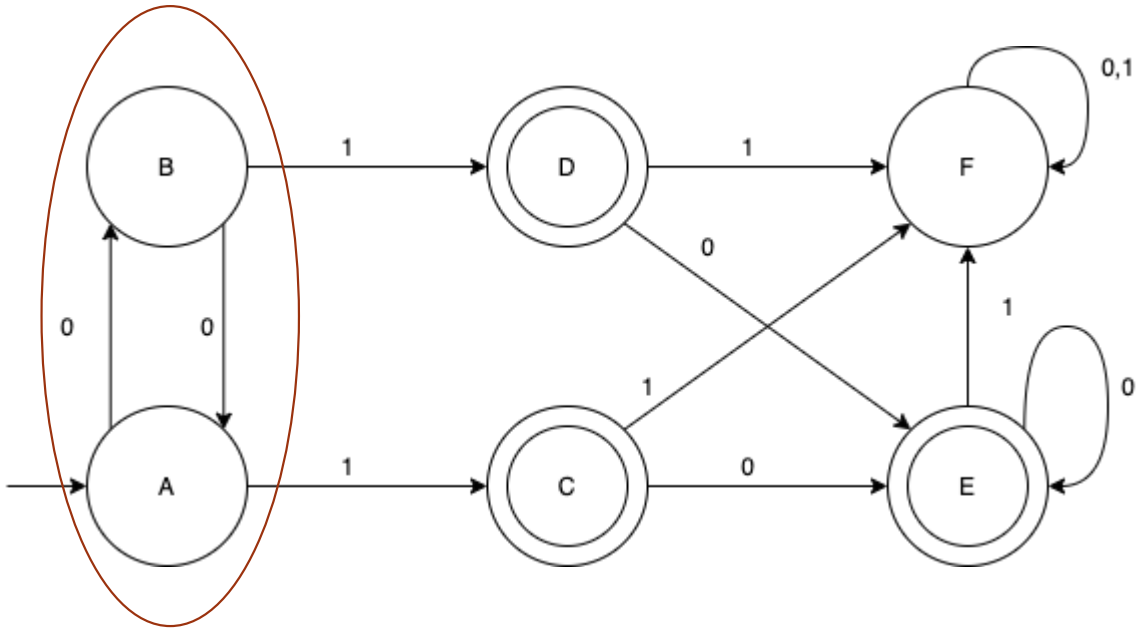
(B,A): $\delta(B,0) = A, \delta(B,1) = D, \delta(A,0) = B, \delta(A,1) = C$ → is AB marked?
→ No, so don't mark
(D,C): $\delta(D,0) = E, \delta(D,1) = \mathrm{F}, \delta(C,0) = E, \delta(C,1) = F$ → EE not marked
→ don't mark
(E,C): $\delta(E,0) = E, \delta(E,1) = \mathrm{F}, \delta(C,0) = E, \delta(C,1) = F$ → EE not marked
→ don't mark
(E,D): $\delta(E,0) = E, \delta(E,1) = \mathrm{F}, \delta(D,0) = E, \delta(D,1) = F$ → EE not marked
→ don't mark
(F,A): $\delta(F,0) = F, \delta(F,1) = \mathrm{F}, \delta(A,0) = B, \delta(A,1) = C$ → FC is marked
→mark
(F,B): $\delta(F,0) = F, \delta(F,1) = \mathrm{F}, \delta(B,0) = A, \delta(B,1) = D$ → FA is marked → mark



(A,B) (C,D) (D,E) (E,C)

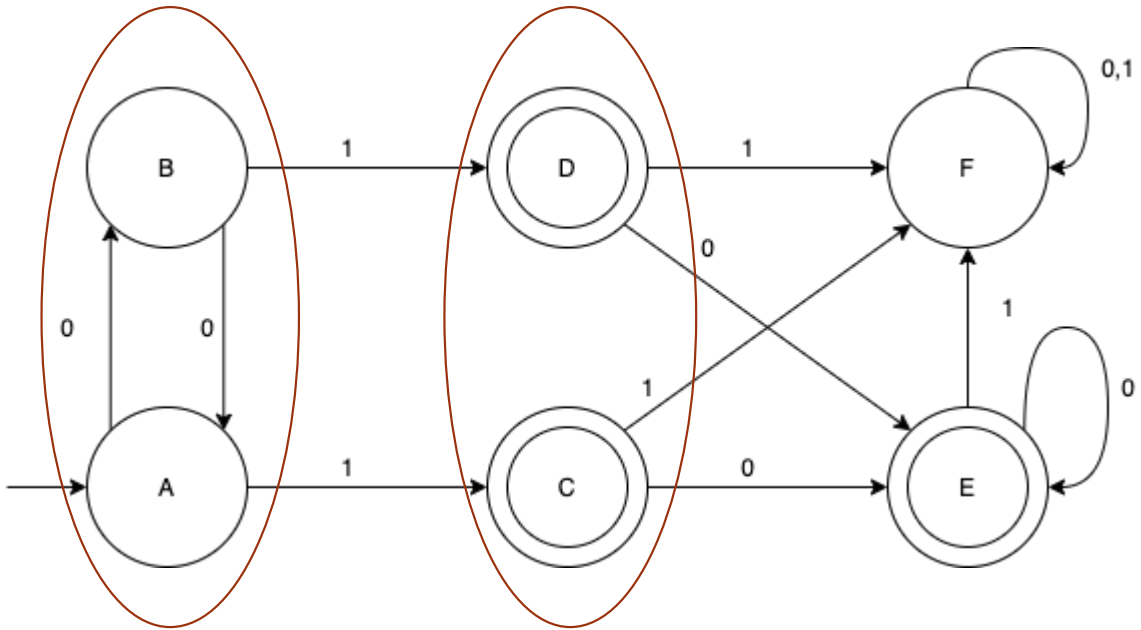
COMBINE ALL UNMARKED PAIRS AND MAKE THEM SINGLE STATE IN THE MINIMIZED DFA.

