

MINIMIZATION AND REGULAR EXPRESSIONS

COMP 4200 – Formal Language



3 equiv.

$$\overbrace{\{A, B\} \{E\} \{H\}}^{\{C\} \{D\} \{F\} \{G\}}$$

1 equiv.

$$\overbrace{\{A, B\} \{E\} \{H\}}^{\{C\} \{D\} \{F\} \{G\}}$$

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

0 equiv.

$$\overbrace{\{A, B, E, H\}}^{\{C\} \{D\} \{F, G\}}$$

1 equiv.

$$\overbrace{\{A, B, E, H\}}^{\{C\} \{D\} \{F, G\}}$$

2 equiv.

$$\overbrace{\{A, B, E\} \{H\}}^{\{F\} \{G\}}$$



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

$$\begin{aligned} \delta(A, B) &= (A, 0) = B & (A, 1) &= F \\ & (B, 0) = A & (B, 1) &= F \end{aligned}$$

$$\begin{aligned} \delta(A, E) &= (A, 0) = B & (A, 1) &= F \\ & (E, 0) = A & (E, 1) &= G \end{aligned}$$

$$\begin{aligned} \delta(C, D) &= (C, 0) = G & (C, 1) &= A \\ & (D, 0) = H & (D, 1) &= B \end{aligned}$$

$$\begin{aligned} \delta(A, H) &= (A, 0) = B & (A, 1) &= F \\ & (H, 0) = A & (H, 1) &= C \end{aligned}$$

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

EXAMPLE

→

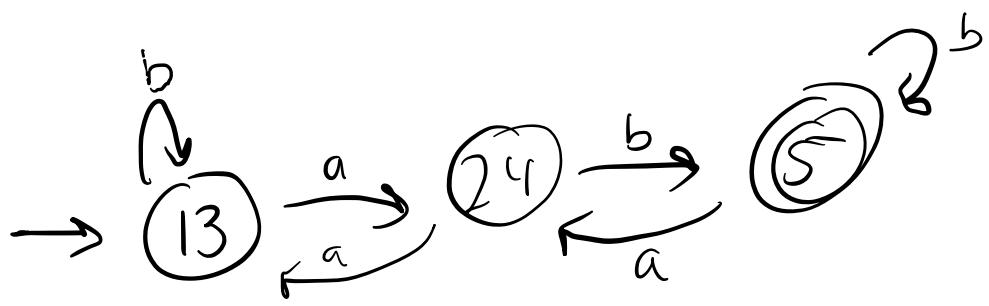
Input State \	a	b
1	2	3
2	3	5
3	4	3
4	3	5
(5)	2	5

0 equiv.
 $\overline{\{1, 2, 3, 4\}} \{5\}$

1 equiv.
 $\overline{\{1, 3\}} \{2, 4\} \{5\}$

2 equiv.
 $\{1, 3\} \{2, 4\} \{5\}$





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 \times 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**.

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



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

↳ repetition,
clean

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

$$\begin{matrix} \{0\} & \{1\} \\ \checkmark \\ \cup \\ \{0, 1\} \\ (01)^* & (10)^* \end{matrix}$$

$$\Sigma = 01$$

$$\Sigma^* = (01)^*$$



ATOMIC REGULAR EXPRESSIONS

- The regular expressions begin with three simple building blocks.
✓ different from ϵ
- The symbol \emptyset is a regular expression that represents the empty language .
- The symbol ϵ is a regular expression that represents the language { ϵ }
 - 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

$$R_1 R_2$$
$$R_1 \cdot R_2$$

- We can combine together existing regular expressions in four ways.
- If R_1 and R_2 are regular expressions, $R_1 R_2$ 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, $R_1 | R_2$ 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 . *unity operator*
- 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 λ and \emptyset are RE ($a, b, c, \dots, \lambda, \emptyset$)
- Union of two RE is also a RE, $[R_1, R_2 \rightarrow (R_1 + R_2)]$
- Concatenation and Closure(*) of two RE is also RE, $[R_1, R_2 \rightarrow (R_1 \circ R_2), R^*]$
- RE over Σ are precisely those obtained recursively by the application of above rules.
- Examples:
 - $\{0, 1, 2\} \rightarrow R = 0 + 1 + 2$
 - $\{\lambda, ab\} \rightarrow R = \lambda ab$
 - $\{\lambda, 0, 00, 000, 0000, \dots\} \rightarrow R = 0^*$
 - $\{1, 11, 111, 1111, \dots\} \rightarrow R = 1^+$

