CS F351: Theory of Computation

04 – Regular Expressions

BITS Pilani
Dubai Campus

Dr Elakkiya R, AP/CS

Regular Expressions

- Motivation for Regular Expression
- Regular Expression: Definition, Properties, Egs.
- Rel Lang, Reg Expr, NFA, DFA Equivlance
- Reg Expr => NFA Construction
- Egs.

Formal Language, chapter 1, slide 5

How is a (Formal) Lang described?

- What is a (Formal) Language.
  - Set of strings over an alphabet with some property
  - $L = \{ w \in \Sigma^* \mid w \text{ has some property} \}$

Increasing level of difficulty

RL description	Method	Human	Computer
Verbal	Textual	Easy	Difficult (NLP)
Reg Expr	Algebraic	Slightly difficult	Diffi
NFA	Diagrammatic	Medium	Medium
DFA	Diagrammatic	Difficult	Easy

All of these are equivalent in terms of the Language described/recognized.
ie. All of them are equally powerful.
ie, L = L(R) = L(NFA) = L(DFA)

Regular Expression

- Regular Expressions provide an algebraic expression framework to describe the same class of strings (i.e those which can be recognized with finite memory)

- For every regular expression (R), there is a corresponding regular set or language L. i.e. L(R) = L

- Consider alphabet $\Sigma = \{0, 1\}$.

- Atomic Regular expression:

  0 means $\{0\}$; i.e. $R = 0$ corresponds to a regular language/set = $\{ 0 \}$
  1 means $\{1\}$; i.e. $R = 1$ corresponds to a regular language/set = $\{ 1 \}$

- Composite Regular expressions use above atomic regular expression and operations
  - union $\cup$
  - concatenation $\circ$ (usually not used explicitly)
  - Kleene star $*$

Interpreting Regular Expressions

Example: $0 \cup 1$ means $\{0\} \cup \{1\}$, which equals $\{0, 1\}$.

Example:

- Consider $(0 \cup 1)0^*$, which means $(0 \cup 1) \circ 0^*$.
- This equals $\{0, 1\} \circ \{0\}^*$.
- Recall $\{0\}^* = \{ \varepsilon, 0, 00, 000, \ldots \}$.
- Thus, $\{0, 1\} \circ \{0\}^*$ is the set of strings that start with symbol 0 or 1, and followed by zero or more 0's. = {0 , 1, 00, 10, 000, 100, 0000, 1000, …}

Example:

- $(0 \cup 1)^*$ means $(\{0\} \cup \{1\})^*$. $\Sigma = \{0, 1\}$.
- This equals $\{0, 1\}^*$,which is the set of all possible strings over the alphabet Σ .
- When $\Sigma = \{0, 1\}$,
  often use shorthand notation Σ to denote regular expression $(0 \cup 1)$.

Formal Definition of Regular Expression

Definition: R is a regular expression with alphabet Σ if R is

1. a for some $a \in \Sigma$
2. ϵ
3. $\emptyset$
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions
5. $(R_1) \circ (R_2)$, also denoted by $(R_1)(R_2)$, where R_1 and R_2 are regular expressions
6. $(R_1)^*$, where R_1 is a regular expression
7. (R_1), where R_1 is a regular expression.

Can remove redundant parentheses, e.g., $((0) \cup (1))(1) \dashrightarrow (0 \cup 1)1$.

Definition: If R is a regular expression, then $L(R)$ is the language generated (or described or defined) by R.

