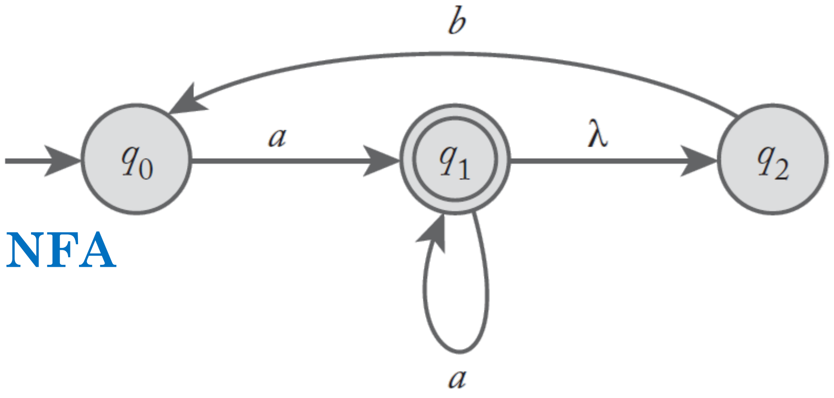
Theory of Machines and Languages

Fatemeh Deldar

1403-1404

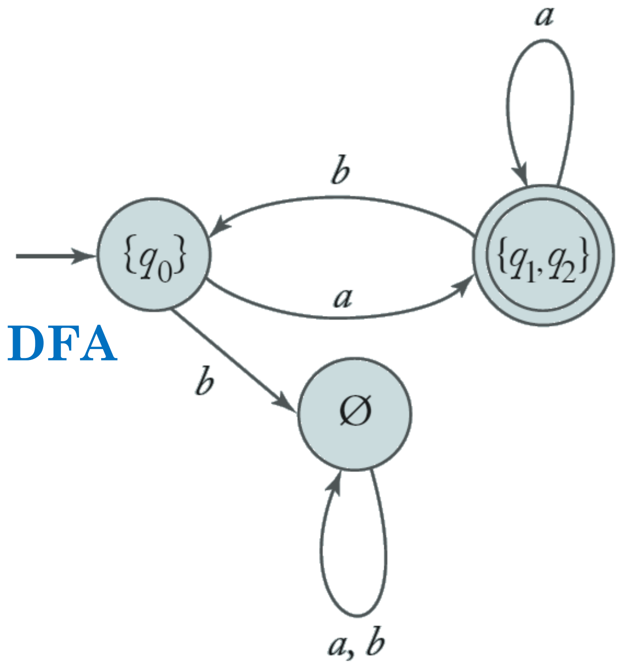
Equivalence of Dfa's and Nfa's

- The classes of dfa's and nfa's are equally powerful: For every language accepted by some nfa, there is a dfa that accepts the same language
- **Example**



