بسم اللّه الرّحمن الرّحیم

دانشگاه صنعتی اصفهان ــ دانشکدهٔ مهندسی برق و کامپیوتر
(نیم‌سال تحصیلی ۴۰۲۱)

نظریهٔ زبان‌ها و ماشین‌ها

حسین فلسفین

The Extended Transition Function δ^* for a DFA

It is convenient to introduce the extended transition function $\delta^* : Q \times \Sigma^* \mapsto Q$. *The second argument of* δ^* *is a string, rather than a single symbol, and its value gives the state the automaton will be in after reading that string.*

Let $M = (Q, \Sigma, \delta, q_0, F)$ *be a DFA. We define the extended transition function* $\delta^* : Q \times \Sigma^* \mapsto Q$ *as follows:*

1. *For every* $q \in Q$, $\delta^*(q, \varepsilon) = q$
2. *For every* $q \in Q$, *every* $y \in \Sigma^*$, *and every* $\sigma \in \Sigma$, $\delta^*(q, y\sigma) = \delta(\delta^*(q, y), \sigma)$

Acceptance by a DFA

Let** $M = (Q, \Sigma, \delta, q_0, F)$ **be a DFA, and let** $x \in \Sigma^*$**. The string** x **is accepted by** M **if** $\delta^*(q_0, x) \in F$ **and is rejected by** M **otherwise.

The language accepted by** M **is the set

$$L(M) = \{x \in \Sigma^* | x \text{ is accepted by } M\}.$$

If** L **is a language over** Σ**,** L **is accepted by** M **if and only if** $L = L(M)$**.

اینجا بحث دربارهٔ *DFA*ها خاتمه می‌یابد؛ وارد بحث عدم‌قطعیت و *NFA*ها می‌شویم.

Nondeterminism & NFAs

☞ *Deterministic computation: When the machine is in a given state and reads the next input symbol, we know what the next state will be—it is determined.*

☞ *In a nondeterministic machine, several choices may exist for the next state at any point.*

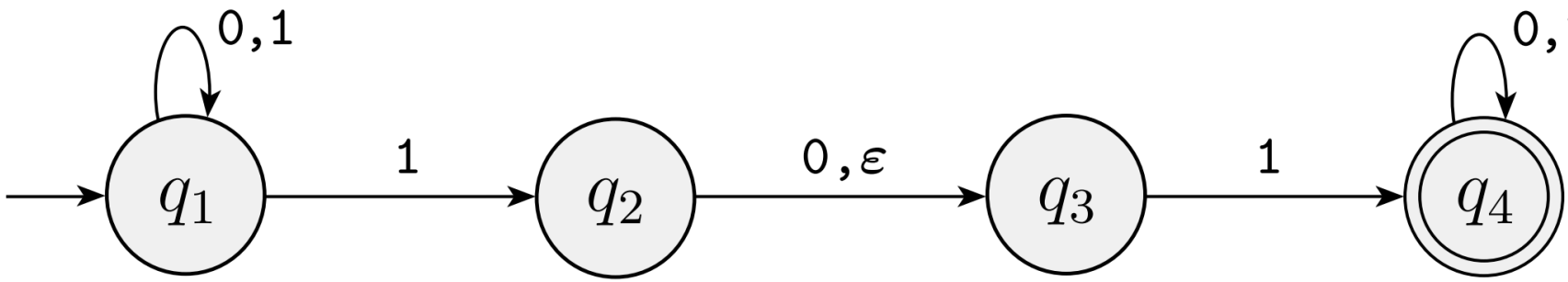
☞ *Nondeterminism is a generalization of determinism, so every DFA is automatically an NFA.*

☞ *It is rather obvious that an NFA is not a machine that we can build directly. So why is it worth considering? One reason is simply that this model has more expressive power than the DFA in the sense that it is easier to design NFAs than DFAs for some languages, and such NFAs usually have fewer states than the corresponding DFA.*

Nondeterminism is, at first sight, an unusual idea. Computers are deterministic machines, and the element of choice seems out of place. Nevertheless, nondeterminism is a useful concept, as we will see.

Nondeterministic finite automata are useful in several respects. As we will show, every NFA can be converted into an equivalent DFA, and constructing NFAs is sometimes easier than directly constructing DFAs. An NFA may be much smaller than its deterministic counterpart, or its functioning may be easier to understand.