↑
no empty
string so 1^+

0 | 1 2 }
0 1 1 2 }
0 + 1 + 2



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 , empty string
 - \emptyset , null language
 - $(R_1 \cup R_2)$, 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 .
- $L(\epsilon) = \{\epsilon\}$; one string, with no symbols.
- $L(\emptyset) = \emptyset$; no strings.
- $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$
- $L(R_1 \circ R_2) = L(R_1) \circ L(R_2)$
- $L(R_1^*) = (L(R_1))^*$

*not
effecting
v*

$$((0 \cup 1)\epsilon)^* \cup 0 = (0 \cup 1)^* \cup 0 \Rightarrow \sum^* \cup 0$$

- Example: Expression $((0 \cup 1)\epsilon)^* \cup 0$ 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.

$$\sum^* | 111 | \sum^*$$



EXAMPLE

- Example: $((0 \cup 1) \epsilon)^* \cup 0$
- Abbreviations:
 - Sometimes omit parentheses, use precedence of operations: * highest, then \cup .
- Example: Abbreviate above as $(0 \cup 1) \epsilon)^* \cup 0$
- Example: $(0 \cup 1)^* 111 (0 \cup 1)^*$



- **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: $L = \text{strings over } \{ 0, 1 \} \text{ with odd number of } 1\text{s.}$**

- $0^* 1 0^* (0^* 1 0^* 1 0^*)^*$

- **Example: $L = \text{strings with substring } 01 \text{ or } 10.$**

- $(0 \cup 1)^* 01 (0 \cup 1)^* \cup (0 \cup 1)^* 10 (0 \cup 1)^*$
- Abbreviate (writing Σ for $(0 \cup 1)$): $\Sigma^* 01 \Sigma^* \cup \Sigma^* 10 \Sigma^*$



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



OPERATOR PRECEDENCE

- Regular expression operator precedence:

(R)

R*

R₁R₂

R₁ | R₂

- 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:
 - $\mathcal{L}(\epsilon) = \{\epsilon\}$
 - $\mathcal{L}() =$
 - $\mathcal{L}(a) = \{a\}$
 - $\mathcal{L}(R_1 R_2) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$
 - $\mathcal{L}(R_1 | R_2) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$
 - $\mathcal{L}(R^*) = \mathcal{L}(R)^*$
 - $\mathcal{L}((R)) = \mathcal{L}(R)$



DECODING REGULAR EXPRESSION

- Let $\Sigma = \{0, 1\}$
- Let $L = \{ w \in \Sigma^* \mid w \text{ contains } 00 \text{ as a substring} \}$

(0 | 1)*00(0 | 1)*

11011100101

0000

11111011110011111



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 $\{0,1\}$.

- $0^*10^* = \{ w \mid w \text{ contains a single } 1 \}$
- $\Sigma^*1\Sigma^* = \{ w \mid w \text{ contains at least one } 1 \}$
- $01 \cup 10 = \{01, 10\}$
- $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$.
- If we let R be any regular expression, we have the following identities.
 - $R \cup \emptyset = R$. , doesn't add anything
 - Adding the empty language to any other language will not change it.
 - $R_1 \circ \varepsilon = R$. , doesn't add anything
 - Joining the empty string to any string will not change it.
 - $R \cup \varepsilon$ may not equal R .
 - For example, if $R = 0$, then $L(R) = \{0\}$ but $L(R \cup \varepsilon) = \{0, \varepsilon\}$.
 - $R_1 \circ \varepsilon$ may not equal R .
 - For example, if $R = 0$, then $L(R) = \{0\}$ but $L(R \circ \emptyset) = \emptyset$



- 0^*10^* : Language will contain a single 1
- $\Sigma^*1\Sigma^*$: your language should have at least a single 1 OR substring of 1
- $\Sigma^*001\Sigma^*$: your language contains a substring 001
- $1^*(01^+)^*$: it should always be followed by a 1
- $(\Sigma\Sigma)^*$: it will be an even length
- $(\Sigma\Sigma\Sigma)^*$: multiple of 3
- $01 \cup 10 = \{01, 10\}$; language contains 2 strings "01" "10"
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$: can make your own assumptions, starts with 0
- $(0 \cup \epsilon)1^*$: $01^* \cup \epsilon 1^*$; $01^* \cup 1^*$: consists of only 1s, start with 0 & followed by any # of 1s
- $(0 \cup \epsilon)(1 \cup \epsilon)$: $01 \cup \epsilon$
- $1^*\emptyset$: R. Ø = Ø
- $\emptyset^* = \epsilon$