Language generated by Reg Expr

innovate achieve lead

- **Regular Expression** | **Regular Set/Language**

Regular Expression	Regular Set/Language
$\emptyset$	$\{\}$
ε	$\{\varepsilon\}$
a for $\in \Sigma$	$\{a\}$
$(R_1 \cup R_2)$	$L(R_1) \cup L(R_2)$
$(R_1 \circ R_2)$	$L(R_1) \circ L(R_2)$
(R^*)	$(L(R))^*$

Eg.
$\Sigma = \{a, b\}$
$L(\emptyset) = \{ \}$
$L(\varepsilon) = \{\varepsilon\}$
$L(a) = \{a\}$
$L((a+b)^*) = \{\varepsilon, a, b, aa, ab, \ldots\} = \Sigma^*$
$L(abba \cup \varepsilon) = \{abba, \varepsilon\}$
$L(\emptyset^*) = \{\varepsilon\}$

- Union is also denoted using $R_1 + R_2$.
- Precedence order: star, concatenation and then union.
- $R^+ = RR^*$ and R^k for k-fold concatenation

- *Eg*. Language described/generated by a regular expression $(a \cup b)^*b$

$$
\begin{aligned}
L((a \cup b)^*b) &= L((a \cup b)^*)\, L(b) \\
&= (L((a \cup b)))^*\, L(b) \\
&= (L(a) \cup L(b))^*\, L(b) \\
&= (\{a\} \cup \{b\})^*\, \{b\} \\
&= \{a, b\}^*\, \{b\}
\end{aligned}
$$

L = The set of all strings that end in b.
(Verbal description)

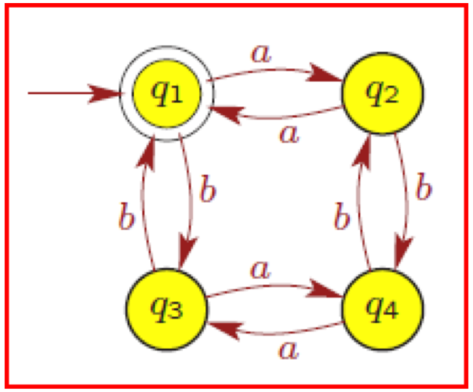
Examples of Regular Expressions

Examples: For $\Sigma = \{0, 1\}$,

1. $(0 \cup 1) = \{0, 1\}$
2. $0^*10^* = \{ w \mid w$ has exactly a single $1 \}$
3. $\Sigma^*1\Sigma^* = \{ w \mid w$ has at least one $1 \}$
4. $\Sigma^*001\Sigma^* = \{ w \mid w$ contains 001 as a substring $\}$
5. $(\Sigma\Sigma)^* = \{ w \mid |w|$ is even $\}$
6. $(\Sigma\Sigma\Sigma)^* = \{ w \mid |w|$ is a multiple of three $\}$
7. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{ w \mid w$ starts and ends with the same symbol $\}$
8. $1^*\emptyset = \emptyset$, anything concatenated with $\emptyset$ is equal to $\emptyset$.
9. $\emptyset^* = \{\varepsilon\}$

- Define EVEN-EVEN over alphabet $\Sigma = \{a, b\}$ as strings with an even number of a's and an even number of b's

  For example, $aababbaaababab \in$ EVEN-EVEN.



- Regular expression: $aa \cup bb \cup ((ab \cup ba)(aa \cup bb)^*(ab \cup ba))^*$

Regular Expr Identities

Examples:

1. $R \cup \emptyset = \emptyset \cup R = R$

2. $R \circ \varepsilon = \varepsilon \circ R = R$

3. $R \circ \emptyset = \emptyset \circ R = \emptyset$

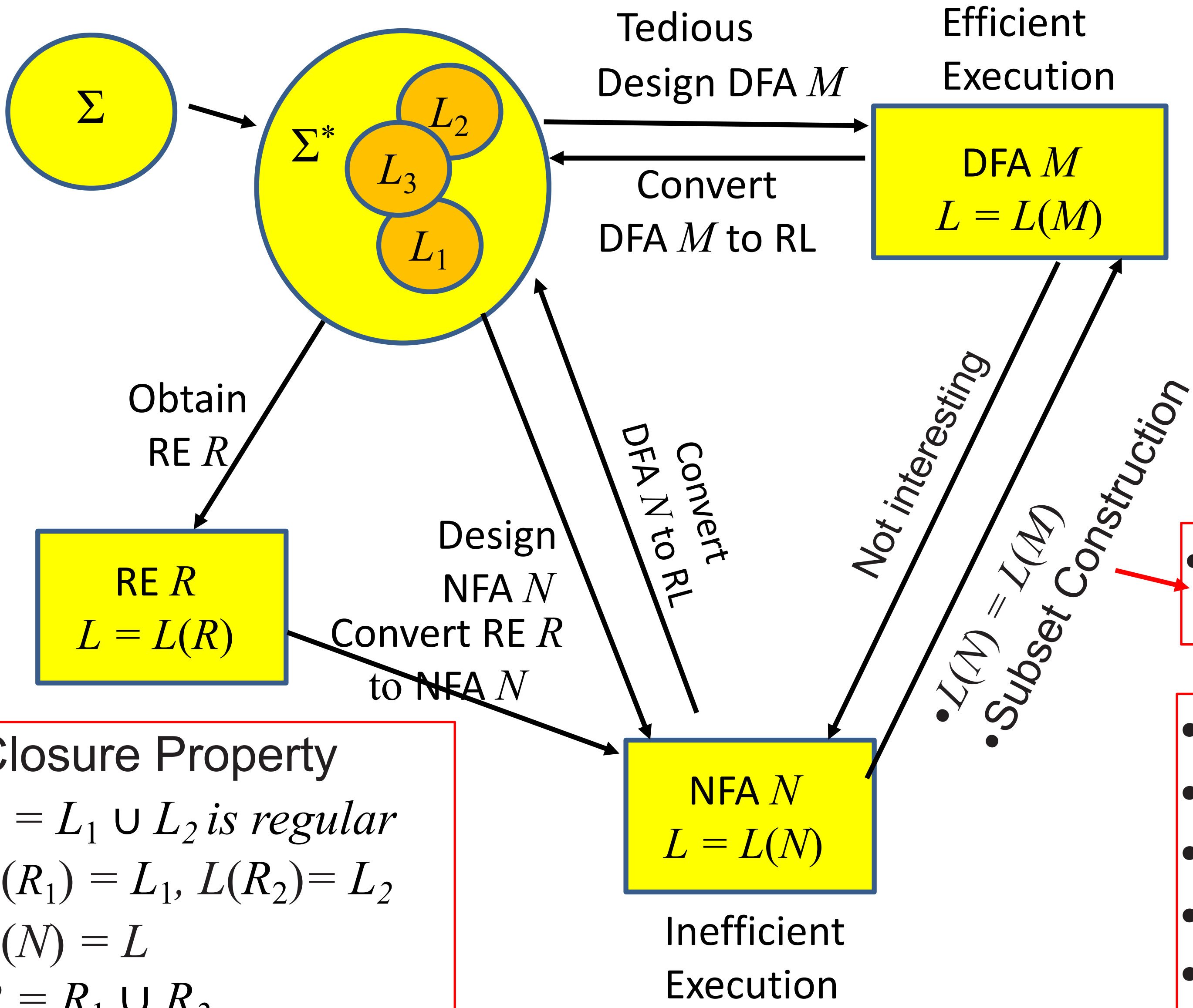
4. $R_1(R_2 \cup R_3) = R_1R_2 \cup R_1R_3$.
   Concatenation distributes over union.

Egs.

L = {w ∈{a, b}* | w contains an odd number of a's}

R = b* (ab*ab*)* ab*
or b* ab* (ab*ab*)*

RL ⇔ RE ⇔ NFA ⇔ DFA



- $L(M) = L$.

- Closure Property
- $L = L_1 \cup L_2$ *is regular*
- $L(M_1) = L_1$, $L(M_2) = L_2$
- $L(M) = L$
- $M = M_1 \times M_2$
- Tedious

- NFAs and DFAs are equivalent

- Closure Property
- $L = L_1 \cup L_2$ *is regular*
- $L(R_1) = L_1$, $L(R_2) = L_2$
- $L(N) = L$
- $R = R_1 \cup R_2$
- Easier

- Closure Property
- $L = L_1 \cup L_2$ *is regular*
- $L(N_1) = L_1$, $L(N_2) = L_2$
- $L(N) = L$
- $N = N_1 \cup N_2$
- Use ε transitions
- Easier

Regular Language <-> Regular Expression

Theorem:

Language L is regular iff L has a regular expression.