دو تمایز عمدهٔ یک *NFA* با یک *DFA*



☞ ***First,*** *every state of a DFA always has exactly one exiting transition arrow for each symbol in the alphabet. In an NFA, a state may have zero, one, or many exiting arrows for each alphabet symbol.*

☞ ***Second,*** *in a DFA, labels on the transition arrows are symbols from the alphabet. This NFA has an arrow with the label* ε.

How does an NFA compute?

☞ *Suppose that we are running an NFA on an input string and come to a state with multiple ways to proceed. After reading that symbol, the machine splits into multiple copies of itself and follows all the possibilities in parallel. Each copy of the machine takes one of the possible ways to proceed and continues as before. If there are subsequent choices, the machine splits again.*

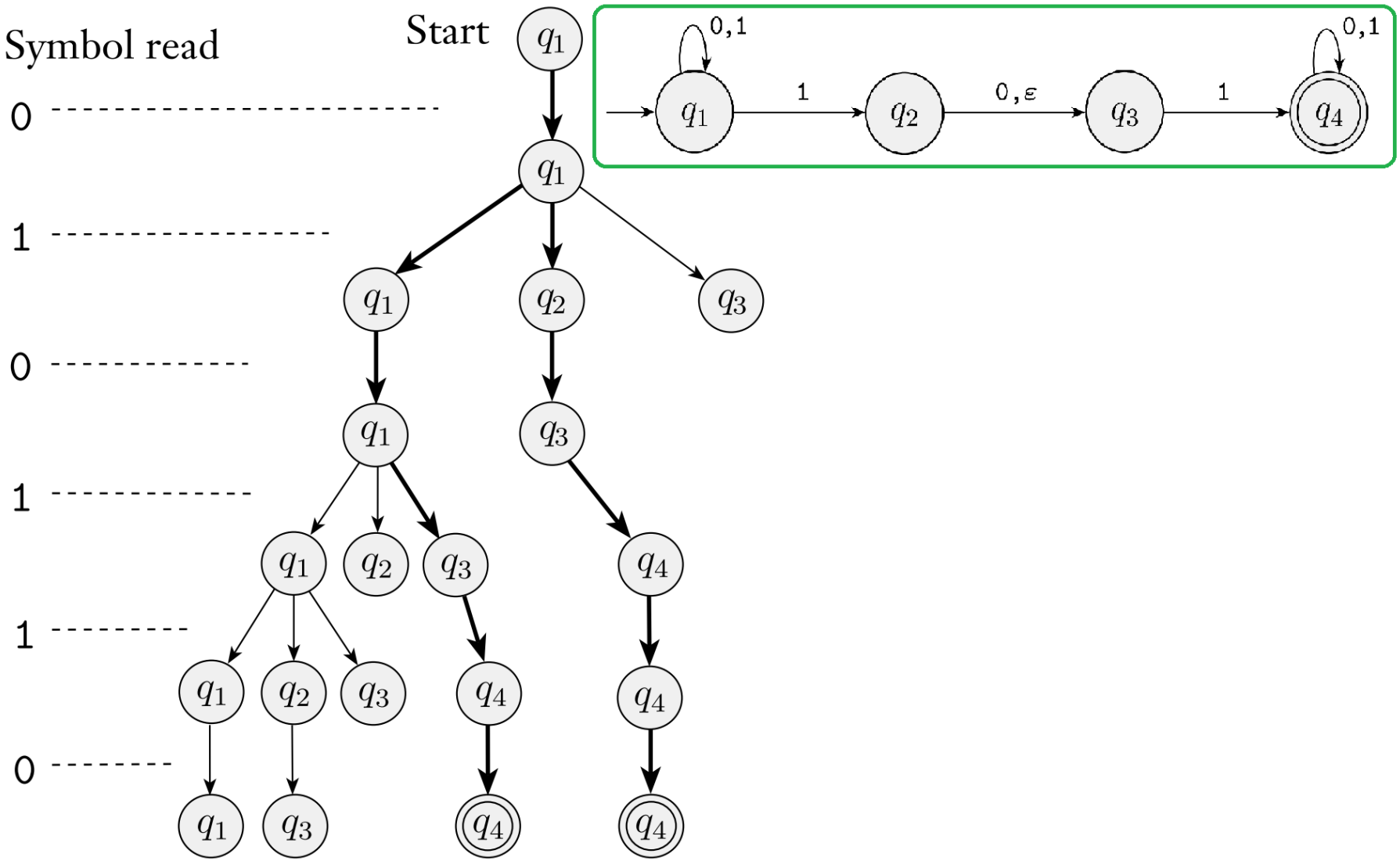
☞ *If the next input symbol doesn't appear on any of the arrows exiting the state occupied by a copy of the machine, that copy of the machine dies, along with the branch of the computation associated with it.*