- $0^*10^* = \{w \mid w \text{ contains a single } 1\}$.
- $\Sigma^*1\Sigma^* = \{w \mid w \text{ has at least one } 1\}$.
- $\Sigma^*001\Sigma^* = \{w \mid w \text{ contains the string } 001 \text{ as a substring}\}$.
- $1^*(01^+)^* = \{w \mid \text{every } 0 \text{ in } w \text{ is followed by at least one } 1\}$.
- $(\Sigma\Sigma)^* = \{w \mid w \text{ is a string of even length}\}$
- $(\Sigma\Sigma\Sigma)^* = \{w \mid \text{the length of } w \text{ is a multiple of } 3\}$.
- $01 \cup 10 = \{01, 10\}$.
- $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$.
- $(0 \cup \varepsilon)1^* = 01^* \cup 1^* \rightarrow$ The expression $0 \cup \varepsilon$ describes the language $\{0, \varepsilon\}$, so the concatenation operation adds either 0 or ε before every string in 1^* .
- $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$.
- $1^*\emptyset = \emptyset \rightarrow$ Concatenating the empty set to any set yields the empty set.
- $\emptyset^* = \{\varepsilon\} \rightarrow$ 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	
$\rightarrow 01$	Language containing only '01'
$\rightarrow 01 + 1$	language containing '01', '1' $(01 + 1^*)$
$\rightarrow (01 + \epsilon)1$	
$(0 + 10)^*(\epsilon + 1)$	

$$(0 \cup 10)^* (\epsilon + 1)$$



Some more examples of REs and their corresponding languages.

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



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 ε symbol.





EQUIVALENCE WITH FA

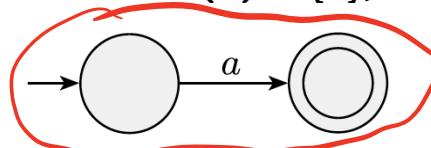
EQUIVALENCE WITH FINITE AUTOMATA

- 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 $a \in \Sigma$. Then $L(R) = \{a\}$, the NFA recognizes $L(R)$.

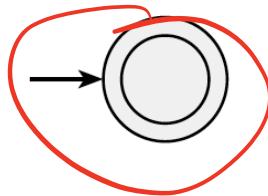


- Formally, $N = \{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\}$, Accepts only a.

where we describe δ by saying that $\delta(q_1, a) = \{q_2\}$ and that $\delta(r, b) = \emptyset$ for $r \neq q_1$ or $b \neq a$.

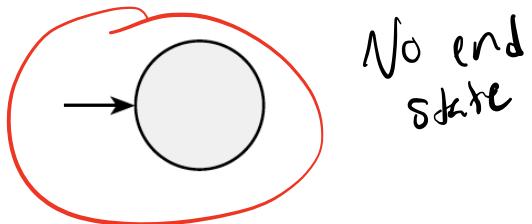


2. $R = \varepsilon$. Then $L(R) = \{\varepsilon\}$, and the following NFA recognizes $L(R)$.



- Formally, $N = \{q_1\}, \Sigma, \delta, q_1, \{q_1\}$, where $\delta(r, b) = \emptyset$ for any r and b .

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



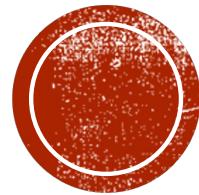
- Formally, $N = \{\{q\}, \Sigma, \delta, q, \emptyset\}$ where $\delta(r, b) = \emptyset$ for any r and b .

4. $R = R_1 \cup R_2$.

5. $R = R_1 \circ R_2$.