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

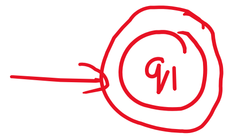
Proof. Procedure to convert regular expression R into NFA N :

1. If $R = a$ for some $a \in \Sigma$, then $L(R) = \{a\}$, which has NFA

$N = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$ where transition function δ

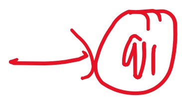
- $\delta(q_1, a) = \{q_2\}$,
- $\delta(r, b) = \emptyset$ for any state $r \neq q_1$ or any $b \in \Sigma_\varepsilon$ with $b \neq a$

2. $N = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$ where



- $\delta(r, b) = \emptyset$ for any state r and any $b \in \Sigma_\varepsilon$.

3. If $R = \emptyset$, then $L(R) = \emptyset$, which has NFA



$N = (\{q_1\}, \Sigma, \delta, q_1, \emptyset)$ where

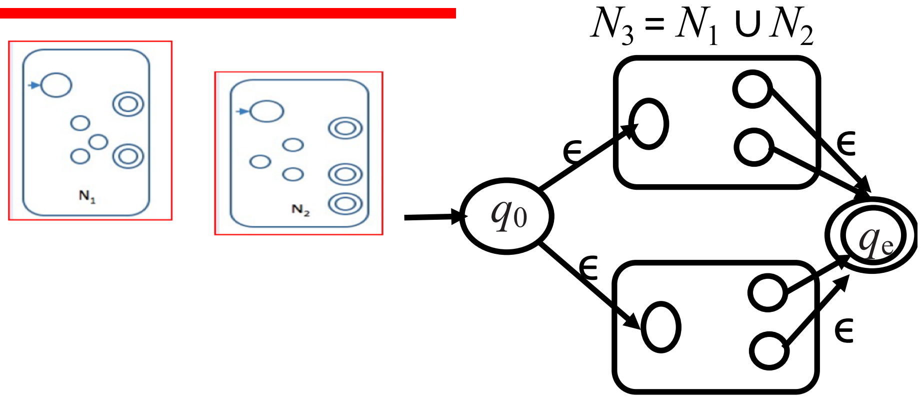
- $\delta(r, b) = \emptyset$ for any state r and any $b \in \Sigma_\varepsilon$.

RE to NFA

4.If $R = (R_1 \cup R_2)$ and

- $L(R_1)$ has NFA N_1, $L(R_2)$ has NFA N_2,

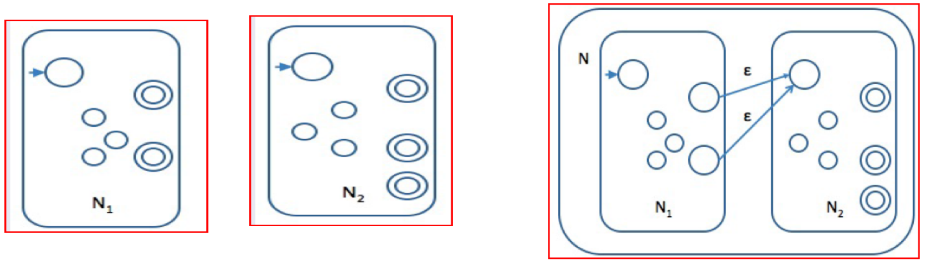
  then $L(R) = L(R_1) \cup L(R_2)$ has NFA N:



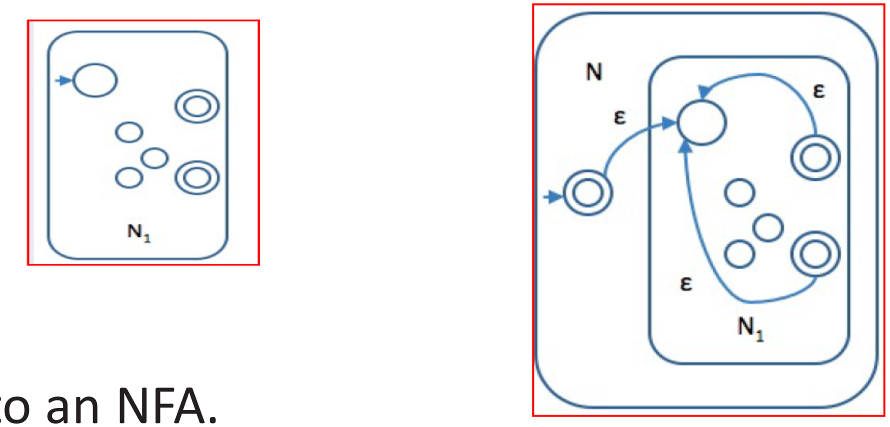
5. If $R = (R_1) \circ (R_2)$ and

- $L(R_1)$ has NFA N_1, $L(R_2)$ has NFA N_2,

  then $L(R) = L(R_1) \circ L(R_2)$ has NFA N :



6. If $R = (R_1)^*$ and $L(R_1)$ has NFA N_1,
   then $L(R) = (L(R_1))^*$ has NFA :

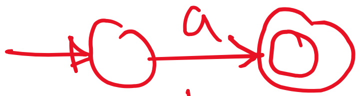


- Thus, can convert any regular expression R into an NFA.
- This implies that the language $L(R)$ is regular.

RE -> NFA Construction eg.

Ex: Build NFA for $(ab \cup a)^*$

a



Rule 1 (Slide 11)

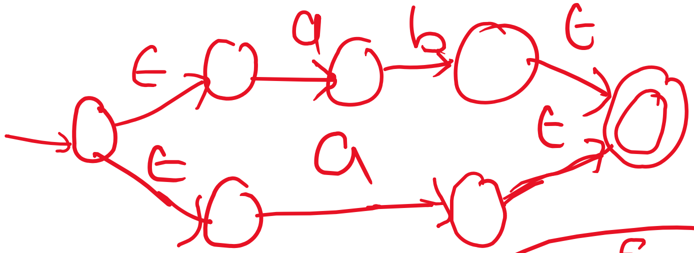
b

Rule 2: (Slide 11)

ab

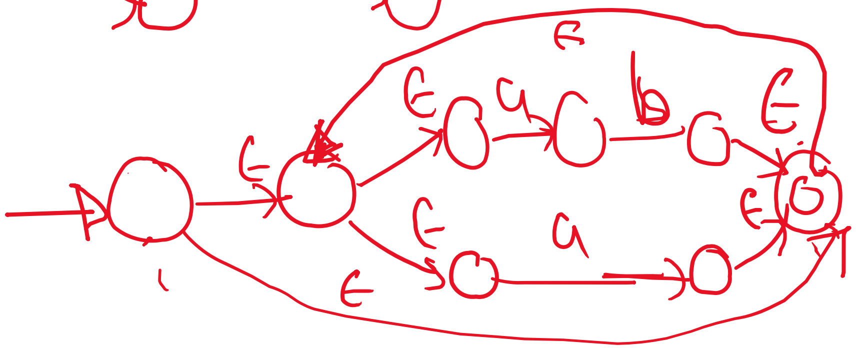
Rule 5: (Slide 12)

$ab \cup a$



Rule 4: (Slide 12)