- (A,B), (D,C), (E,C), (E,D) → Unmarked pairs



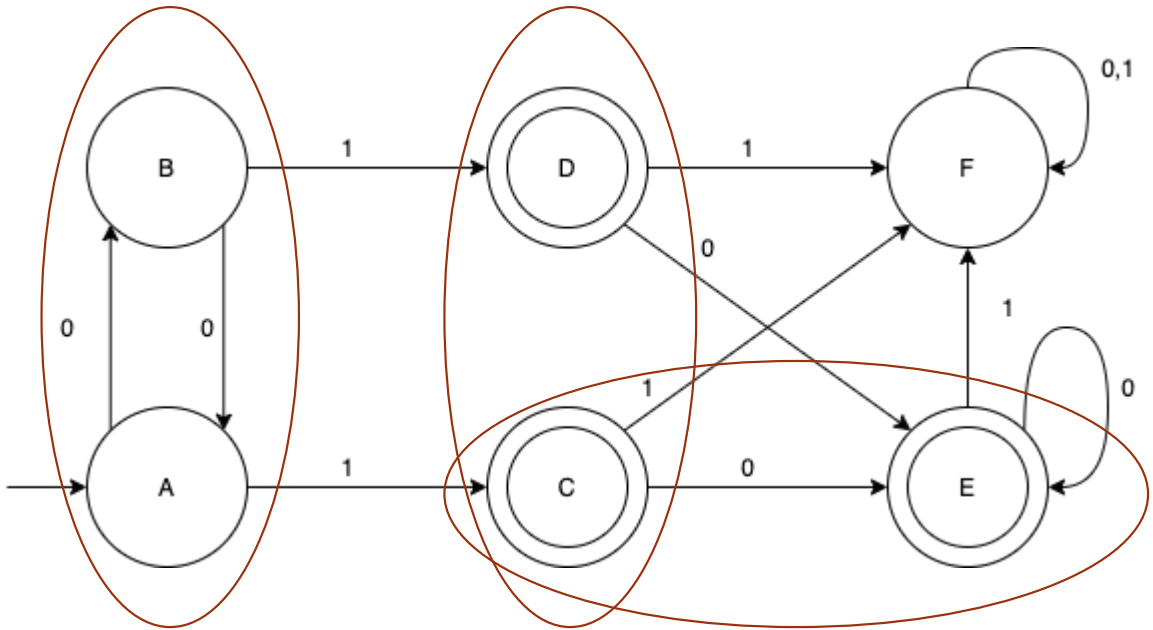
COMBINE ALL UNMARKED PAIRS AND MAKE THEM SINGLE STATE IN THE MINIMIZED DFA.

- (A,B), (D,C), (E,C), (E,D) → Unmarked pairs



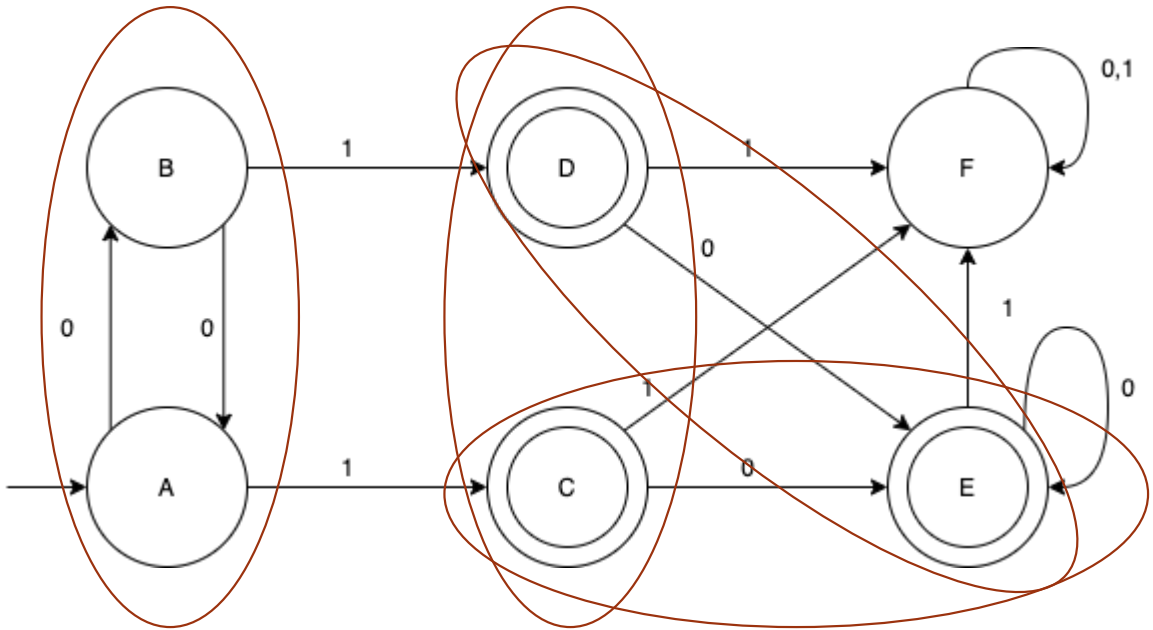
COMBINE ALL UNMARKED PAIRS AND MAKE THEM SINGLE STATE IN THE MINIMIZED DFA.

- (A,B), (D,C), (E,C), (E,D) → Unmarked pairs



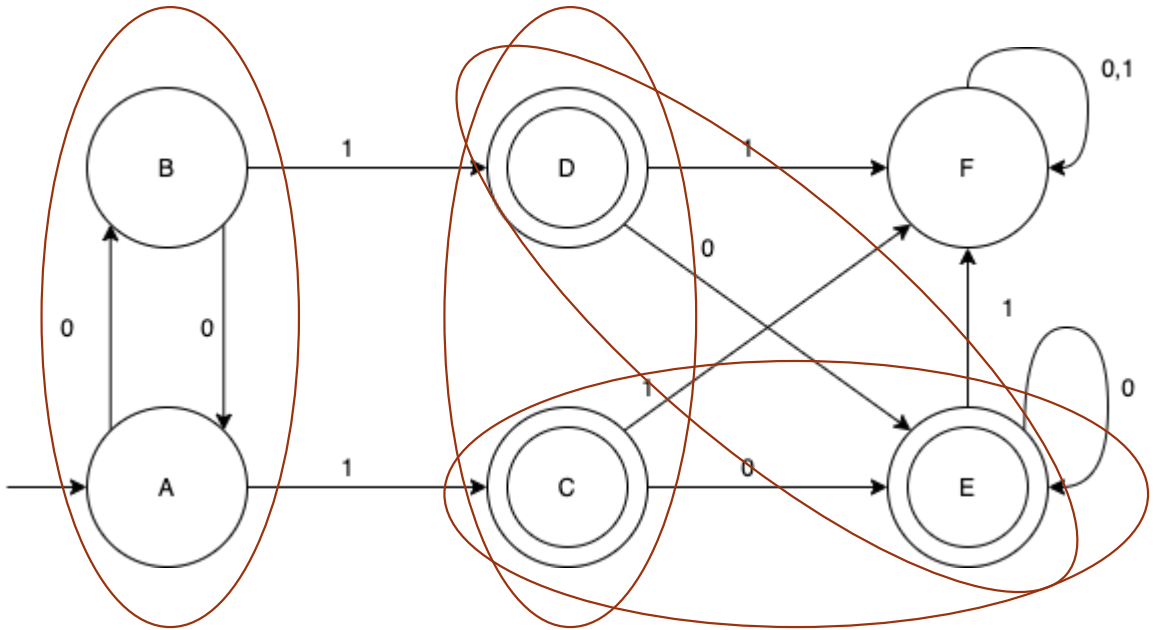
COMBINE ALL UNMARKED PAIRS AND MAKE THEM SINGLE STATE IN THE MINIMIZED DFA.

