بسم اللّه الرّحمن الرّحیم

دانشگاه صنعتی اصفهان ـ دانشکدهٔ مهندسی برق و کامپیوتر
(نیم‌سال تحصیلی ۴۰۰۱)

نظریهٔ زبان‌ها و ماشین‌ها

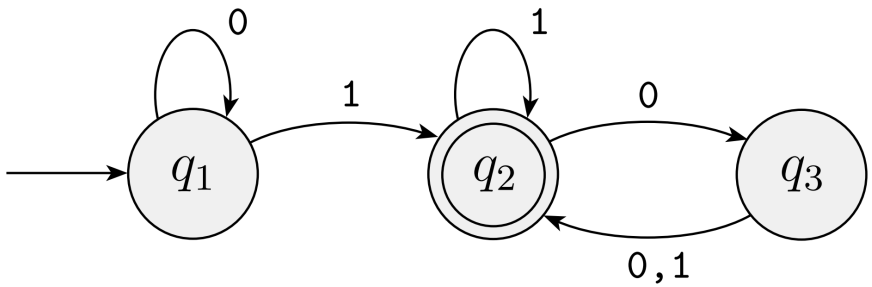
حسین فلسفین

The formal definition of finite automata

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

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

The formal definition precisely describes what we mean by a finite automaton.



We can describe M_1 formally by writing $M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0, 1\}$,
3. δ *is described as*

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2,

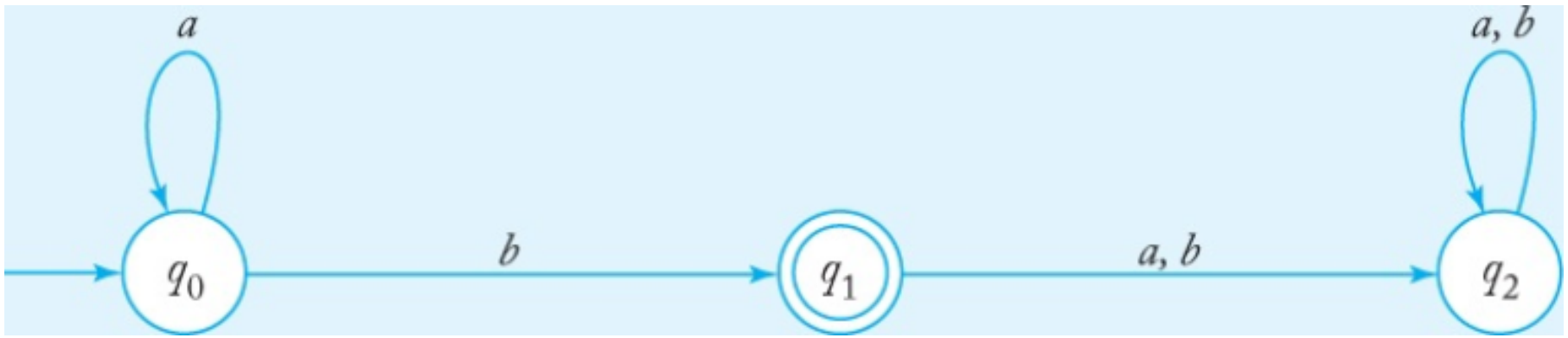
4. q_1 *is the start state, and*
5. $F = \{q_2\}$.

If A is the set of all strings that machine M accepts, we say that A is the language of machine M and write $L(M) = A$. We say that M recognizes A or that M accepts A. Because the term accept has different meanings when we refer to machines accepting strings and machines accepting languages, we prefer the term recognize for languages in order to avoid confusion. A machine may accept several strings, but it always recognizes only one language. If the machine accepts no strings, it still recognizes one language—namely, the empty language $\varnothing$. In our example, let