$(ab \cup a)^*$

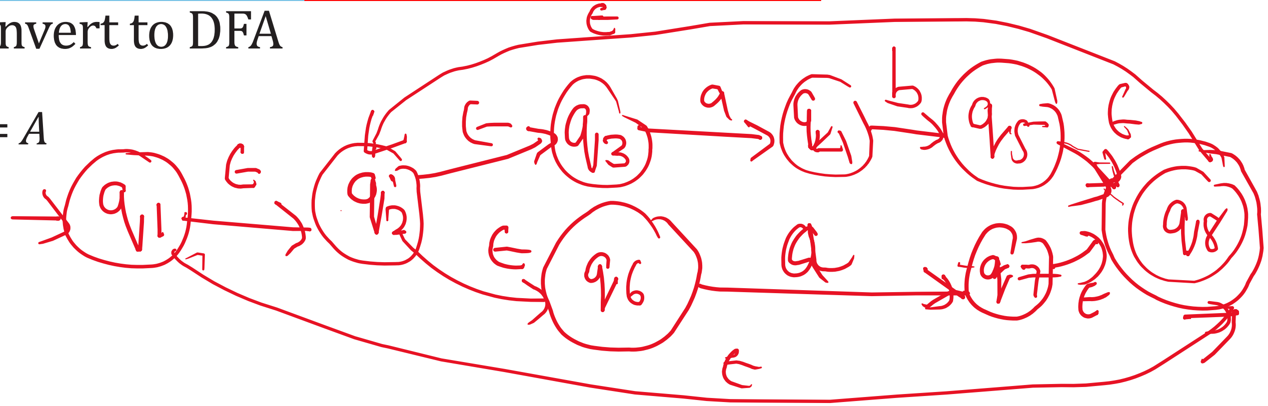


Rule 6: (Slide 12)

This NFA can be converted to equivalent DFA using subset construction method

RE -> NFA -> DFA Construction eg.

Ex: Build NFA for $(ab \cup a)^*$, Convert to DFA



$q'_0 = \varepsilon C(\{q_1\}) = \{q_1, q_2, q_3, q_6, q_8\} = A$

$\delta'(A, a) = \varepsilon C(\delta(\{q_1, q_2, q_3, q_6, q_8\}, a))$
$= \varepsilon C(\{q_4, q_7\})$
$= \{q_4, q_7, q_8, q_2, q_3, q_6\} = B$

$\delta'(A, b) = \varepsilon C(\delta(\{q_1, q_2, q_3, q_6, q_8\}, b)) = \emptyset$

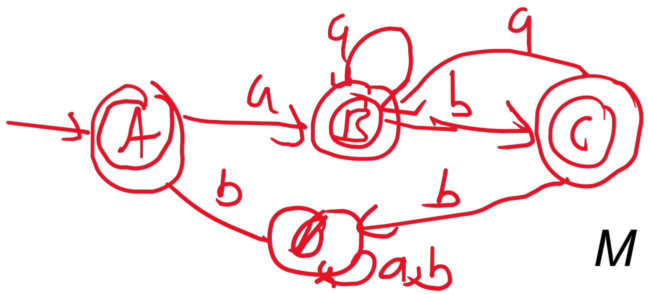
$\delta'(B, a) = \varepsilon C(\delta\{q_4, q_7, q_8, q_2, q_3, q_6\}, a))$
$= \varepsilon C(\{q_4, q_7\})$
$= \{q_4, q_7, q_8, q_2, q_3, q_6\} = B$

$\delta'(B, b) = \varepsilon C(\delta\{q_4, q_7, q_8, q_2, q_3, q_6\}, b))$
$= \varepsilon C(\{q_5\})$
$= \{q_5, q_8, q_2, q_3, q_6\} = C$

$\delta'(C, a) = \varepsilon C(\delta\{q_5, q_8, q_2, q_3, q_6\}, a))$
$= \varepsilon C(\{q_4, q_7\})$
$= \{q_4, q_7, q_8, q_2, q_3, q_6\} = B$

$\delta'(C, b) = \varepsilon C(\delta\{q_5, q_8, q_2, q_3, q_6\}, b)) = \emptyset$

δ'	a	b
A	B	$\emptyset$
B	B	C
C	B	$\emptyset$



$F' = \{A, B, C\}$

$\because q_8 \in A, B, C$

$M = (Q', \Sigma, \delta', q'_0, F')$.