6. $R = R^*$

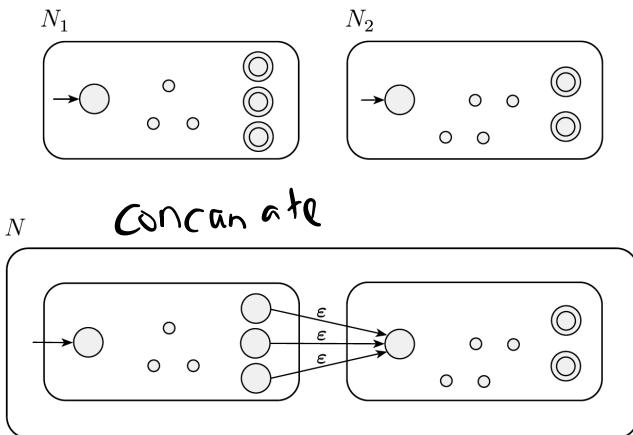
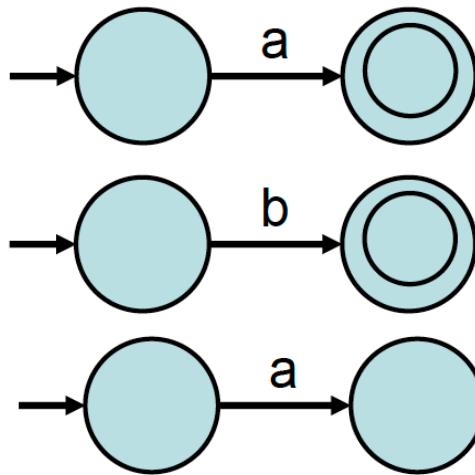


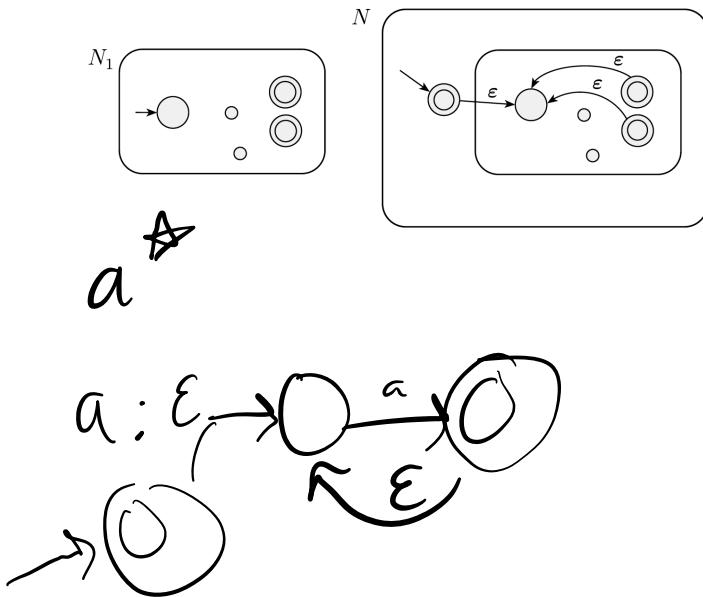
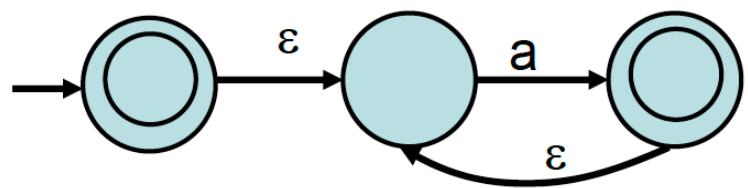
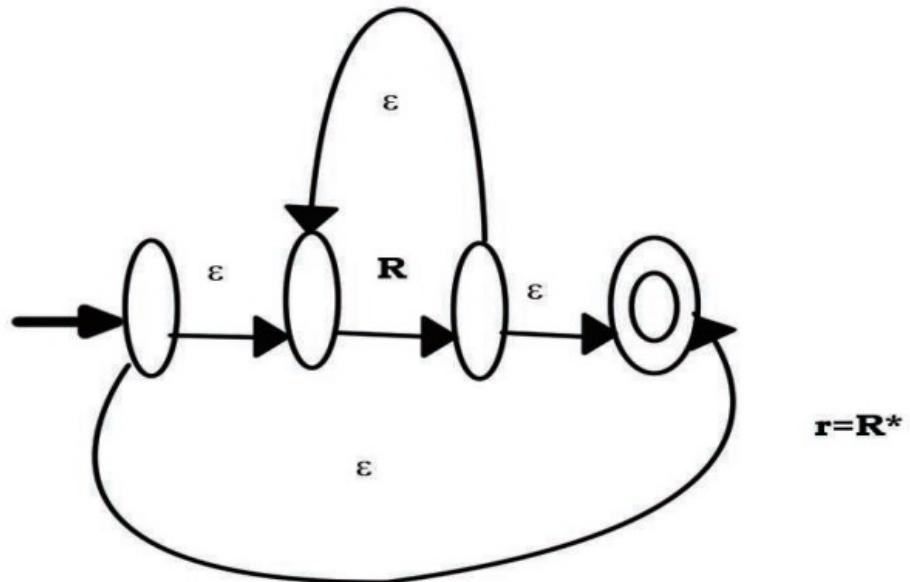