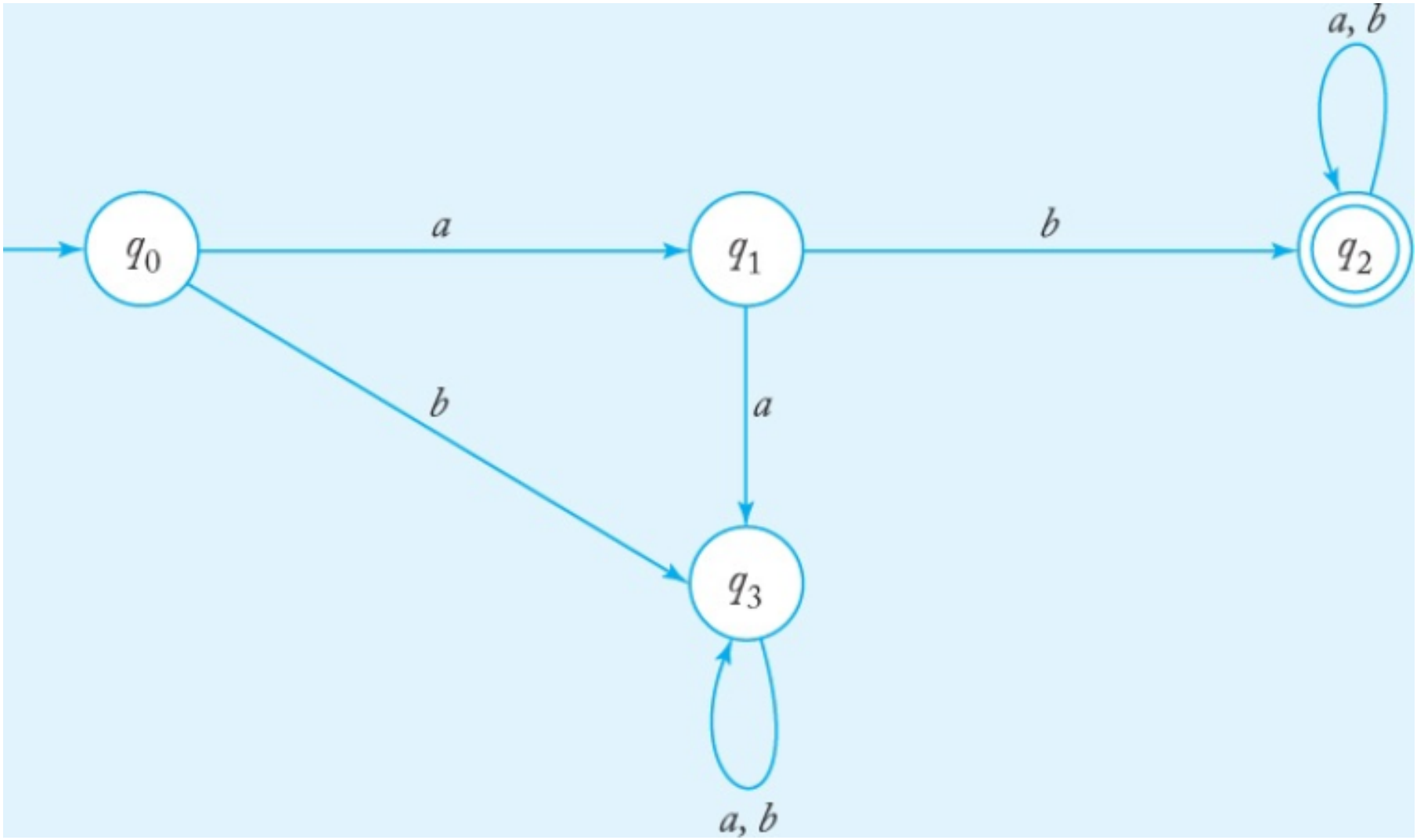
$$A = \{w \mid w \text{ contains at least one 1 and an even number of 0s follow the last 1}\}.$$

Then $L(M_1) = A$, or equivalently, M_1 recognizes A.

