

Finite Automata 04

Thumrongsak Kosiyatrakul
tkosiyat@cs.pitt.edu

Regular Expressions

- A **regular expression** can be constructed using the following rules:
 - 1 a is a regular expression for some a in the alphabet Σ ,
 - 2 ε is a regular expression
 - 3 \emptyset is a regular expression
 - 4 If R_1 and R_2 are regular expressions, $R_1 \cup R_2$ is a regular expression
 - 5 If R_1 and R_2 are regular expressions, $R_1 \circ R_2$ or $R_1 R_2$ is a regular expression
 - 6 If R is a regular expression, R^* is a regular expression
- Note that to use rules 4, 5, or 6, you need to have regular expressions R_1 , R_2 , or R first which can only be constructed from rules 1, 2, or 3
- This is a recursive definition
 - Once you get regular expressions, you can apply rules 4, 5, or 6 multiple time to obtain more and more regular expressions

Regular Expression Examples

- Suppose $\Sigma = \{0, 1\}$
 - From rule 1, we have:
 - 0 is a regular expression
 - 1 is a regular expression
 - From rule 2, we have:
 - ε is a regular expression
 - From rule 3, we have:
 - \emptyset is a regular expression
- So far, we have four regular expressions, 0, 1, ε , and \emptyset
 - From rule 4, $0 \cup 1$ is a regular expression
 - From rule 5, 11 is a regular expression
 - From rule 6, 0^* is a regular expression
- We can keep building larger and larger regular expressions by applying rules 4, 5, and 6 multiple times

Regular Expression Examples

- Suppose $\Sigma = \{0, 1\}$
 - From rule 1, 0 and 1 are regular expressions
 - From rule 4, $0 \cup 1$ is a regular expression
 - From rule 6, $(0 \cup 1)^*$ is a regular expression
 - From rule 5, 01 is a regular expression
 - From rule 5, 011 is a regular expression
 - From rule 5, $(0 \cup 1)^*011$ is a regular expression
 - From rule 5, $(0 \cup 1)^*011(0 \cup 1)^*$ is a regular expression

Regular Expressions

- In arithmetic, an arithmetic expression can be used to represent its object (number)
 - $5 + 12$ is an arithmetic expression
 - $5 + 12$ can be used to represent 17
 - We usually write $5 + 12 = 17$
- In theory of computation, a regular expression can be used to express a language
 - Let $\Sigma = \{0, 1\}$
 - The regular expression $0 = \{0\}$
 - The regular expression $1 = \{1\}$
 - The regular expression $\varepsilon = \{\varepsilon\}$
 - The regular expression $\emptyset = \{ \}$
 - Suppose regular expressions $R_1 = A$ and $R_2 = B$ for languages A and B , the regular expression $R_1 \cup R_2 = A \cup B$
 - Suppose regular expressions $R_1 = A$ and $R_2 = B$ for languages A and B , the regular expression $R_1 R_2 = AB$
 - Suppose the regular expression $R = A$ for a language A , $R^* = A^*$

Regular Expressions (Other Notations)

- R^+ is a shorthand notation for RR^* for a regular expression R
 - All strings that are 1 or more concatenations of strings from R
 - Examples:
 - $1^+ = 11^*$
 - $(0 \cup 1)^+ = (0 \cup 1)(0 \cup 1)^*$
- R^k is a shorthand notation for the concatenation of k R 's with each other
 - Examples:
 - $1^5 = 11111$
 - $(0 \cup 1)^3 = (0 \cup 1)(0 \cup 1)(0 \cup 1)$

Regular Expressions (By Examples)

- A regular expression expresses a language:
 - $0 \cup 1$
 - $0 \cup 1 = \{0\} \cup \{1\} = \{0, 1\}$
 - A language that consists of two strings, 0 and 1
 - 0^*
 - $0^* = \{0\}^*$
 - Recall that $A^* = \{x_1x_2 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$
 - $\{0\}^* = \{\varepsilon, 0, 00, 000, 0000, 00000, \dots\}$
 - 0^* expresses the language that consists of all string that contains nothing but 0s including the empty string
 - $(0 \cup 1)0^*$
 - $(0 \cup 1)0^* = \{0, 1\} \circ 0^* = \{0, 1\} \circ \{\varepsilon, 0, 00, 000, \dots\}$
 - Recall that $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
 - $\{0, 1\}0^* = \{0, 1, 00, 10, 000, 100, 0000, 1000, \dots\}$