BUILDING AN NFA FROM THE REGULAR EXPRESSION

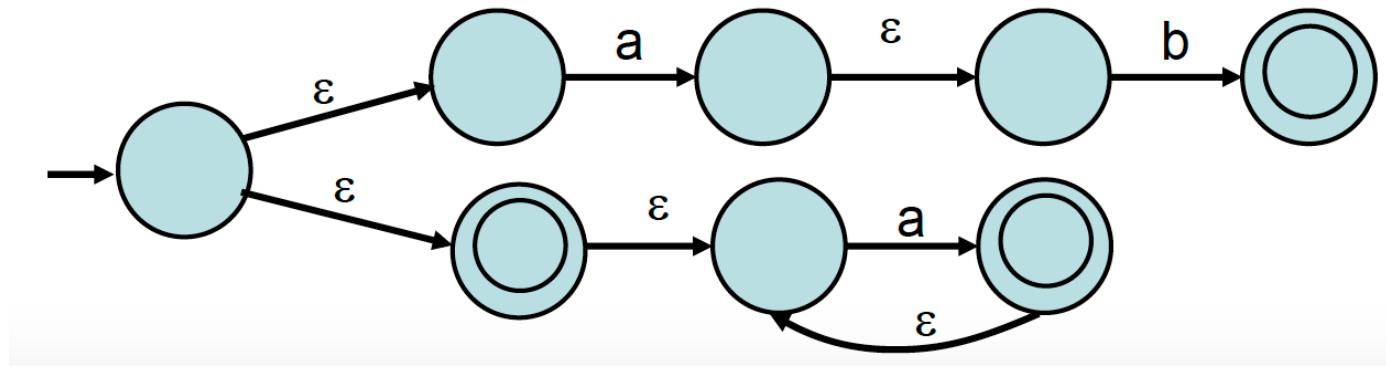
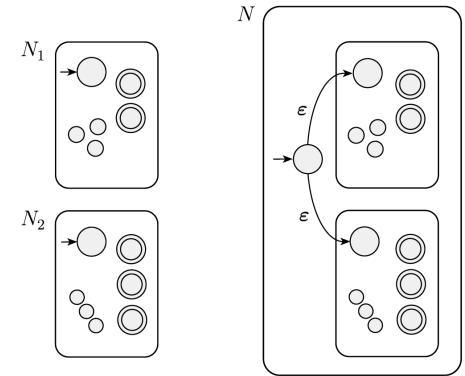
BASE CASES