- (A,B), (D,C), (E,C), (E,D) → Unmarked pairs



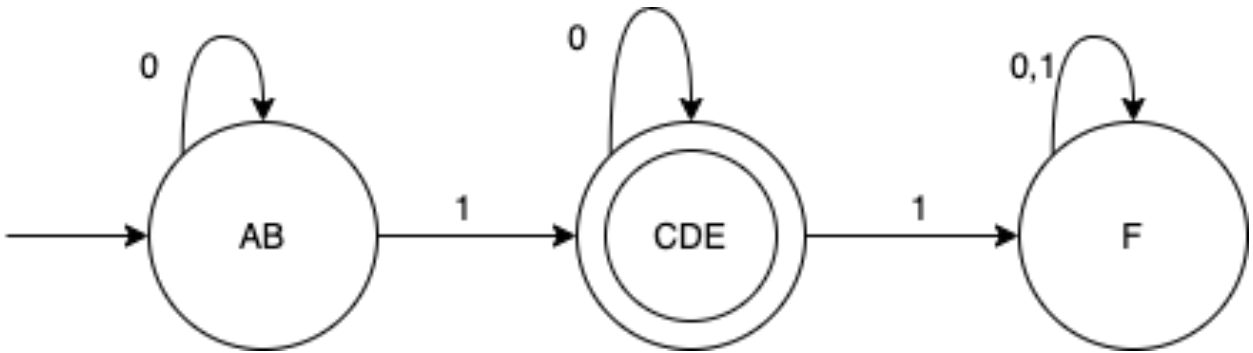
COMBINE ALL UNMARKED PAIRS AND MAKE THEM SINGLE STATE IN THE MINIMIZED DFA.

- (A,B), (D,C), (E,C), (E,D) → Unmarked pairs



(A,B), (C,D,E)

MINIMIZED DFA



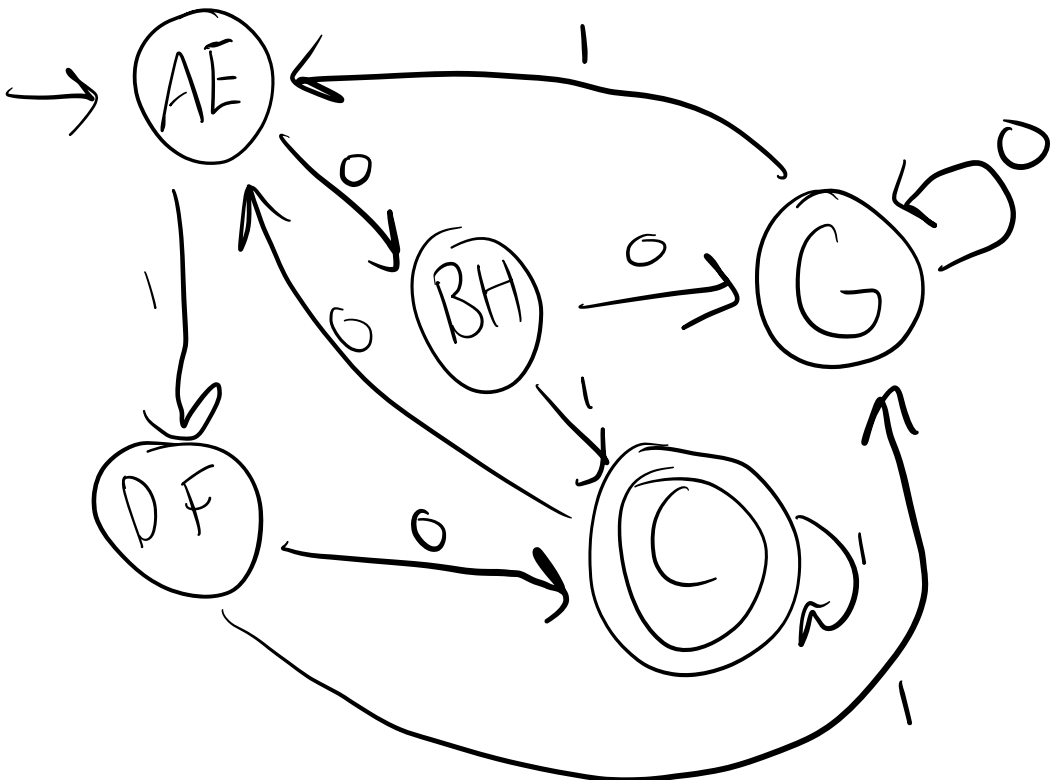
EXAMPLE

	0	1
→ A	B	F
B	G	C
Ⓒ	A	C
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C

	0	1
→A	B	F
B	G	C
ⓒ	A	C
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C

	A	B	C	D	E	F	G	H
A								
B	X							
C	X	X						
D	X	X	X					
E		X	X	X				
F	X	X	X		X			
G	X	X	X	X	X	X		
H	X		X	X	X	X	X	

(B, H) (D, F) (A, E)



EXAMPLE

State	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_4	q_3
q_2	q_4	q_3
q_3	q_5	q_6
q_4	q_7	q_6
q_5	q_3	q_6
q_6	q_6	q_6
q_7	q_4	q_6

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_0	-	-	-	-	-	-	-	-
q_1	X	-	-	-	-	-	-	-
q_2	X		-	-	-	-	-	-
q_3	X	X	X	-	-	-	-	-
q_4	X	X	X		-	-	-	-
q_5	X			X	X	-	-	-
q_6				X	X		-	-
q_7	X			X	X			-