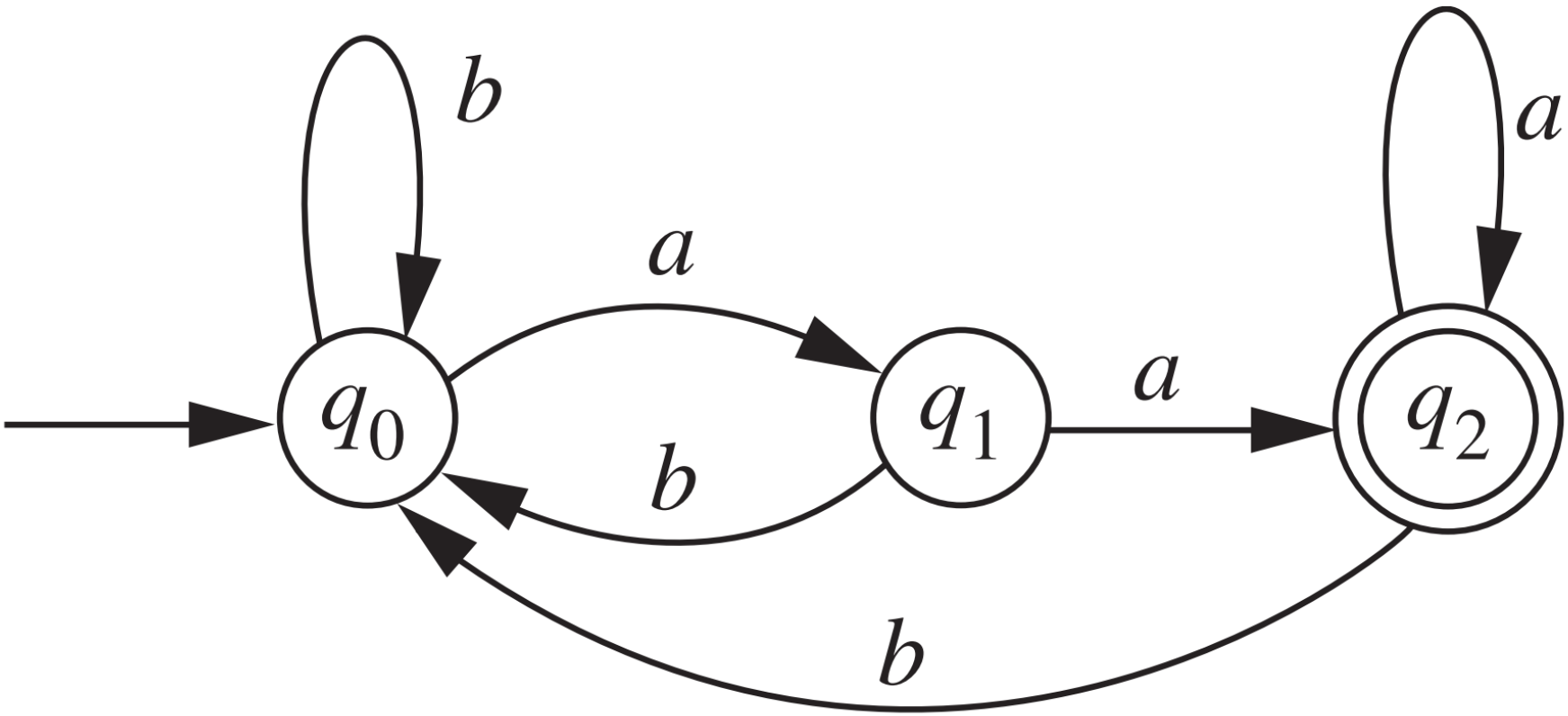
Example



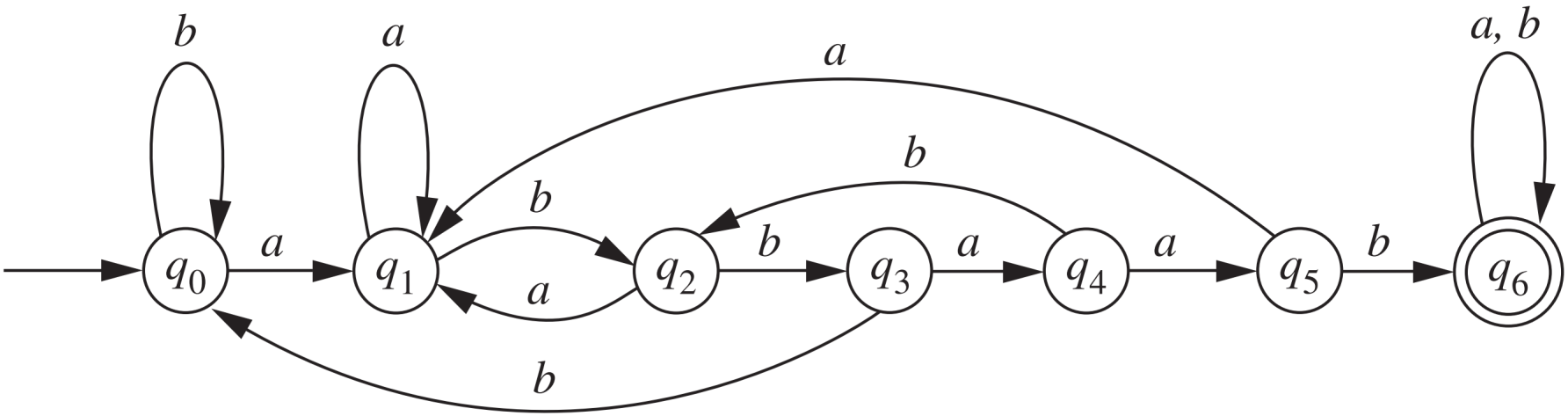
Example (prefix)