Regular Expressions (By Examples)

- A regular expression expresses a language:
 - $(0 \cup 1)^*$
 - $(0 \cup 1)^* = \{0, 1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}$
 - This is the set of all strings over 0 and 1
 - We generally use Σ^* instead of $(0 \cup 1)^*$
 - 0^*10^*
 - $0^*10^* = \{0\}^* \circ \{1\} \circ \{0\}^*$
 - $\{0\}^*$ is a language consisting of all strings containing zero or more 0s ($\{\epsilon, 0, 00, 000, \dots\}$)
 - $0^*10^* = \{w \mid w \text{ contains a single } 1\}$
 - $\Sigma^*1\Sigma^*$
 - $\Sigma^*1\Sigma^* = \Sigma^* \circ \{1\} \circ \Sigma^*$
 - Σ^* is the language consisting of all strings over Σ
 - $\Sigma^*1\Sigma^* = \{w \mid w \text{ has at least one } 1\}$
 - $\Sigma^*001\Sigma^*$
 - $\Sigma^*001\Sigma^* = \Sigma^* \circ \{001\} \circ \Sigma^*$
 - $\Sigma^*001\Sigma^* = \{w \mid w \text{ contains the string } 001 \text{ as a substring}\}$

Regular Expressions (By Examples)

- A regular expression expresses a language:

- $1^*(01^+)^*$

- $1^*(01^+)^* = \{1\}^* \circ (01^+)^*$
- $\{1\}^*$ is a language consisting of all strings containing zero or more 1s ($\{\varepsilon, 1, 11, 111, \dots\}$)
- $01^+ = \{0\} \circ \{1\}^+ = \{0\} \circ (\{1\} \circ \{1\}^*) = \{01, 011, 0111, \dots\}$
- $(01^+)^* = \{01, 011, 0111, \dots\}^* = \{\varepsilon, 01, 0101, 01011, 01101, 011011, \dots\}$
- $1^*(01^+)^* = \{w \mid \text{every 0 in } w \text{ is followed by at least one 1}\}$

- $(\Sigma\Sigma)^*$

- $\Sigma\Sigma = \Sigma \circ \Sigma = \{0, 1\} \circ \{0, 1\} = \{00, 01, 10, 11\}$
- $(\Sigma\Sigma)^* = \{00, 01, 10, 11\}^*$
- $(\Sigma\Sigma)^* = \{w \mid w \text{ is a string of even length}\}$

- $(\Sigma\Sigma\Sigma)^*$

- $\Sigma\Sigma\Sigma = \{0, 1\} \circ \{0, 1\} \circ \{0, 1\} = \{000, 001, 010, 011, 100, 101, 110, 111\}$
- $(\Sigma\Sigma\Sigma)^* = \{000, 001, 010, 011, 100, 101, 110, 111\}^*$
- $(\Sigma\Sigma\Sigma)^* = \{w \mid \text{the length of } w \text{ is a multiple of 3}\}$

Regular Expressions (By Examples)

- A regular expression expresses a language:
 - $01 \cup 10$
 - $01 \cup 10 = (\{0\} \circ \{1\}) \cup (\{1\} \circ \{0\}) = \{01\} \cup \{10\} = \{01, 10\}$
 - $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$
 - $0\Sigma^*0 = \{0\} \circ \Sigma^* \circ \{0\}$ is a set of all string that start and end with 0
 - $1\Sigma^*1$ is a set of all string that start and end with 1
 - $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$
 - $(0 \cup \varepsilon)1^*$
 - $0 \cup \varepsilon = \{0\} \cup \{\varepsilon\} = \{0, \varepsilon\}$
 - $1^* = \{\varepsilon, 1, 11, 111, \dots\}$
 - $(0 \cup \varepsilon)1^* = \{0, \varepsilon\} \circ \{\varepsilon, 1, 11, 111, \dots\}$
 - $(0 \cup \varepsilon)1^* = \{\varepsilon, 0, 01, 011, \dots, 1, 11, 111, \dots\}$
 - $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$
 - $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{0, \varepsilon\} \circ \{1, \varepsilon\} = \{\varepsilon, 0, 1, 01\}$
 - $1^*\emptyset = \{1\}^* \circ \emptyset = \emptyset$
 - $\emptyset^* = \{\varepsilon\}$