☞ *Finally, if any one of these copies of the machine is in an accept state at the end of the input, the NFA accepts the input string.*

☞ *If a state with an ε symbol on an exiting arrow is encountered, then without reading any input, the machine splits into multiple copies, one following each of the exiting ε-labeled arrows and one staying at the current state. Then the machine proceeds nondeterministically as before.*

Tree of Possibilities (Computation Tree)

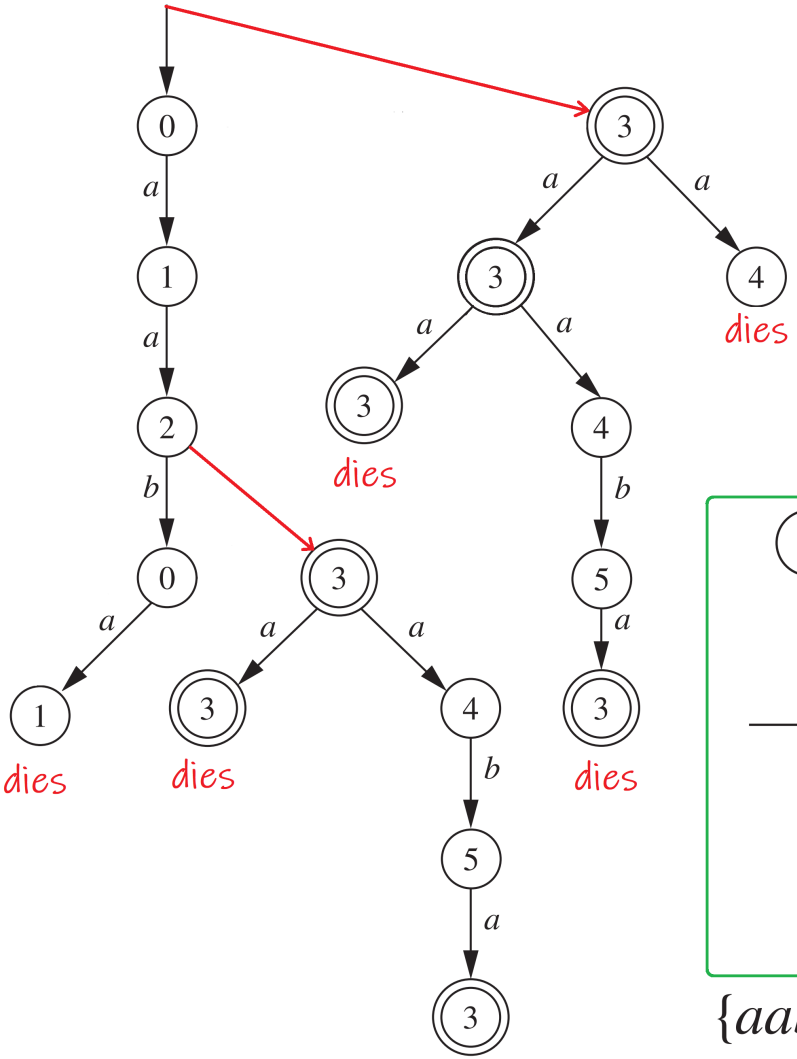
Example 1: The computation of N_1 on input 010110

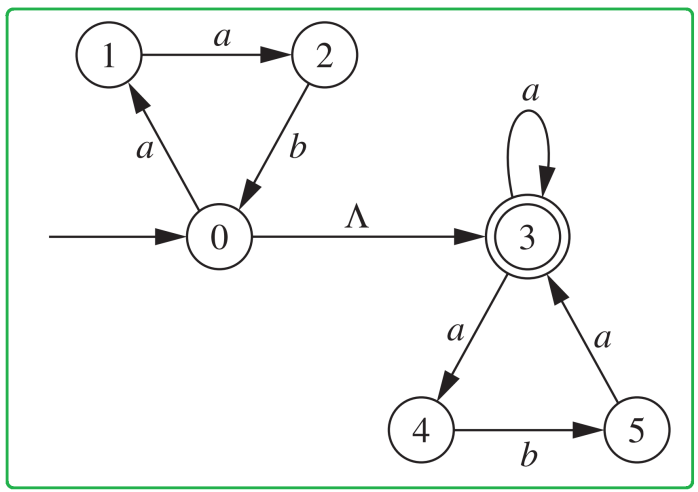


Example 2:

The input string is:

aababa

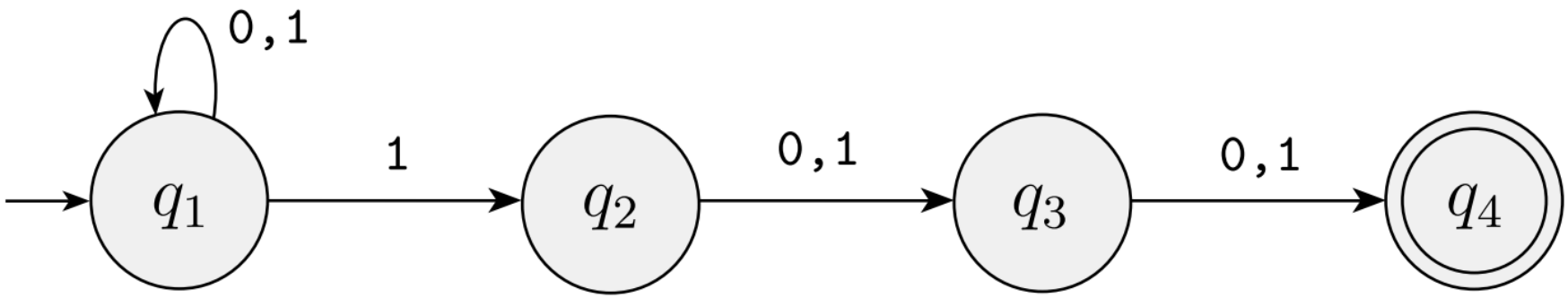




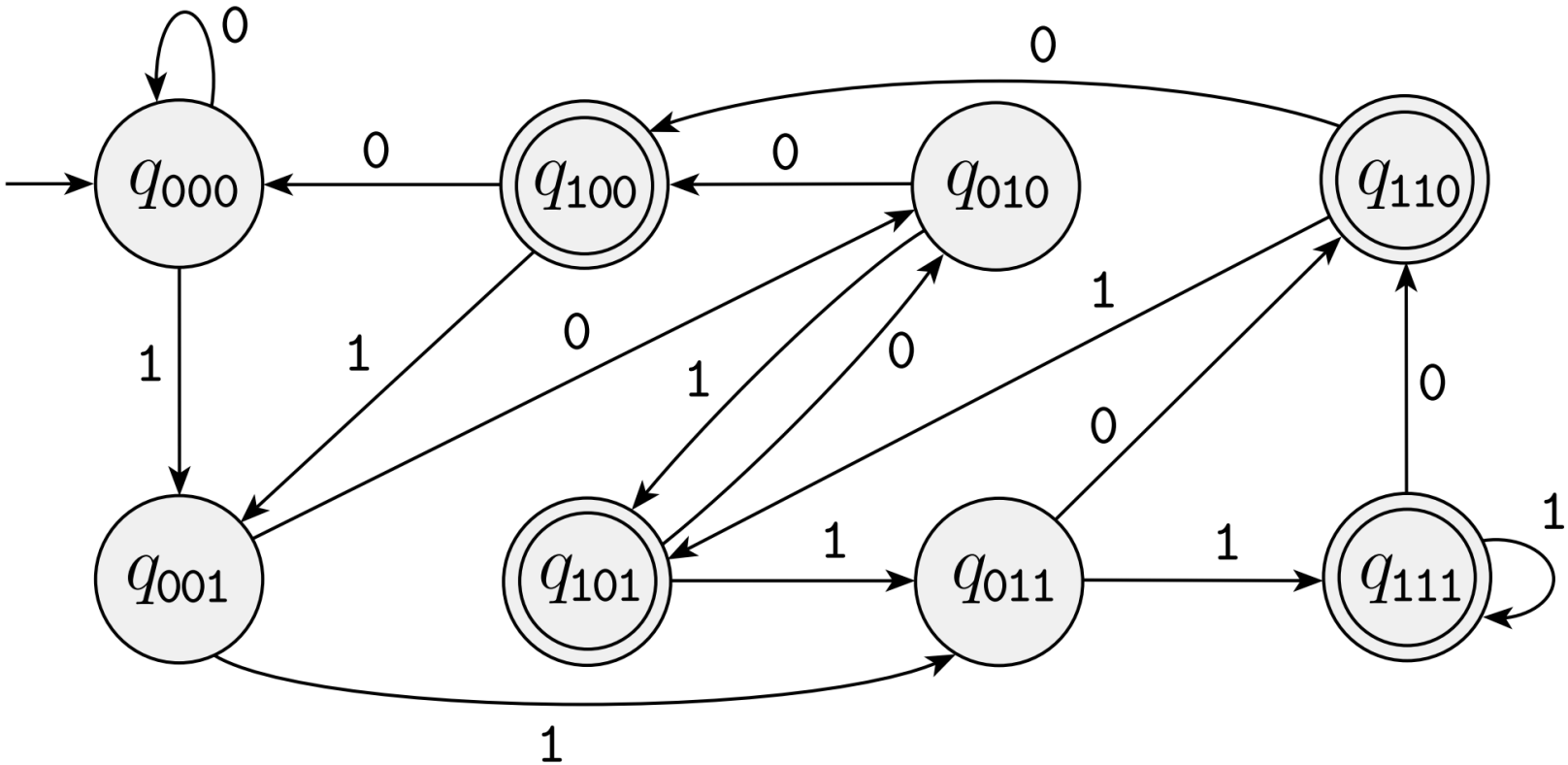
$\{aab\}^*\{a, aba\}^*$

Example:

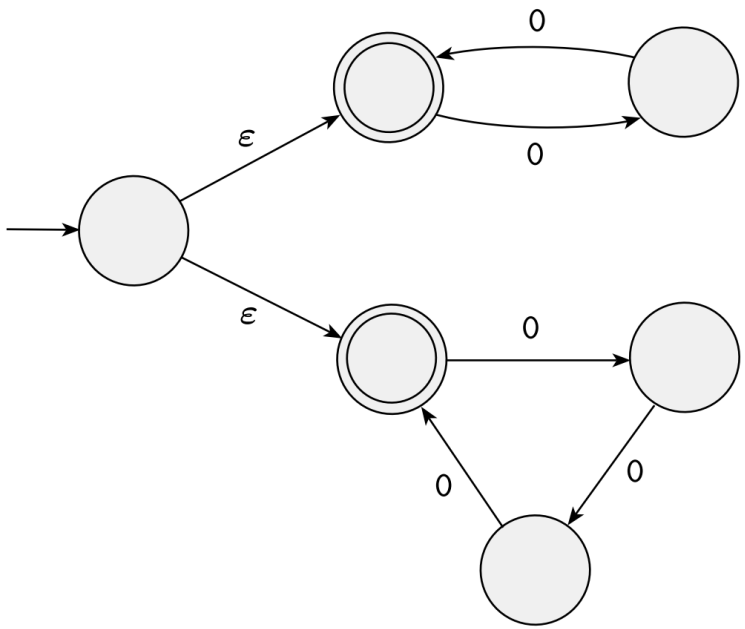
Let** A **be the language consisting of all strings over** $\{0, 1\}$ **containing a** 1 **in the third position from the end (e.g.,** 000100 **is in** A **but** 0011 **is not). The following four-state NFA recognizes** A**.



As mentioned earlier, every NFA can be converted into an equivalent DFA; but sometimes that DFA may