INCLASS ASSIGNMENT

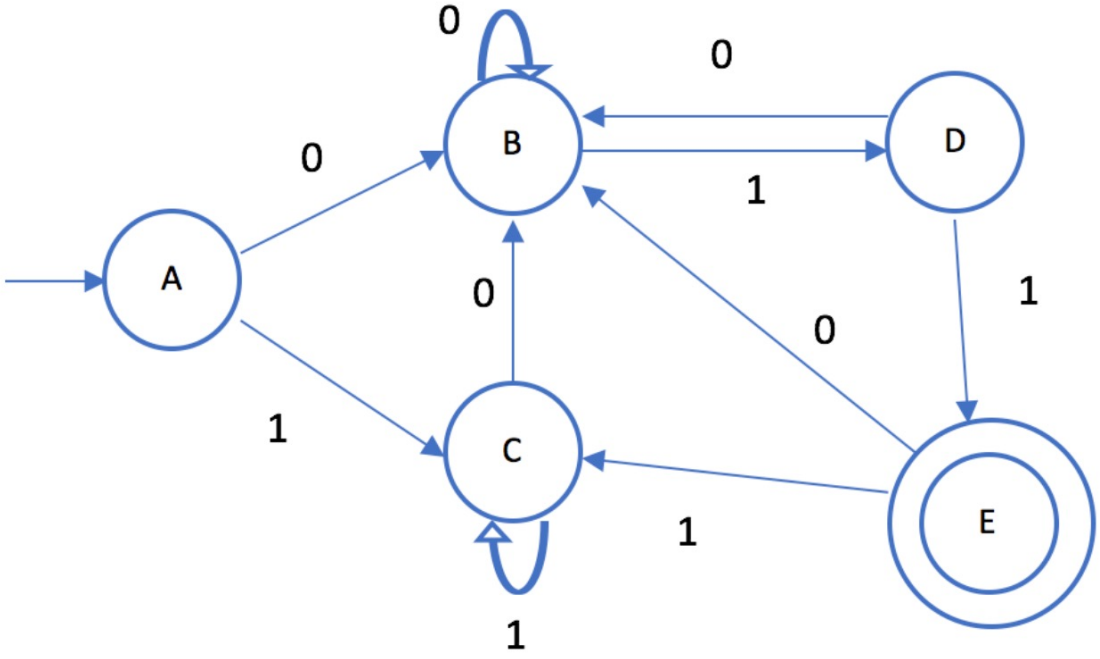
	a	b
A*	B	E
B	C	D
C	H	I
D	I	H
E	F	G
F	H	I
G	H	I
H	H	H
I	I	I

*→ start state, bold → accept states

EXAMPLE

Present State	Next State	
	I/P = a	I/P = b
→ A	B	F
B	A	F
C	G	A
D	H	B
E	A	G
F	H	C
G	A	D
H	A	C

EXAMPLE



EXAMPLE

State \ Input	a	b
→ 1	2	3
2	3	5
3	4	3
4	3	5
(5)	2	5

REGULAR EXPRESSIONS

- Regular expressions(RE) are a family of descriptions that can be used to capture the regular languages.
- Often provide a compact and human-readable description of the language.
- Used as the basis for numerous software systems (Perl, flex, grep, etc.)
- RE are used for representing certain set of strings in algebraic fashion.

RE

In arithmetic, we can use the operations + and x to build up expressions such as

$$(2 + 3) \times 4$$

We can use the regular operations to build up expressions describing languages, which are called ***regular expressions***.

$$(0 \cup 1)0^*$$

WHAT WOULD BE THE LANGUAGE FOR $(0 \cup 1)^*$?

ATOMIC REGULAR EXPRESSIONS

- The regular expressions begin with three simple building blocks.
- The symbol $\emptyset$ is a regular expression that represents the empty language .
- The symbol ε is a regular expression that represents the language $\{ \varepsilon \}$
  - This is not the same as $\emptyset$
- For any $a \in \Sigma$, the symbol a is a regular expression for the language { a }.

COMPOUND REGULAR EXPRESSIONS

- We can combine together existing regular expressions in four ways.
- If R_1 and R_2 are regular expressions, R_1R_2 is a regular expression for the concatenation of the languages of R_1 and R_2.
- If R_1 and R_2 are regular expressions, $R_1 \mid R_2$ is a regular expression for the union of the languages of R_1 and R_2.
- If R is a regular expression, R* is a regular expression for the Kleene closure of the language of R.
- If R is a regular expression, (R) is a regular expression with the same meaning as R.

RULES

- Any terminal symbols i.e., symbols $\in \Sigma$ *including* $\wedge$ *and* $\emptyset$ are RE (a,b,c,…. $\wedge, \emptyset$)
- Union of two RE is also a RE, [R_1,R_2 → (R_1+R_2)]
- Concatenation and Closure(*) of two RE is also RE, [R_1,R_2 → ($R_1 \circ R_2$), R^*]
- RE over Σ are precisely those obtained recursively by the application of above rules.
- Examples:
  - {0,1,2} → R = 0 + 1 + 2
  - {^, ab} → R = ^ab
  - {^, 0, 00, 000, 0000,….} → R = 0^*
  - {1,11,111,1111,…..} → R = 1^+

REGULAR EXPRESSION

- An algebraic-expression notation for describing (some)languages, rather than a machine representation.
- Languages described by regular expressions are exactly the FA-recognizable languages.
  - That's why FA-recognizable languages are called "regular".
- Definition: R is a regular expression over alphabet Σ exactly if R is one of the following: – a, for some a in Σ,
- ε,
- $\emptyset$,
- $(R_1 \cup R_2)$, where R_1 and R_2 are smaller regular expressions,
- $(R_1 \circ R_2)$, where R_1 and R_2 are smaller regular expressions, or
- (R_1^*), where R_1 is a smaller regular expression.

HOW REGULAR EXPRESSIONS DENOTE LANGUAGE

- Define the languages recursively, based on the expression structure:
- Definition:
  - $L(a) = \{ a \}$; one string, with one symbol a.
  - $L(\varepsilon) = \{ \varepsilon \}$; one string, with no symbols.
  - $L(\emptyset) = \emptyset$; no strings.
  - $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$
  - $L(R_1 \circ R_2) = L(R_1) \circ L(R_2)$
  - $L(R_1^*) = (L(R_1))^*$
- Example: Expression $((0 \cup 1) \varepsilon)^* \cup 0$ denotes language $\{ 0, 1 \}^* \cup \{ 0 \} = \{ 0, 1 \}^*$, all strings.
- Example: $(0 \cup 1)^* 111 (0 \cup 1)^*$ denotes $\{ 0, 1 \}^* \{ 111 \} \{ 0, 1 \}^*$, all strings with substring 111.

EXAMPLE

- Example: (((0 ∪ 1) ε)* ∪ 0)
- Abbreviations:
  - Sometimes omit parentheses, use precedence of operations: * highest, then ∪.
- Example: Abbreviate above as ((0 ∪ 1) ε)* ∪ 0
- Example: (0 ∪ 1)* 1 1 1 (0 ∪ 1)*

- Definition:
  - L(a) = { a }; one string, with one symbol a.
  - L(ε) = { ε}; one string, with no symbols.
  - L(∅) = ∅; no strings.
  - $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$
  - $L(R_1 \circ R_2) = L(R_1) \circ L(R_2)$
  - $L(R_1^*) = (L(R_1))^*$
- Example: L = strings over { 0, 1 } with odd number of 1s.
  - 0* 1 0* (0* 1 0* 1 0*)*
- Example: L = strings with substring 01 or 10.
  - (0 ∪1)* 01 (0 ∪1)* ∪(0 ∪1)* 10 (0 ∪1)*
  - Abbreviate (writing Σfor (0 ∪1)): Σ* 01 Σ* ∪Σ* 10 Σ*

- Example: L = strings with neither substring 01 or 10.
  - Can't write complement. – But can write: 0* ∪1*.
- Example: L = strings with no more than two consecutive 0s or two consecutive 1s
  - Would be easy if we could write complement.
  - (ε∪1 ∪11) ((0 ∪00) (1 ∪11))* (ε∪0 ∪00) – Alternate one or two of each.
- L = string ending with 00 on Σ
- L = string has any number of a's and b's but no null string.