Operator Precedences and Identities

- Operator Precedences:
 - $*$ has the highest precedence.
 - \cup has the lowest precedence.
- Some properties:
 - $R \cup \emptyset = R$
 - Adding the empty language to any other language will not change it
 - $R \circ \varepsilon = R$
 - Joining the empty string to any string will not change it
 - $R \cup \{\varepsilon\} \neq R$
 - If $R = \{0, 1\}$, $R \cup \{\varepsilon\} = \{0, 1, \varepsilon\} \neq R$
 - $R \circ \emptyset \neq R$
 - If $R = \{0, 1\}$, $R \circ \emptyset = \{0, 1\} \circ \emptyset$
 - $\{0, 1\} \circ = \{xy \mid x \in \{0, 1\} \text{ and } y \in \emptyset\} = \emptyset$

Example: Regular Expression

- **Problem:** Write a program to evaluate a string representing a floating-point number
 - $72 \rightsquigarrow \text{true}$
 - $35.9 \rightsquigarrow \text{true}$
 - $20\ 67 \rightsquigarrow \text{false}$
 - $+7. \rightsquigarrow \text{true}$
 - $71a3 \rightsquigarrow \text{false}$
 - $-.29 \rightsquigarrow \text{true}$
- **Solution**
 - Create a DFA that recognize a set of all strings that are valid floating-point numbers and turn it into a program
 - It would be great if there is an easy way to do this
 - For now, can we create a regular expression that express the set of all valid floating-point representations?

$$(+ \cup - \cup \varepsilon)(D^+ \cup D^+.D^* \cup D^*.D^+)$$

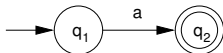
where $D = \{0, 1, 2, \dots, 9\}$

Regular Expression and Finite Automata

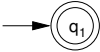
- Now we know that a regular expression can be used to express a language
- **Question:** Is language expressed by a regular expression regular?
 - Let's try to prove that a regular expression expresses a regular language
- Recall a regular expression:
 - A regular expression is defined **recursively** (those 6 rules)
 - So, we need to show that a regular expressions constructed from those rules expresses a regular language


Lemma 1.55

- From rule 1: a for some a in the alphabet Σ is a regular expression
 - We need to show that a language expressed by a regular expression generated from this rule is regular
 - Given a symbol $a \in \Sigma$, according to this rule, a is a regular expression
 - a expresses the language $\{a\}$
 - Is $\{a\}$ regular?
 - Can we construct a DFA that recognizes the language $\{a\}$?
 - This is an NFA that recognizes $\{a\}$



- For every NFA, there is an equivalent DFA
 - Thus $\{a\}$ is a regular language
 - Therefore, a expresses a regular language
- A regular expression constructed by rule 1 expresses a regular language

- From rule 2: ε is a regular expression
 - Again, we need to show that a language expressed by ε is regular
 - Recall that ε expresses the language $\{\varepsilon\}$
 - Is $\{\varepsilon\}$ regular?
 - Can we construct a DFA that recognizes the language $\{\varepsilon\}$?
 - This is an NFA that recognizes $\{\varepsilon\}$ 
 - For every NFA, there is an equivalent DFA
 - Thus, $\{\varepsilon\}$ is a regular language
 - Therefore, ε expresses a regular language
- A regular expression constructed by rule 2 expresses a regular language