have many more states. The smallest DFA for A contains eight states. Furthermore, understanding the functioning of the NFA is much easier, as you may see by examining the following figure for the DFA.

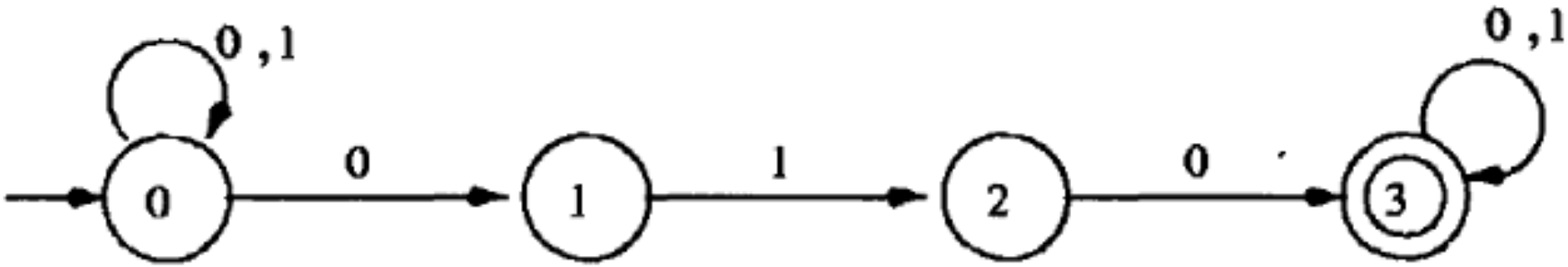


Example: This machine demonstrates the convenience of having ε ***arrows. It accepts all strings of the form*** 0^k ***where*** k ***is a multiple of*** 2 ***or*** 3***. For example, it accepts the strings*** ε, 00, 000, 0000, ***and*** 000000, ***but not*** 0 ***or*** 00000.