OPERATOR PRECEDENCE

- Regular expression operator precedence:

(R)

R*

R_1R_2

$R_1 \mid R_2$

- So ab*c|d is parsed as ((a(b*))c)|d

REGULAR EXPRESSIONS, FORMALLY

- The language of a regular expression is the language described by that regular expression.
- Formally:
  - $\mathcal{L}(\varepsilon) = \{\varepsilon\}$
  - $\mathcal{L}(\ \ ) =$
  - $\mathcal{L}(a) = \{a\}$
  - $\mathcal{L}(R_1R_2) = \mathcal{L}(R_1)\ \mathcal{L}(R_2)$
  - $\mathcal{L}(R_1 \mid R_2) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$
  - $\mathcal{L}(R^*) = \mathcal{L}(R)^*$
  - $\mathcal{L}((R)) = \mathcal{L}(R)$

DECODING REGULAR EXPRESSION

- Let $\Sigma = \{0, 1\}$
- Let L = { w ∈ Σ* | w contains 00 as a substring }

$$(0 \mid 1)^*00(0 \mid 1)^*$$

1101110010 1

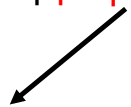
0000

1111101111 00 11111

EXAMPLE

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

Let $L = \{ w \in \Sigma^* \mid |w| = 4 \}$



The length of a string w is denoted $|w|$

$\Sigma\Sigma\Sigma\Sigma / \Sigma^4$

0000

1010

1111

1000

SOME MORE EXAMPLES

In the following instances, we assume that the alphabet is {0,1}.

- 0*10* = { w | w contains a single 1 }
- Σ*1 Σ* = { w | w contains at least one 1 }
- 01 ∪ 10 = {01, 10}
- (0 ∪ ε)(1 ∪ ε) = {ε, 0, 1, 01}.
- If we let R be any regular expression, we have the following identities.
  - R U ∅ = R.
    - Adding the empty language to any other language will not change it.
  - R_1 ° ε = R.
    - Joining the empty string to any string will not change it.
  - R U ε may not equal R.
    - For example, if R = 0, then L(R) = {0} but L(R U ε) = {0, ε}.
  - R_1 ° ε may not equal R.
    - For example, if R = 0, then L(R) = {0} but L(R ° ∅) = ∅

- 0^*10^*
- $\Sigma^*1\Sigma^*$
- $\Sigma^*001\Sigma^*$
- $1^*(01^+)^*$
- $(\Sigma\Sigma)^*$
- $(\Sigma\Sigma\Sigma)^*$
- $01 \cup 10 = \{01, 10\}$.
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$
- $(0 \cup \varepsilon)1^*$
- $(0 \cup \varepsilon)(1 \cup \varepsilon)$
- $1^*\emptyset$
- $\emptyset^*$

- $0^*10^* = \{w \mid w \text{ contains a single } 1\}$.
- $\Sigma^*1\Sigma^* = \{w \mid w \text{ has at least one } 1\}$.
- $\Sigma^*001\Sigma^* = \{w \mid w \text{ contains the string } 001 \text{ as a substring}\}$.
- $1^*(01^+)^* = \{w \mid \text{every } 0 \text{ in } w \text{ is followed by at least one } 1\}$.
- $(\Sigma\Sigma)^* = \{w \mid w \text{ is a string of even length}\}$
- $(\Sigma\Sigma\Sigma)^* = \{w \mid \text{the length of } w \text{ is a multiple of } 3\}$.
- $01 \cup 10 = \{01, 10\}$.
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$.
- $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$ → The expression $0 \cup \varepsilon$ describes the language $\{0, \varepsilon\}$, so the concatenation operation adds either 0 or ε before every string in 1^*.
- $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$.
- $1^*\emptyset = \emptyset$ → Concatenating the empty set to any set yields the empty set.
- $\emptyset^* = \{\varepsilon\}$ → The star operation puts together any number of strings from the language to get a string in the result. If the language is empty, the star operation can put together 0 strings, giving only the empty string.

Some more examples of REs and their corresponding languages.

R	
01	
$01 + 1$	
$(01 + \epsilon)1$	
$(0 + 10)^*(\epsilon + 1)$	

Some more examples of REs and their corresponding languages.

$\mathbf{R}$	$\mathbf{L(R)}$
01	$\{01\}$
$01 + 1$	$\{01, 1\}$
$(01 + \epsilon)1$	$\{011, 1\}$
$(0 + 10)^*(\epsilon + 1)$	$\{\epsilon, 0, 10, 00, 001, 010, 0101, \ldots\}$

IDENTITIES

- $\emptyset + R = R$
- $\emptyset R + R\emptyset = \emptyset$
- $\varepsilon R + R\varepsilon = R$
- $\varepsilon^* = \varepsilon \; and \; \emptyset^* = \varepsilon$
- $R + R = R$
- $R^*R^* = R^*$
- $RR^* = R^*R$
- $(R^*)^* = R^*$
- $\varepsilon + RR^* = \varepsilon + R^*R = R^*$
- $(PQ)^* = (P^*Q^*)^* = (P^* + Q^*)^*$
- $(P + Q)R = PR + QR \; and \; R(P + Q) = RP + RQ$

$R^+ \rightarrow Closure \; excluding \; \varepsilon \; symbol.$

EQUIVALENCE WITH FA

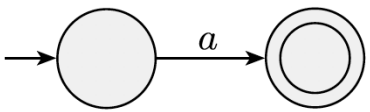
EQUIVALENCE WITH FINITE AUTOMATA

- A language is regular if and only if some regular expression describes it.
- This theorem has two directions.
  - If a language is described by a regular expression, then it is regular.
  - If a language is regular, then it is described by a regular expression.

IF A LANGUAGE IS DESCRIBED BY A REGULAR EXPRESSION, THEN IT IS REGULAR.

- We have a RE, R describing some language A. We show how to convert R into an NFA recognizing A.
- First, convert R into NFA N. We consider 6 cases from formal definition.

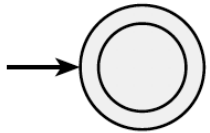
1. R = a for some a ∈ Σ. Then L(R) = {a}, the NFA recognizes L(R).