- From rule 3: \emptyset is a regular expression
 - Again, we need to show that a language expressed by \emptyset is regular
 - Recall that \emptyset expresses the language $\{ \}$
 - Is $\{ \}$ regular?
 - Can we construct a DFA that recognizes the language $\{ \}$?
 - This is an NFA that recognizes $\{ \}$ 

```
graph LR; start(( )) --> q((q));
```
 - For every NFA, there is an equivalent DFA
 - Thus, \emptyset is a regular language
 - Therefore, \emptyset expresses a regular language
- A regular expression constructed by rule 3 expresses a regular language

Lemma 1.55

- Recall that you cannot use rules 4, 5, and 6 to create a new regular expressions unless you already have some regular expressions created by rules 1, 2, or 3
- We already prove that each regular expression created by rules 1, 2, or 3 expresses a regular language
- For rule 4:
 - R_1 and R_2 are regular expressions that express regular languages (from rules 1, 2, or 3)
 - Recall that $R_1 = A$ and $R_2 = B$ for some languages A and B ,
 $R_1 \cup R_2 = A \cup B$
 - We already prove that if A and B are regular languages, $A \cup B$ is regular (regular language is closed under union operation)
 - Thus, a regular expression obtain by rule 4 express a regular language
- Same for rules 5 and 6
 - Regular language is closed under concatenation and star operations

Lemma 1.55

If a language is described by a regular expression, then it is regular.

- We just proved the above lemma
- By proving the above lemma, it gives us a tool to construct an NFA that recognizes the language expressed by a regular expression
 - Based on how we prove that regular language is closed under union, concatenation, and star operations

Regular Expressions to NFA

- Suppose we want to construct an NFA that recognizes the language expressed by the regular expression $(ab \cup a)^*$ where $\Sigma = \{a, b\}$
- We need to apply those 6 rules to obtain the regular expression
 - This will be a guideline step-by-step to construct an NFA
- Here are steps to construct the regular expression $(ab \cup a)^*$
 - 1 a is a regular expression (rule 1)
 - 2 b is a regular expression (rule 1)
 - 3 ab is a regular expression (rule 5 with (1) and (2))
 - 4 $ab \cup a$ is a regular expression (rule 4 with (3) and (1))
 - 5 $(ab \cup a)^*$ is a regular expression (rule 6 with (4))

Regular Expression to NFA

- Convert the regular expression $(ab \cup a)^*$ to an NFA.

⌈

a

b

ab

ab U a

$(ab \cup a)^*$

⌋

Regular Expression to NFA

- Convert the regular expression $(ab \cup a)^*$ to an NFA.



a

b

ab

$ab \cup a$

$(ab \cup a)^*$

」

Regular Expression to NFA

- Convert the regular expression $(ab \cup a)^*$ to an NFA.



ab

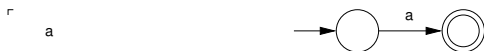
$ab \cup a$

$(ab \cup a)^*$

┘

Regular Expression to NFA

- Convert the regular expression $(ab \cup a)^*$ to an NFA.

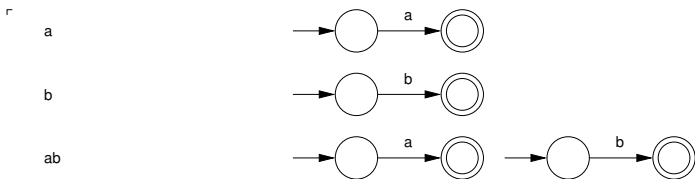


$ab \cup a$

$(ab \cup a)^*$

Regular Expression to NFA

- Convert the regular expression $(ab \cup a)^*$ to an NFA.



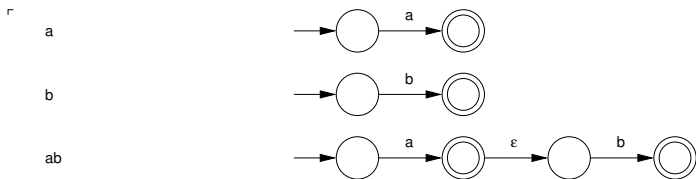
$ab \cup a$

$(ab \cup a)^*$

┘

Regular Expression to NFA

- Convert the regular expression $(ab \cup a)^*$ to an NFA.