- a,
- b,
- ab,

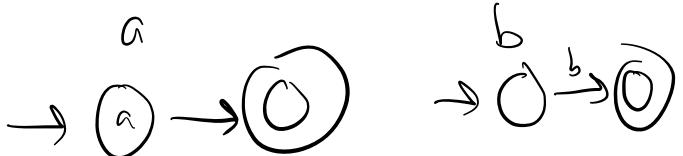




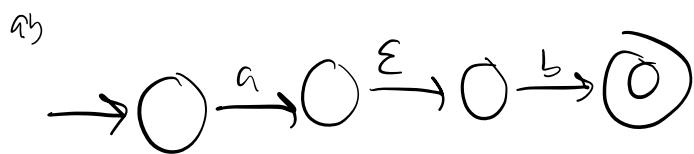
$$L = AB \cup A^*$$



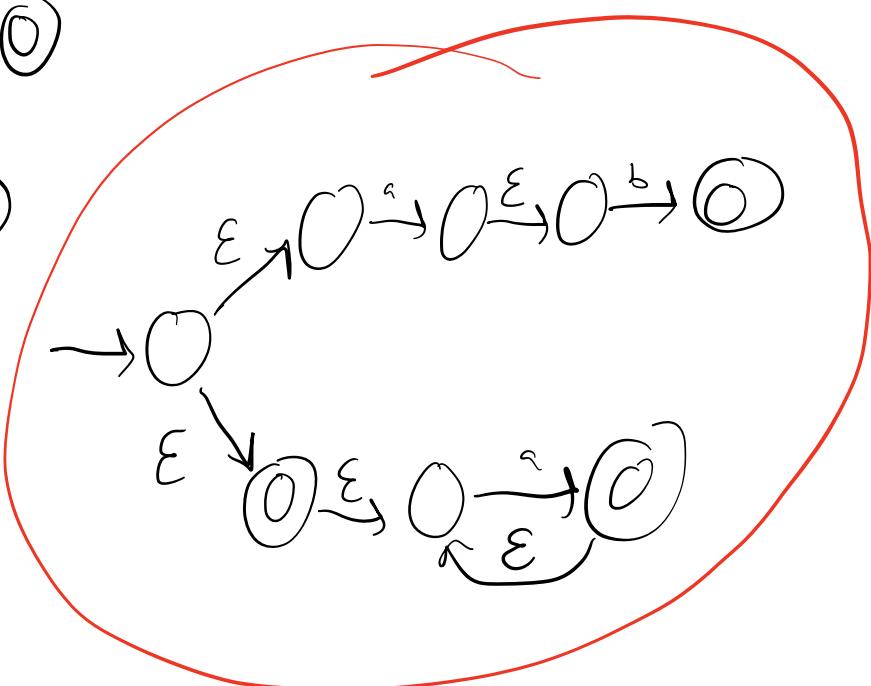
$A \cup A^*$



ab



A diagram illustrating a function mapping between two sets. It consists of two circles representing sets, with a curved arrow labeled ϵ pointing from the left circle to the right one.

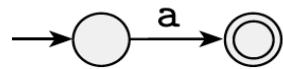


(AB \cup A)*

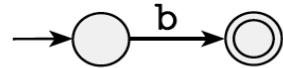


$$(AB \cup A)^*$$

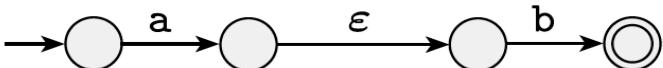
a



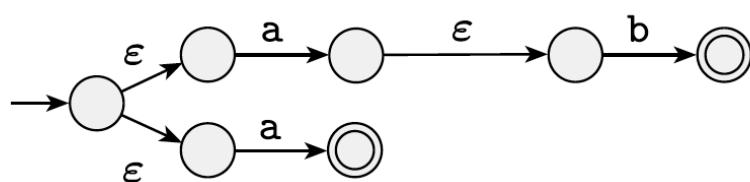
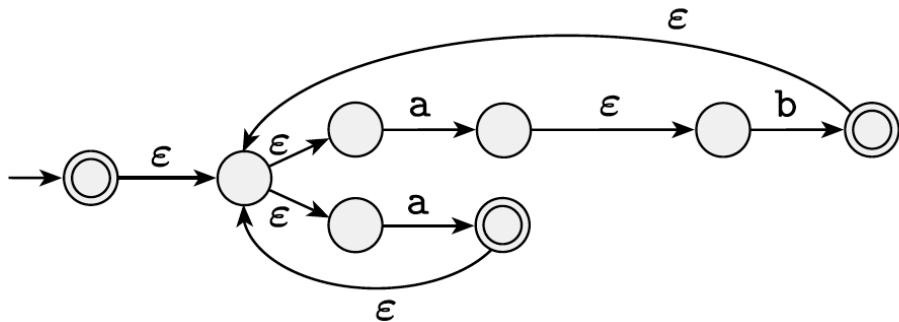
b



ab



ab ∪ a

 $(ab \cup a)^*$ 

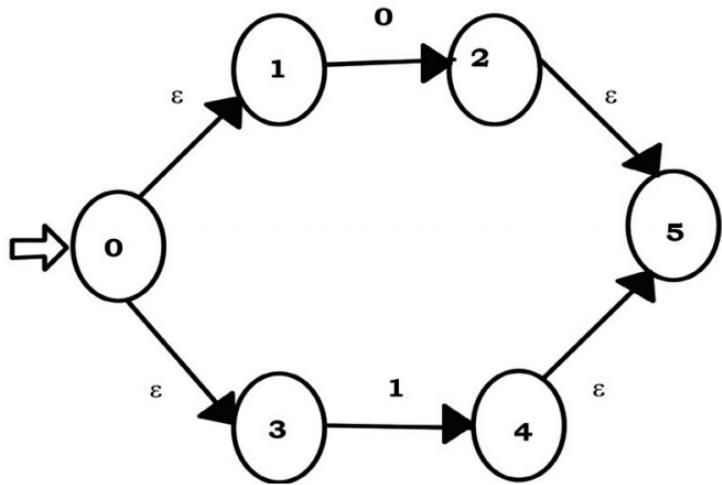
WHY?

- 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!