- Formally, N = {q1, q2}, Σ, δ, q1, {q2}, Accepts only a.

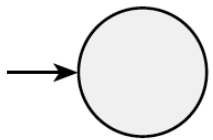
where we describe δ by saying that δ(q1, a) = {q2} and that δ(r, b) = ∅ for r ≠ q1 or b ≠ a.

2. $R = \varepsilon$. Then $L(R) = \{\varepsilon\}$, and the following NFA recognizes $L(R)$.



- Formally, $N = \{q1\}, \Sigma, \delta, q1, \{q1\}$, where $\delta(r, b) = \emptyset$ for any r and b.

3. $R = \emptyset$. Then $L(R) = \emptyset$, and the following NFA recognizes $L(R)$.



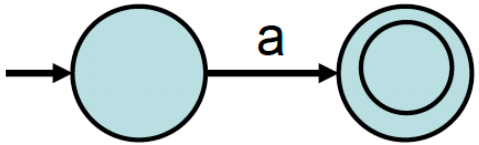
- Formally, $N = \{\{q\}, \Sigma, \delta, q, \emptyset,\}$ where $\delta(r, b) = \emptyset$ for any r and b.

4. $R = R_1 \cup R_2$.
5. $R = R_1 \circ R_2$.
6. $R = R^*$

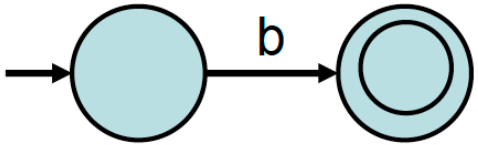
BUILDING AN NFA FROM THE REGULAR EXPRESSION