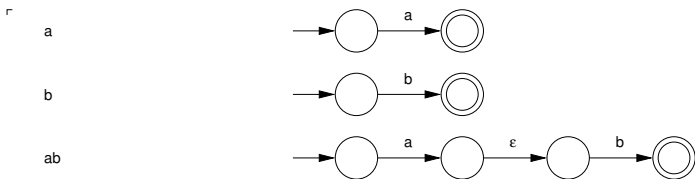
$ab \cup a$

$(ab \cup a)^*$

┘

Regular Expression to NFA

- Convert the regular expression $(ab \cup a)^*$ to an NFA.



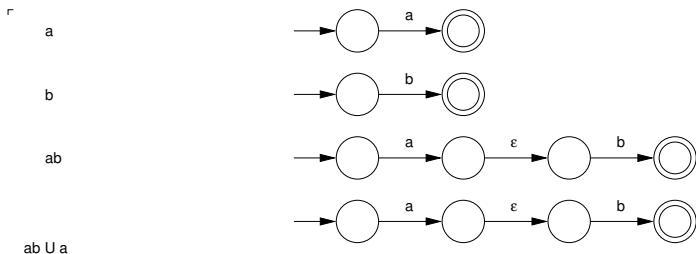
$ab \cup a$

$(ab \cup a)^*$

」

Regular Expression to NFA

- Convert the regular expression $(ab \cup a)^*$ to an NFA.

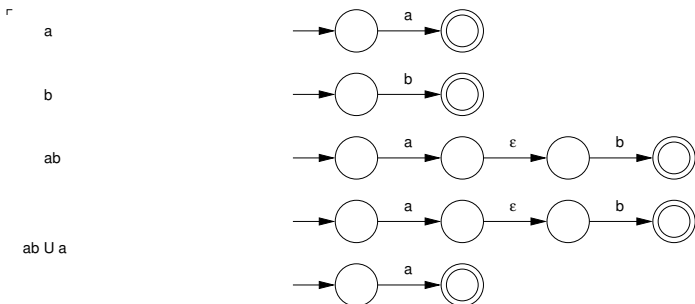


$(ab \cup a)^*$

┘

Regular Expression to NFA

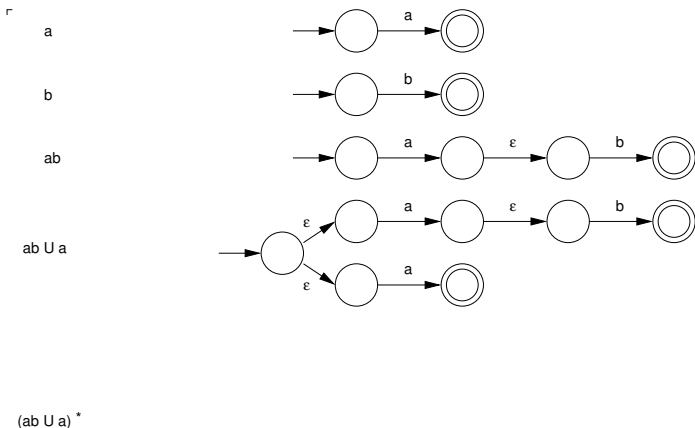
- Convert the regular expression $(ab \cup a)^*$ to an NFA.



$(ab \cup a)^*$

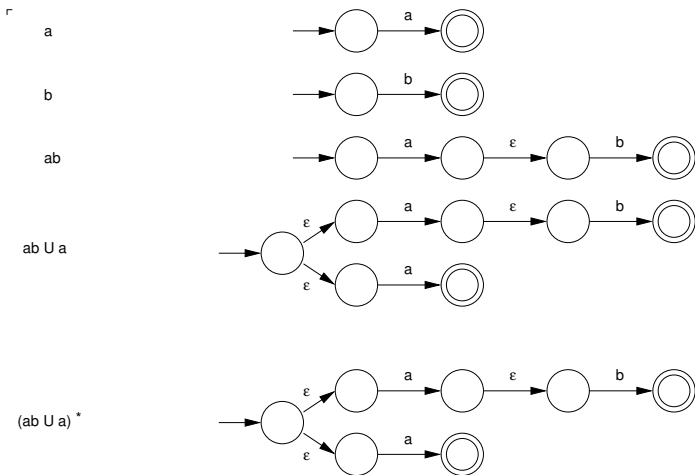
Regular Expression to NFA

- Convert the regular expression $(ab \cup a)^*$ to an NFA.