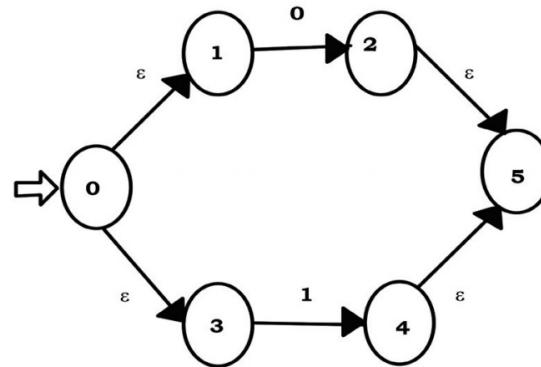


BASIC CONVERSIONS

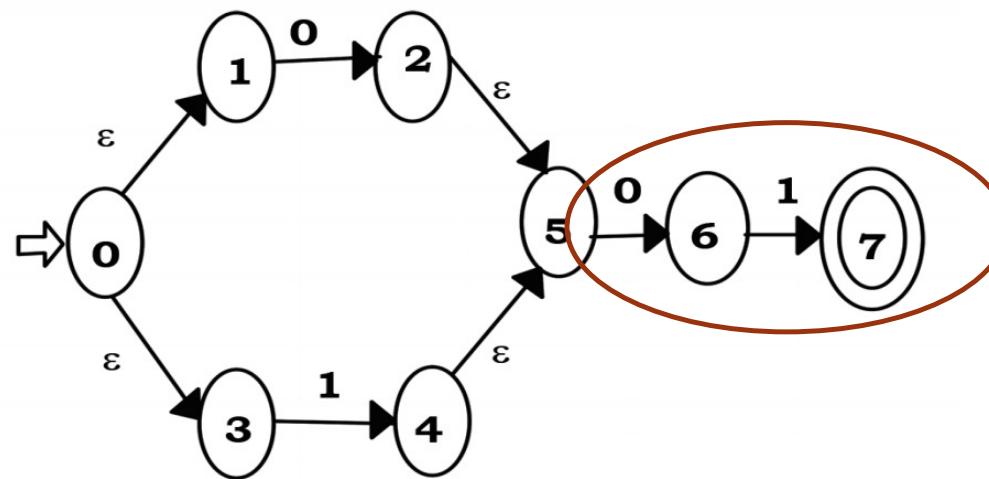
- 0 + 1



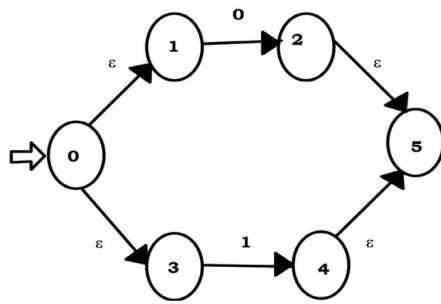
- $(0+1)01$
- $(0+1)$



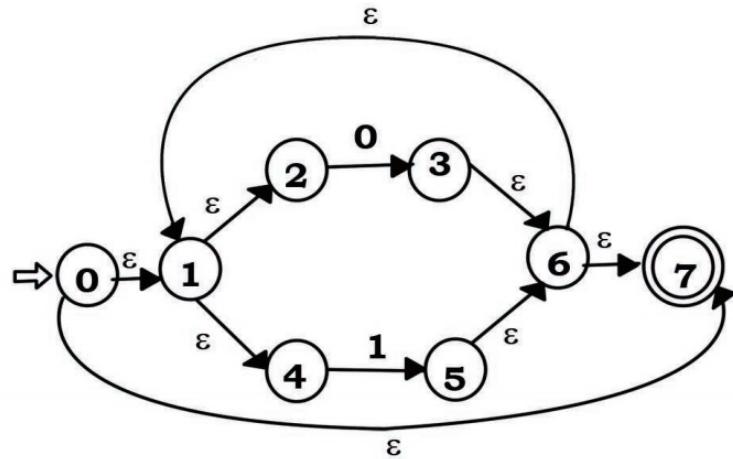
- $(0+1)01$



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



- $(0+1)^*$

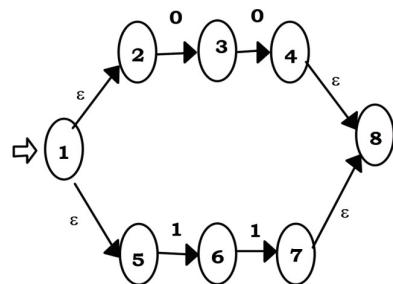


(00+11)*

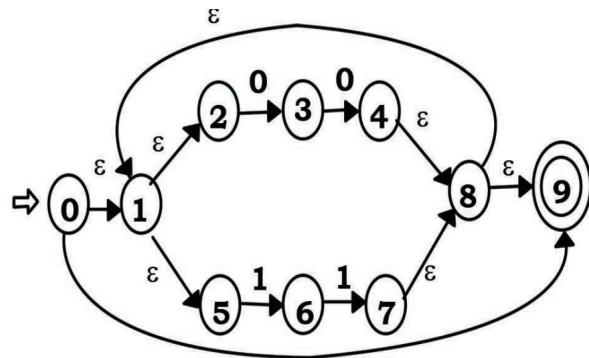