BASE CASES

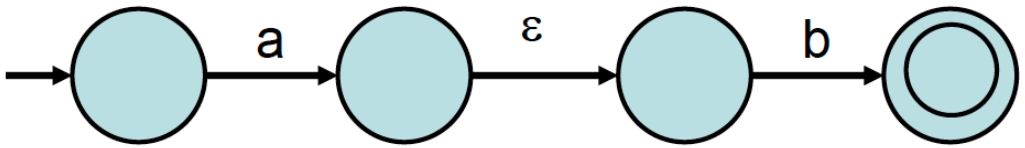
- a,

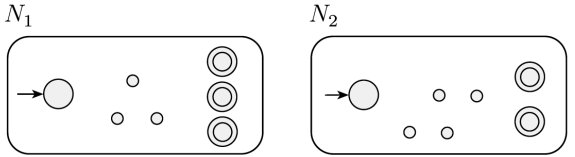


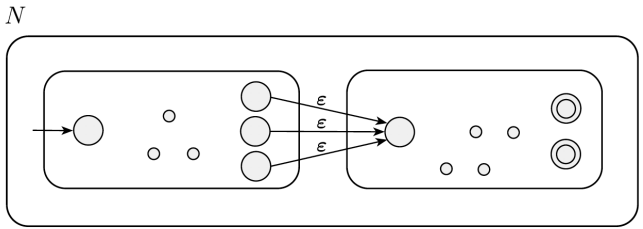
- b,

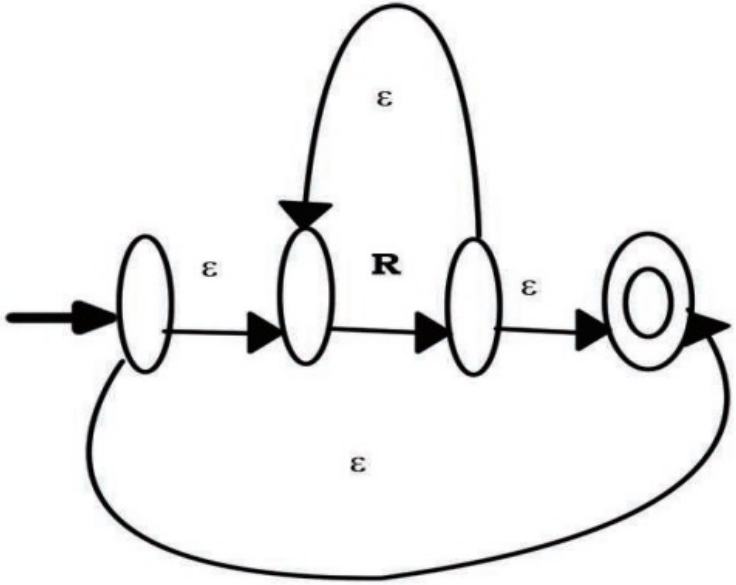


- ab,

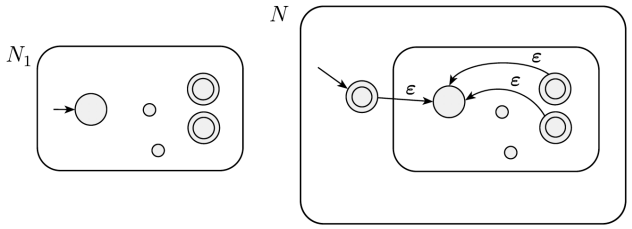


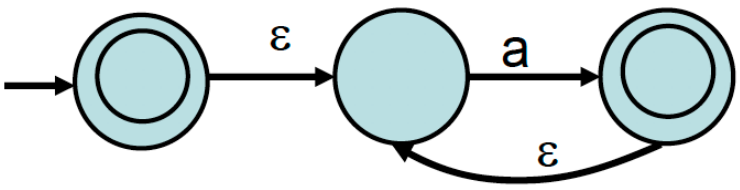




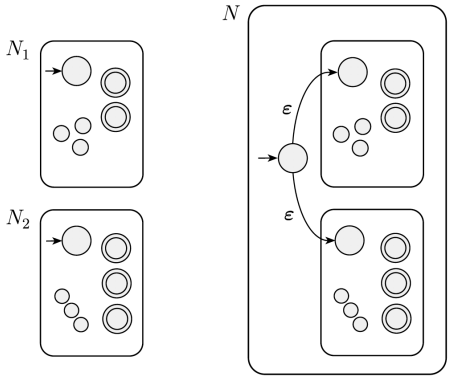


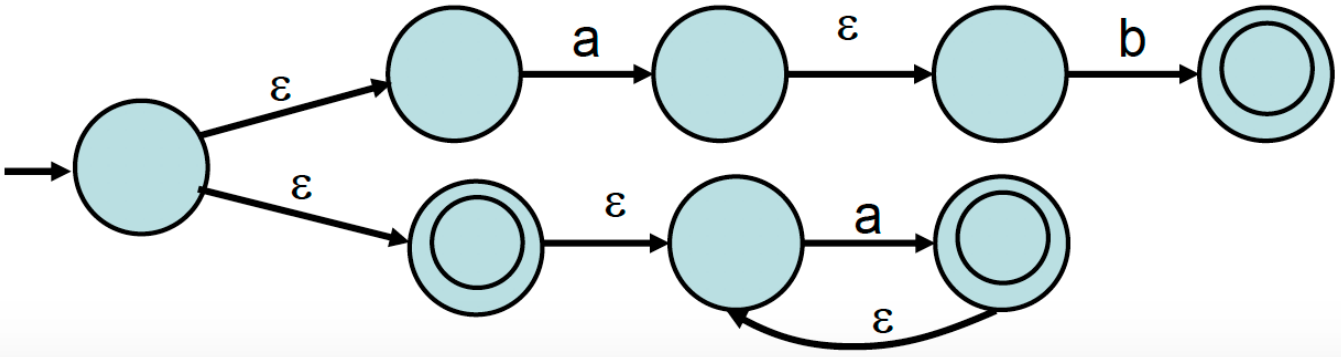
r=R*





L = AB ∪ A*

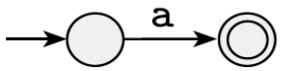




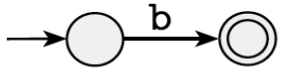
$(AB \cup A)^*$

(AB ∪ A)*

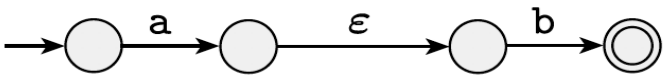
a



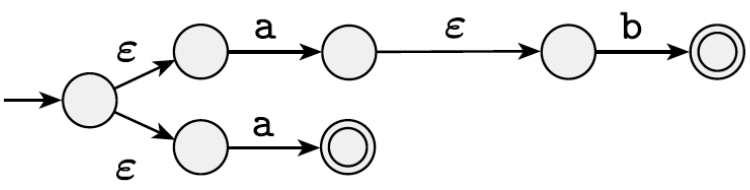
b



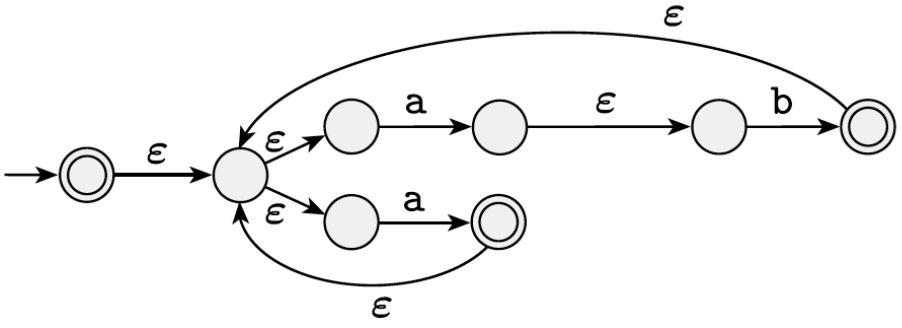
ab



ab ∪ a



(ab ∪ a)*

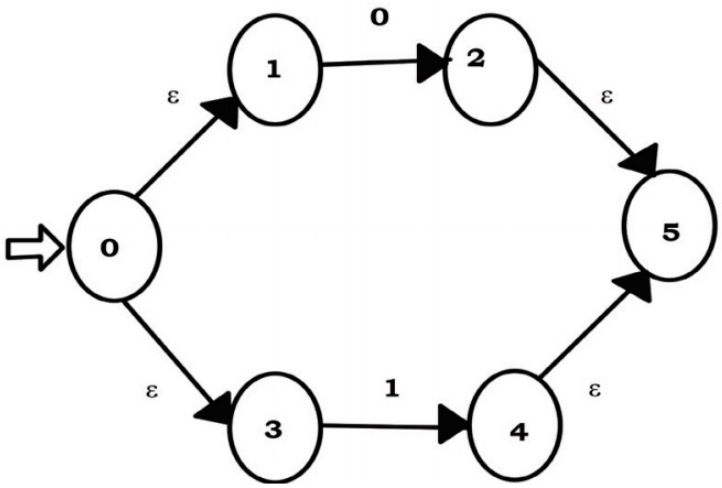


WHY?

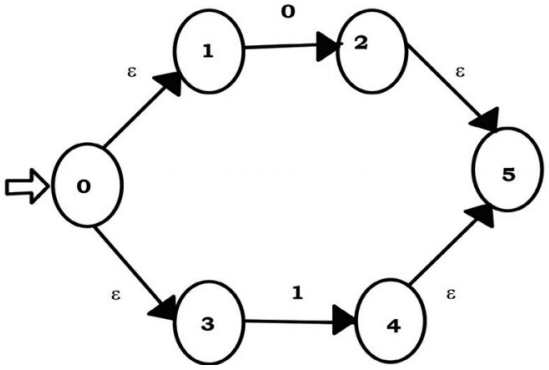
- Many software tools work by matching regular expressions against text.
- One possible algorithm for doing so:
  - Convert the regular expression to an NFA.
  - (Optionally) Convert the NFA to a DFA using the subset construction.
  - Run the text through the finite automaton and look for matches.
- Runs extremely quickly!

BASIC CONVERSIONS

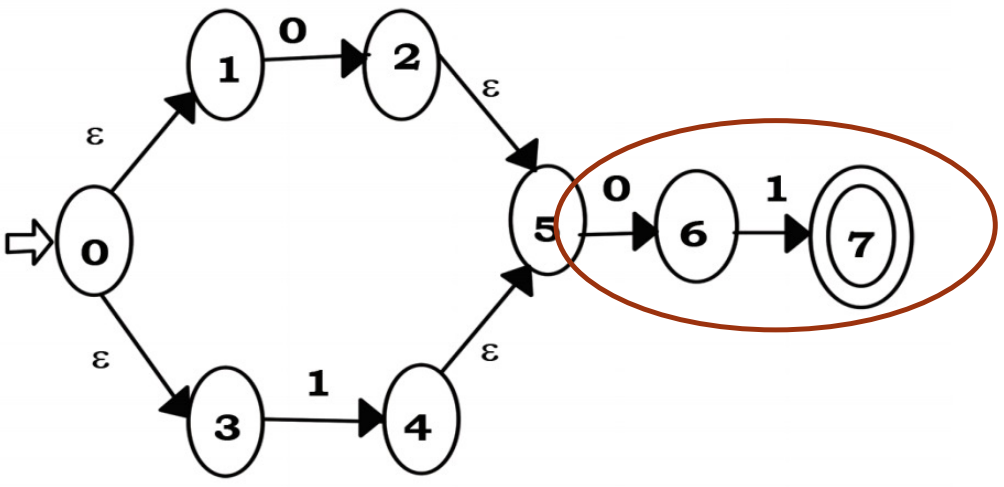
- 0 + 1



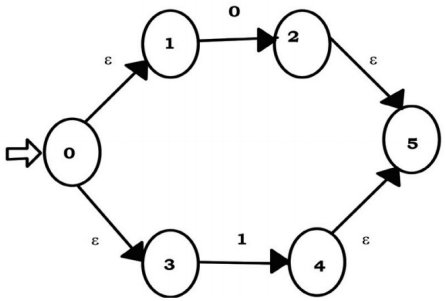
- (0+1)01
  - (0+1)