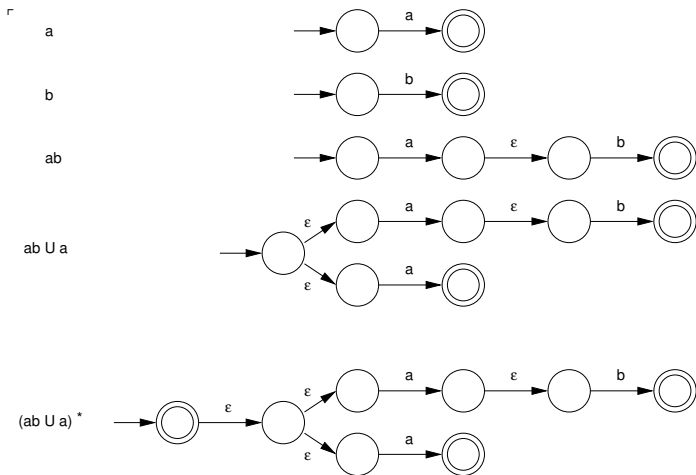
Regular Expression to NFA

- Convert the regular expression $(ab \cup a)^*$ to an NFA.



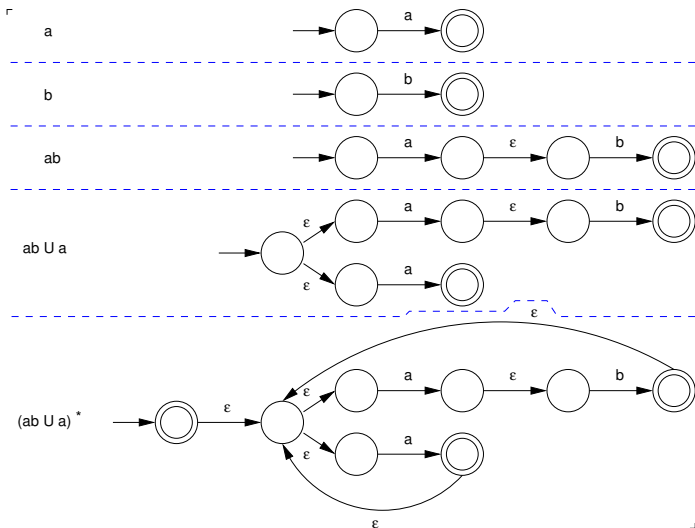
Regular Expression to NFA

- Convert the regular expression $(ab \cup a)^*$ to an NFA.



Regular Expression to NFA

- Convert the regular expression $(ab \cup a)^*$ to an NFA.



Regular Expressions to NFA

- Suppose we want to construct an NFA that recognizes the language expressed by the regular expression $(a \cup b)^*aba$ where $\Sigma = \{a, b\}$
- Again, we need to apply those 6 rules to get a guideline
- Here are steps to construct the regular expression $(a \cup b)^*aba$
 - 1 a is a regular expression (rule 1)
 - 2 b is a regular expression (rule 1)
 - 3 $a \cup b$ is a regular expression (rule 4 with (1) and (2))
 - 4 $(a \cup b)^*$ is a regular expression (rule 6 with (3))
 - 5 ab is a regular expression (rule 5 with (1) and (2))
 - 6 aba is a regular expression (rule 5 with (5) and (1))
 - 7 $(a \cup b)^*aba$ is a regular expression (rule 5 with (4) and (6))

Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.

r

a

b

$a \cup b$

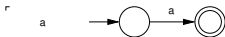
$(a \cup b)^*$

aba

$(a \cup b)^*aba$

Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.



b

$a \cup b$

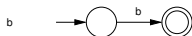
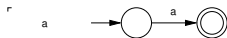
$(a \cup b)^*$

aba

$(a \cup b)^*aba$

Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.



