Finite Automata with ε-Moves

- A NFA $M$ is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \ldots, q_n\}$ is a finite set of states
  - $\Sigma$ is a finite input alphabet
  - $\delta$ is a transition function mapping $Q \times (\Sigma \cup \{\varepsilon\})$ to $2^Q$.
    $2^Q$ is the power set of $Q$, the set of all subsets of $Q$.
  - $q_0 \in Q$ is the initial state.
  - $F \subseteq Q$ is the set of final states.
\(\varepsilon\)-Closures

- \(\varepsilon\)-closure of a state \(q\) is the set of states that can be reached from \(q\) along a path in which all arcs are labeled \(\varepsilon\).

- \(q\) is in \(\varepsilon\)-closure(\(q\))

- If \(p\) is in \(\varepsilon\)-closure(\(q\)) and there is an \(\varepsilon\) transition from \(p\) to \(r\), then \(r\) is in \(\varepsilon\)-closure(\(q\))
Transition Function \( \hat{\delta} \) on String

- \( \hat{\delta} \) is a transition function mapping \( Q \times \Sigma^* \) to \( Q \).

1. \( \hat{\delta}(q, \varepsilon) = \varepsilon\text{-closure}(q) \)
2. For all strings \( w = xa \), where \( x \) is a string and \( a \) is symbol in \( \Sigma \)
   - \( \hat{\delta}(q, x) = \{p_1, \ldots, p_k\} \)
   - \( \delta(p_1, a) \cup \delta(p_2, a) + \ldots + \delta(p_k, a) = \{r_1, \ldots, r_m\} \)
   - \( \hat{\delta}(q, w) = \varepsilon\text{-closure}(r_1) + \ldots + \varepsilon\text{-closure}(r_m) \)
Language Defined by An $\varepsilon$-NFA

- A string $x$ is said to be accepted by an $\varepsilon$-NFA $M = (Q, \Sigma, \delta, q_0, F)$ if $\hat{\delta}(q_0, x) = P$ and $P$ contains some final state.

- A language accepted by $\varepsilon$-NFA $M$, designated $L(M)$, is the set $\{x | \hat{\delta}(q_0, x) \text{ contains some state in } F\}$
Equivalence of NFA’s with and Without $\varepsilon$-Move

• Theorem:
  Let $L$ be a set accepted by a nondeterministic finite automaton with $\varepsilon$-Moves. Then there exists a deterministic finite automaton without $\varepsilon$-Move that accepts $L$.

• Theorem:
  Let $L$ be a set accepted by a deterministic finite automaton without $\varepsilon$-Move. Then there exists a nondeterministic finite automaton with $\varepsilon$-Moves that accepts $L$. 
Regular Language

• A language is regular if it can be defined by a DFA.

• Three equivalent machines:
  – DFA
  – NFA
  – $\varepsilon$-NFA
Regular Expressions

• Regular expression is an algebraic approach for defining a language over an alphabet.

• Each regular expression defines a language, namely a set of strings over an alphabet.

• Only a subset of languages, regular languages, can be defined by regular expressions.
Regular Expression Operators

• Union: +
  – Examples: \((0 + 1); (ab + cd)\)

• Concatenation: •
  – Examples: \(01•10 = 0110\)

• Star Closure: * (repeat 0 or more times)
  – \((00)^*\): the set of all strings with even number of 0s, including \(\varepsilon\).
Regular Expressions

• Regular expression over the alphabet $\Sigma$.

1. $\varepsilon$, $\phi$, $a \in \Sigma$ are regular expressions
2. if $r$ and $s$ are regular expressions, then
   - $r+s$ is a regular expression
   - $rs$ is a regular expression
   - $r^*$ is a regular expression
3. $r$ is a regular expression if and only if $r$ can be derived from 1 by a finite number of applications of 2.

$$r^+ = rr^*$$
Precendence

- * has the highest precedence
- Concatenation is the next
- + has the lowest precedence

- Examples:
  - ab*+c = (a(b)*)+c
Regular Expression Examples

- $(0+1)^*$
- $(a+b)^*a(a+b)^*$
- $b(a+b)^*b$
Language Defined by Regular Expression

- let L(r) represents the language defined by the regular expression r.
  1. $\emptyset$ is the language defined by $\varepsilon$
  2. $\phi$ is the language defined by $\phi$
  3. $\{a\}$ is the language defined by a
  4. $L(r+s) = L(r) \cup L(s)$
  5. $L(rs) = L(r)L(s)$
  6. $L(r^*) = L(r)^*$
Examples

• (0+1)*
  all binary strings

• (a+b)*a(a+b)*
  all strings with at least one \( a \)

• b(a+b)*b
  all strings starting with a \( b \) and ending with a \( b \)
Examples

• an odd number of a’s followed by an even number of b’s

• no more than 3 a’s and ends with ab

• binary strings with no substring 001
Equivalence

• Two regular expressions are equivalent if and only if the languages defined by the two expressions are the same.

• Examples:
  - $a^*(a+b)^* = (a+ba)^*$
  - $(a*b^*)^* = (a+b)^*$
Finite Languages

- Finite languages (languages with finite number of strings) are regular

- Assume L is a finite language with n strings $s_1, \ldots, s_n$. A regular expression defining L is $s_1^+ \ldots + s_n$
Equivalence of Finite Automata and Regular Expressions

• Theorem:
  Let $L$ be a set accepted by a finite automaton. Then there exists a regular expression that accepts $L$.

• Theorem:
  Let $L$ be a set accepted by a regular expression. Then there exists a finite automaton that accepts $L$. 