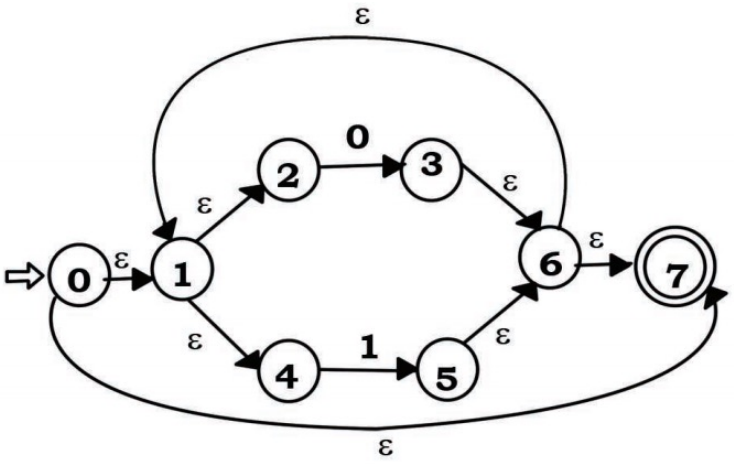
  - (0+1)01



- (0+1)*
  - (0+1)



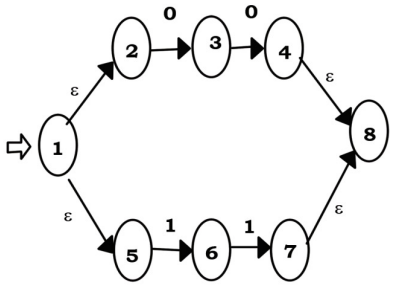
  - (0+1)*



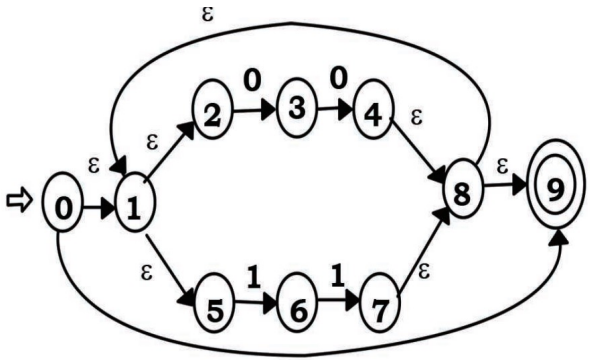
$$(00+11)^*$$

$(00+11)^*$

- Step 1: (00+11)

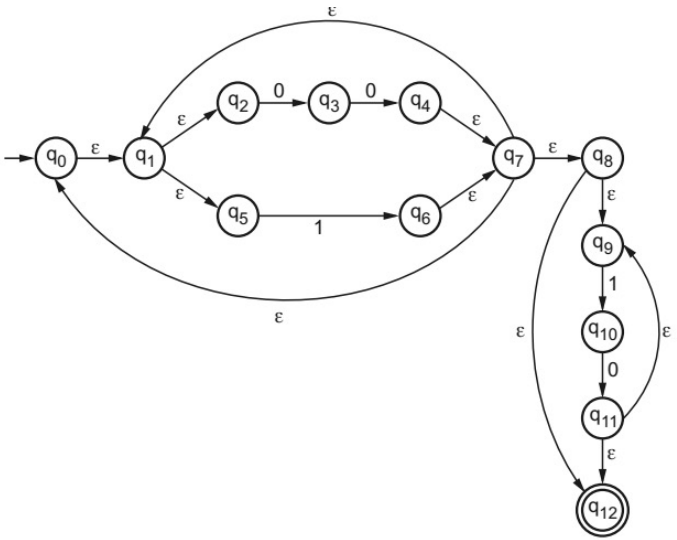


- Step 2: $(00+11)^*$



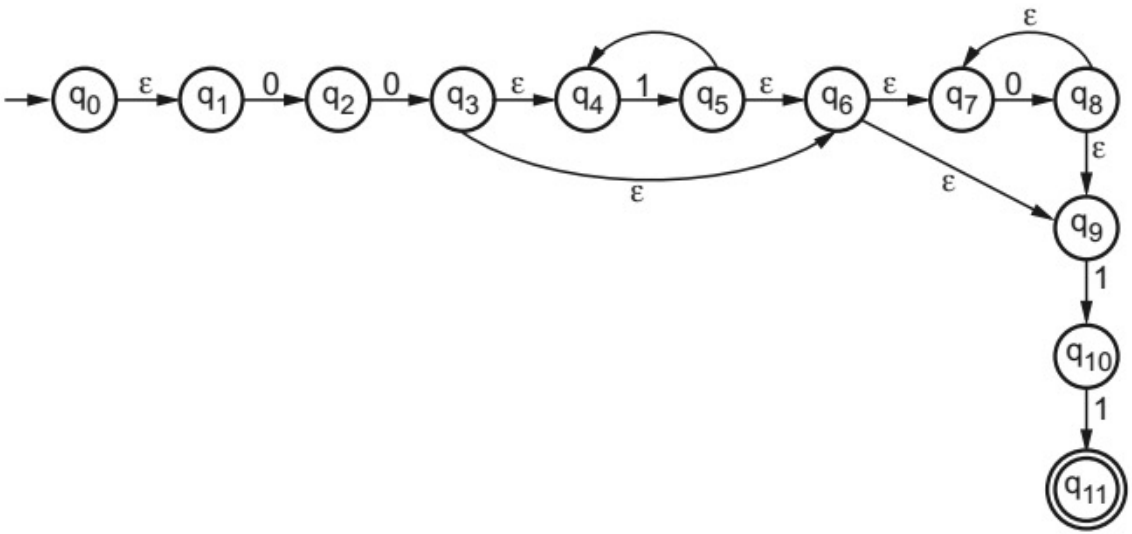
$$(00 + 1)^* (10)^*$$

(00 + 1)* (10)*



001*0*11

001*0*11



TRY YOURSELF!

- (a ∪ b)*aba
- (000)* 1 U (00)*1
- 00(0 U 1)*
- (ab U c)*b
- (0+1)* (00+11) (0+1)*
- (ab+c*)*b
- a(abb)* U b
- a(bc)*