$a \cup b$

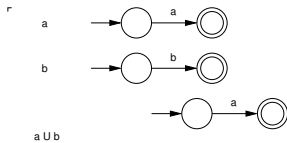
$(a \cup b)^*$

aba

$(a \cup b)^*aba$

Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.



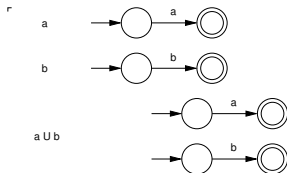
$(a \cup b)^*$

aba

$(a \cup b)^*aba$

Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.



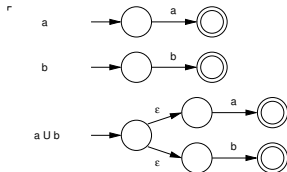
$(a \cup b)^*$

aba

$(a \cup b)^*aba$

Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.



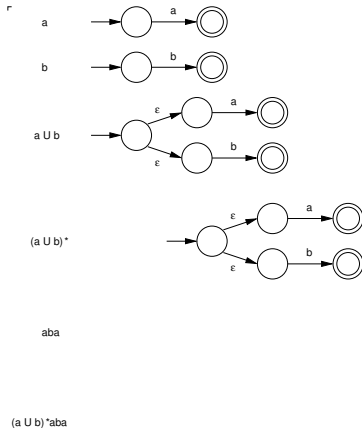
$(a \cup b)^*$

aba

$(a \cup b)^*aba$

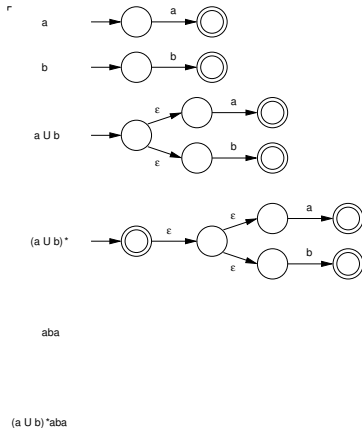
Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.



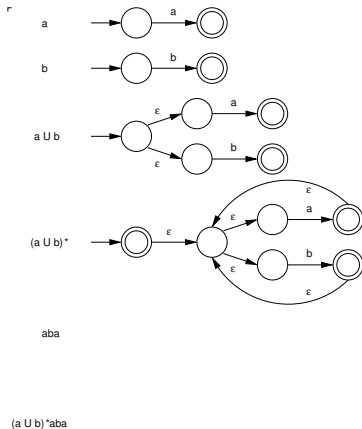
Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.



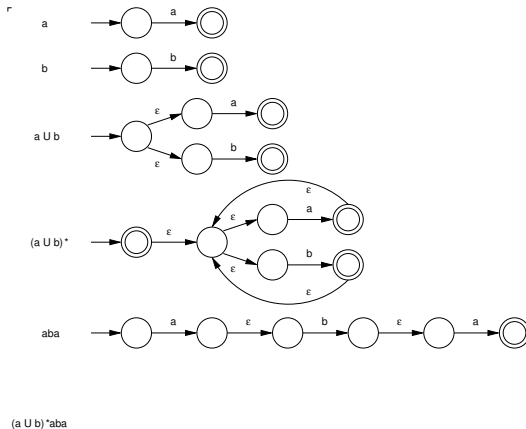
Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.