$$(00+11)^*$$

- Step 1: $(00+11)$

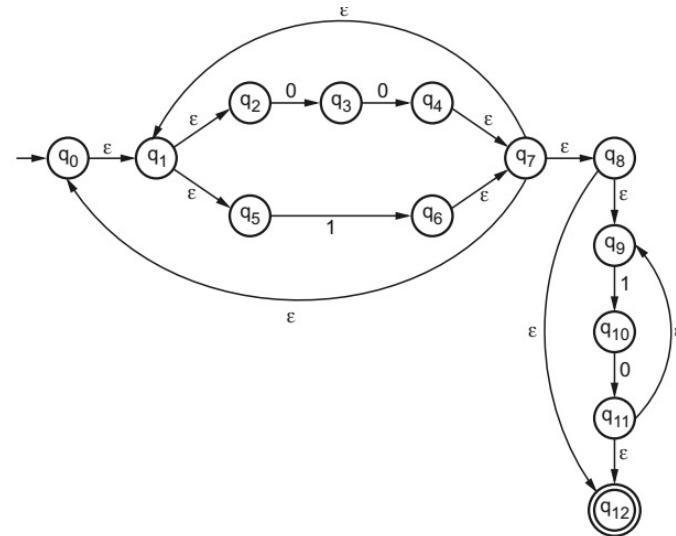


- Step 2: $(00+11)^*$



(00 + 1)* (10)*

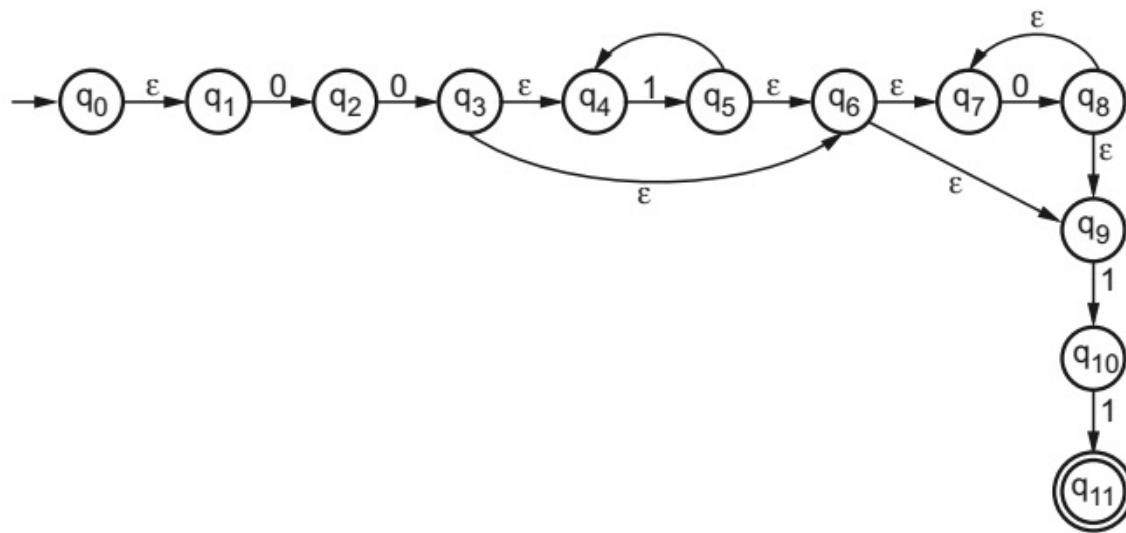


$$(00 + 1)^* \ (10)^*$$


001*0*11



001*0*11



TRY YOURSELF!

- $(a \cup b)^*aba$
- $(000)^* 1 \cup (00)^*1$
- $00(0 \cup 1)^*$
- $(ab \cup c)^*b$
- $(0+1)^* (00+11) (0+1)^*$
- $(ab+c^*)^*b$
- $a(abb)^* \cup b$
- $a(bc)^*$