Of course, we could replace this machine by one that doesn't have ε ***arrows or even any nondeterminism at all, but the machine shown is the easiest one to understand for this language.***

Example: Find an NFA that accepts the set of binary strings having a substring 010***.***



تعریف رسمی یک NFA

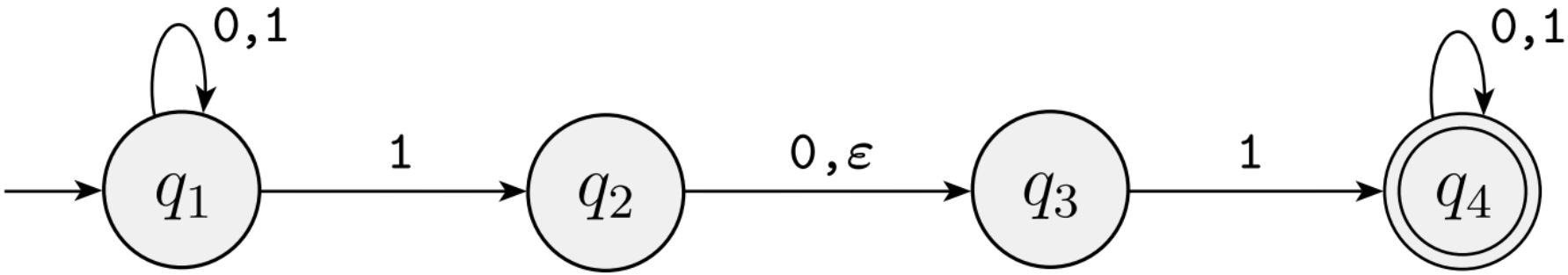
A ***nondeterministic finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

For any set Q we write $\mathcal{P}(Q)$ to be the collection of all subsets of Q. It can also be denoted by 2^Q. Here $\mathcal{P}(Q)$ is called the power set of Q. For any alphabet Σ we write Σ_ε to be $\Sigma \cup \{\varepsilon\}$.

یادآوری: برای یک **DFA**، تابع انتقال به‌شکل $\delta : Q \times \Sigma \mapsto Q$ بود.

مثال:



1. $Q = \{q_1, q_2, q_3, q_4\}$,

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

3. δ is given as

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$,

4. q_1 is the start state, and

5. $F = \{q_4\}$.

دربارهٔ هم‌ارزی *DFA*ها و *NFA*ها

☞ *Deterministic and nondeterministic finite automata recognize the same class of languages. Such equivalence is both surprising and useful. It is surprising because NFAs appear to have more power than DFAs, so we might expect that NFAs recognize more languages. It is useful because describing an NFA for a given language sometimes is much easier than describing a DFA for that language.*

☞ *Say that two machines are equivalent if they recognize the same language.*

☞ *Given an NFA that accepts some language L, there exists an equivalent DFA that accepts L.*

☞ *Since a DFA is in essence a restricted kind of NFA, it is clear that any language that is accepted by a DFA is also accepted by some NFA.*

☞ *But the converse is not so obvious. We have added nondeterminism, so it is at least conceivable that there is a language accepted by some NFA for which, in principle, we cannot find a DFA. But it turns out that this is not so. 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.*

☞ *The argument will be constructive. This means that we can actually give a way of converting any NFA into an equivalent DFA.*

قبل از بیان اثبات، مفهوم بستار ε را معرفی می‌کنیم.

The ε-Closure of a Set of States

Suppose $M = (Q, \Sigma, \delta, q_0, F)$ is an NFA, and $R \subseteq Q$ is a set of states. The ε-closure of R is the set $E(R)$ that can be defined recursively as follows.

1. $R \subseteq E(R)$.
2. **For every $q \in E(R)$, $\delta(q, \varepsilon) \subseteq E(R)$.**

In other words, for any subset R of Q, we define $E(R)$ to be the collection of states that can be reached from members of R by going only along ε arrows, including the members of R themselves. Formally, for $R \subseteq Q$ let

$$E(R) = \{q \mid$$

q can be reached from R by traveling along 0 or more ε arrows$\}$.