$L = \{\varepsilon, a, ab, aab, aba, \ldots\ aaaabaaabaa, \ldots\}$

Eg RL -> RE -> NFA

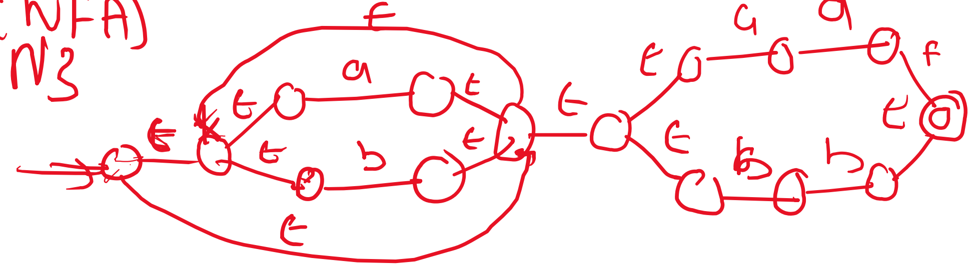
- *Example:*
  - $L_1 = \{wbb \mid w \in \{a,b\}^*\}$ = strings that ends with *bb*
  - $L_2 = \{waa \mid w \in \{0,1\}^*\}$ = strings that ends with *aa*
  - $L = L_1 \cup L_2 = \{x \in \{0,1\}^* \mid x$ ends with *bb* **or** *aa*$\}$

- Possible options
  - Construct DFA directly for L (03-NFA Slide 28)
  - Construct DFA M_1, DFA M_2 and then construct DFA M_3 from M_1 and M_2(03-NFA Slide 28)
  - Construct NFA N_1, NFA N_2 and then construct NFA N_3 from N_1 and N_2,
  - Convert N_3 to M_3 (03-NFA Slide 29)

R_1=((a+b)*aa); R_2=((a+b)*bb)
R_3=((a+b)*aa) + ((a+b)*bb)
= (a+b)*(aa + bb)

- Write RE R_1 for L_1 , R_2 for L_2
- $R_3 = R_1 \cup R_2$
- Construct N_3 from R_3
- Convert N_3 to M_3

(NFA)
N3



M3
(DFA)

Thank you!

Tutorial V

1. Interpret the following regular expressions over the input symbols {0,1}:
   - $(0 \cup 1)0^*$
   - $0(0 \cup 101)^*$
   - 0^*10^*
   - $\Sigma^*1\Sigma^*$

Tutorial IV

2. Give regular expressions that generate each of the following languages. In all cases, the alphabet is $\Sigma = \{a, b\}$.
   – The language $\{w \in \Sigma* \mid |w| \text{ is odd}\}$.
   – The language $\{ w \in \Sigma* \mid w \text{ ends in a double letter} \}$.
   – The language $\{ w \in \Sigma* \mid w \text{ has an odd number of a's} \}$
   – The language $\{ w \in \Sigma* \mid w \text{ contains at least two a's, or exactly two b's} \}$.

Tutorial IV

5. Suppose we define a restricted version of the Java programming language in which variable names must satisfy all of the following conditions:

a) A variable name can only use Roman letters (i.e., a, b, . . . , z, A, B, . . . , Z) or Arabic numerals (i.e., 0, 1, 2, . . . , 9); i.e., underscore and dollar sign are not allowed.

b) A variable name must start with a Roman letter: a, b, . . . , z, A, B, . . . , Z

c) The length of a variable name must be no greater than 8.

d) A variable name cannot be a keyword (e.g., if). The set of keywords is finite. Let L be the set of all valid variable names in our restricted version of Java.

Let L0 be the set of strings satisfying the first 3 conditions above; i.e., we do not require the last condition. Give a regular expression for L0.