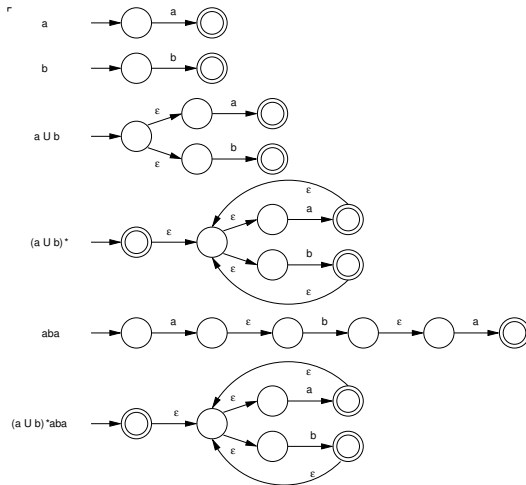
Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.



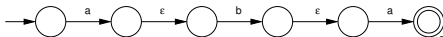
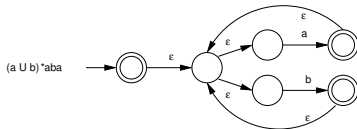
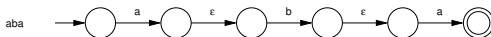
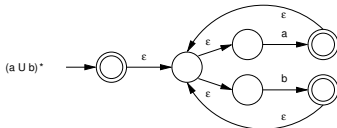
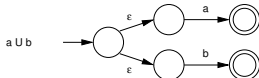
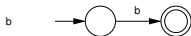
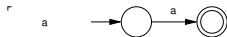
Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.



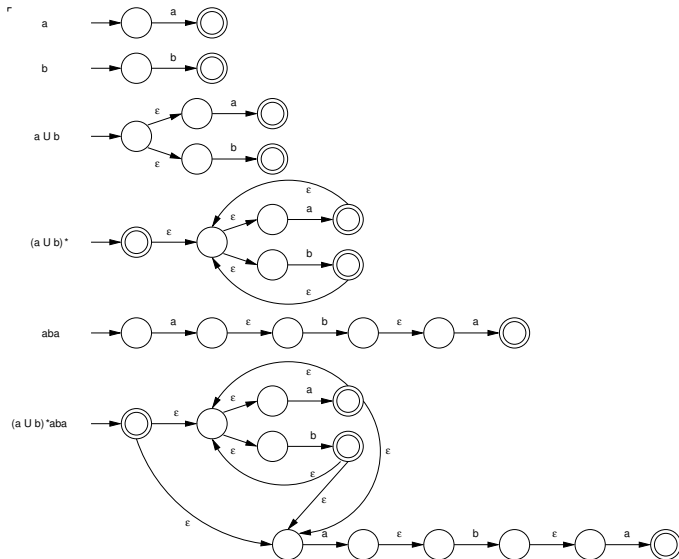
Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.



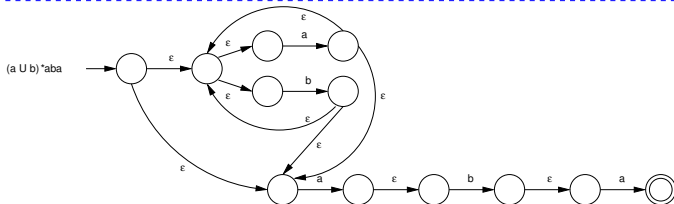
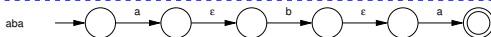
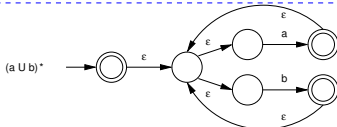
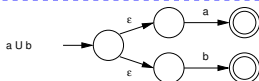
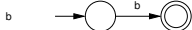
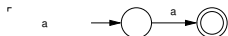
Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.



Regular Expression to NFA

- Convert the regular expression $(a \cup b)^*aba$ to an NFA.



- Recall that the set of all valid floating-point representation is expressed by a regular expression

$$(+ \cup - \cup \varepsilon)(D^+ \cup D^+.D^* \cup D^*.D^+)$$

where

- $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Now, we can easily convert the above regular expression into an NFA, construct an equivalent DFA, and turn the DFA into a program
 - This process is done by computer
 - Often used in compiler

Conclusion

- A regular expression expresses a regular language
- We have a tool to convert a regular expression into an equivalent NFA
- What is next?
 - We still do not know whether every regular language can be expressed by a regular expression