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

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

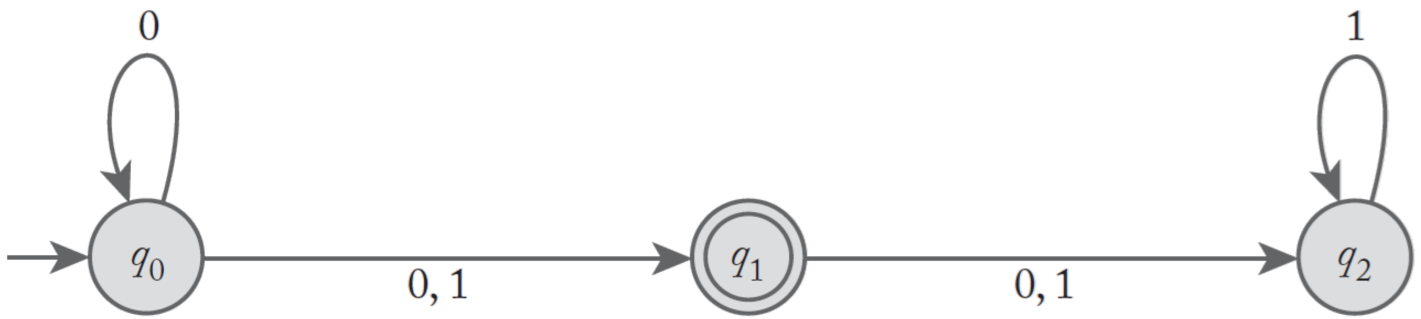


Equivalence of Dfa's and Nfa's

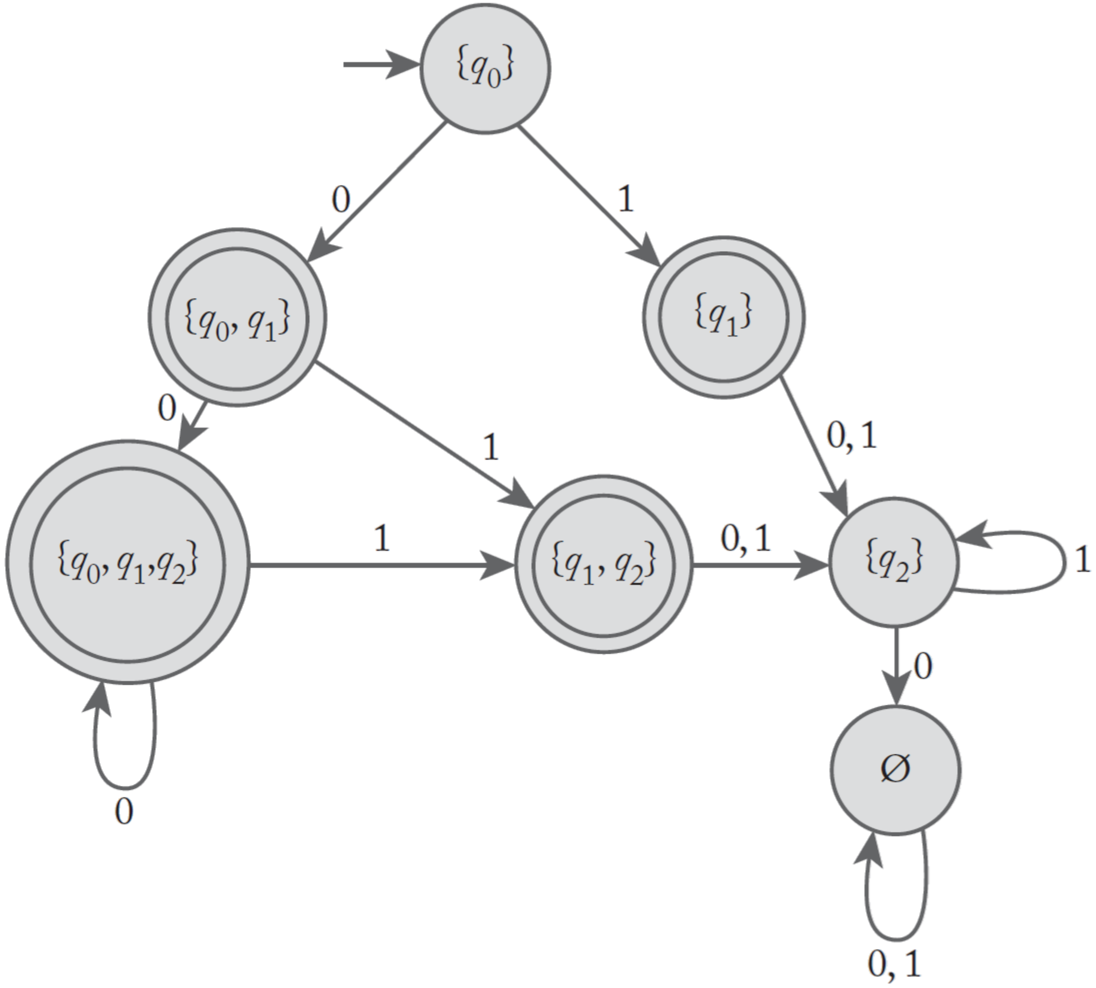
Let L be the language accepted by a nondeterministic finite accepter $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$. Then there exists a deterministic finite accepter $M_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ such that

$$L = L(M_D).$$

- **Example**



Equivalence of Dfa's and Nfa's



Example

Prove that for every nfa with an arbitrary number of final states there is an equivalent nfa with only one final state. Can we make a similar claim for dfa's?

Example

Find an nfa without λ-transitions and with a single final state that accepts the set $\{a\} \cup \{b^n : n \geq 2\}$.

Regular Languages and Regular Grammars

Regular Expressions

- **One way of describing regular languages is via the notation of regular expressions**
- **This notation involves a combination of strings of symbols from some alphabet Σ, parentheses, and the operators $+$, $\cdot$, and $*$**
- **Example**
  - $(\boldsymbol{a} + (\boldsymbol{b} \cdot \boldsymbol{c}))^*$
    - $\{\lambda, a, bc, aa, abc, bca, bcbc, aaa, aabc, \ldots\}$

Regular Expressions

- **Formal Definition**

  Let Σ be a given alphabet. Then

  1. $\varnothing, \lambda$, and $a \in \Sigma$ are all regular expressions. These are called **primitive regular expressions**.

  2. If r_1 and r_2 are regular expressions, so are $r_1 + r_2$, $r_1 \cdot r_2$, r_1^*, and (r_1).

  3. A string is a regular expression if and only if it can be derived from the primitive regular expressions by a finite number of applications of the rules in (2).

- **Example**
  - $(a + b \cdot c)^* \cdot (c + \varnothing)$ is a regular expression
  - $(a + b +)$ is not a regular expression

Regular Expressions

❑ Languages Associated with Regular Expressions

The language $L(r)$ denoted by any regular expression r is defined by the following rules.

1. $\varnothing$ is a regular expression denoting the empty set.
2. λ is a regular expression denoting $\{\lambda\}$.
3. For every $a \in \Sigma$, a is a regular expression denoting $\{a\}$.

   If r_1 and r_2 are regular expressions, then

4. $L(r_1 + r_2) = L(r_1) \cup L(r_2)$,
5. $L(r_1 \cdot r_2) = L(r_1) L(r_2)$,
6. $L((r_1)) = L(r_1)$,
7. $L(r_1^*) = (L(r_1))^*$.

Regular Expressions

- **Example**
  - $$L(a^* \cdot (a+b)) = L(a^*)\, L(a+b)$$
    $$= (L(a))^* (L(a) \cup L(b))$$
    $$= \{\lambda, a, aa, aaa, ...\}\{a, b\}$$
    $$= \{a, aa, aaa, ..., b, ab, aab, ...\}$$
- **Precedence rules**
  - Star-closure
  - Concatenation
  - Union
- **Example**
  - $r = (a+b)^* (a+bb)$ ⇒ $L(r) = \{a, bb, aa, abb, ba, bbb, ...\}$
  - $r = (aa)^* (bb)^* b$ ⇒ $L(r) = \{a^{2n} b^{2m+1} : n \geq 0,\ m \geq 0\}$