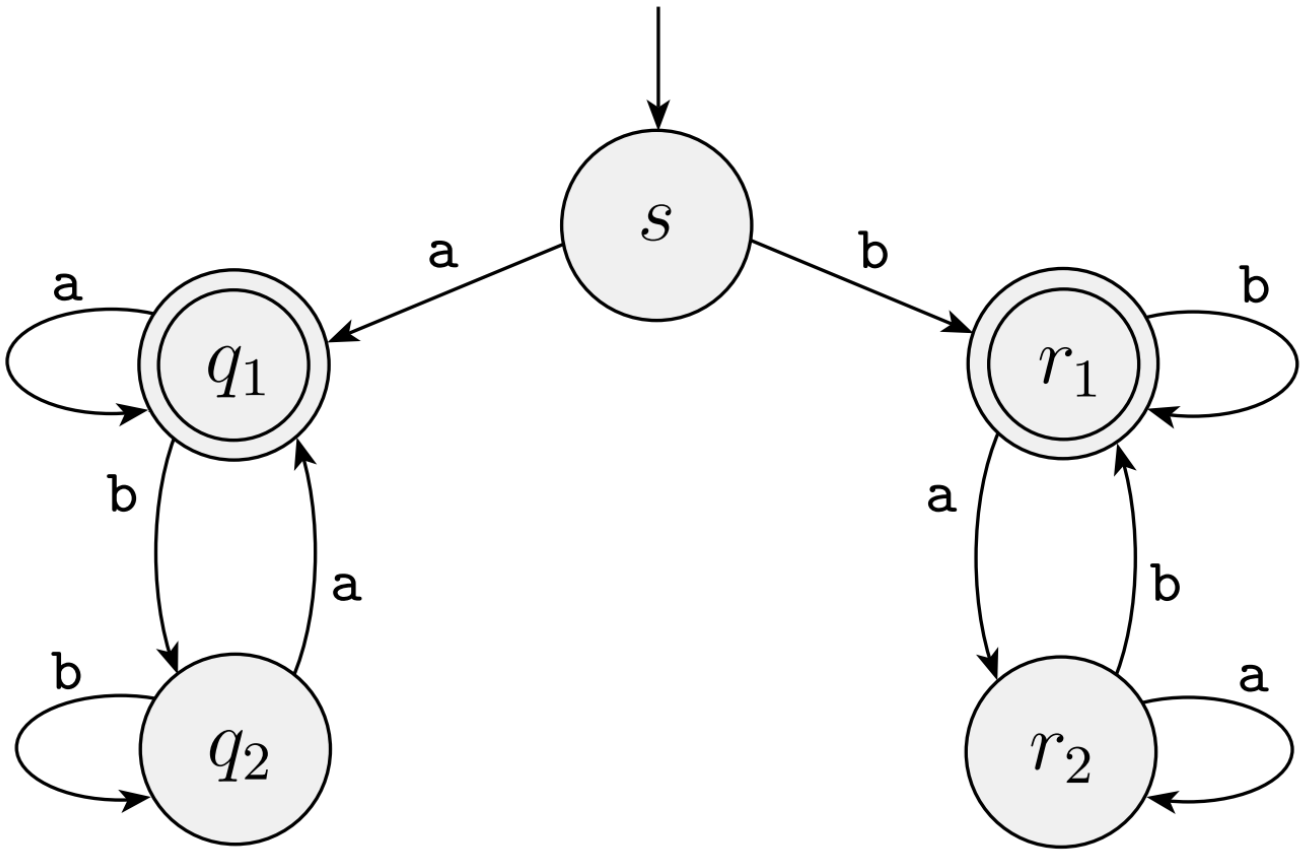
Example (suffix)



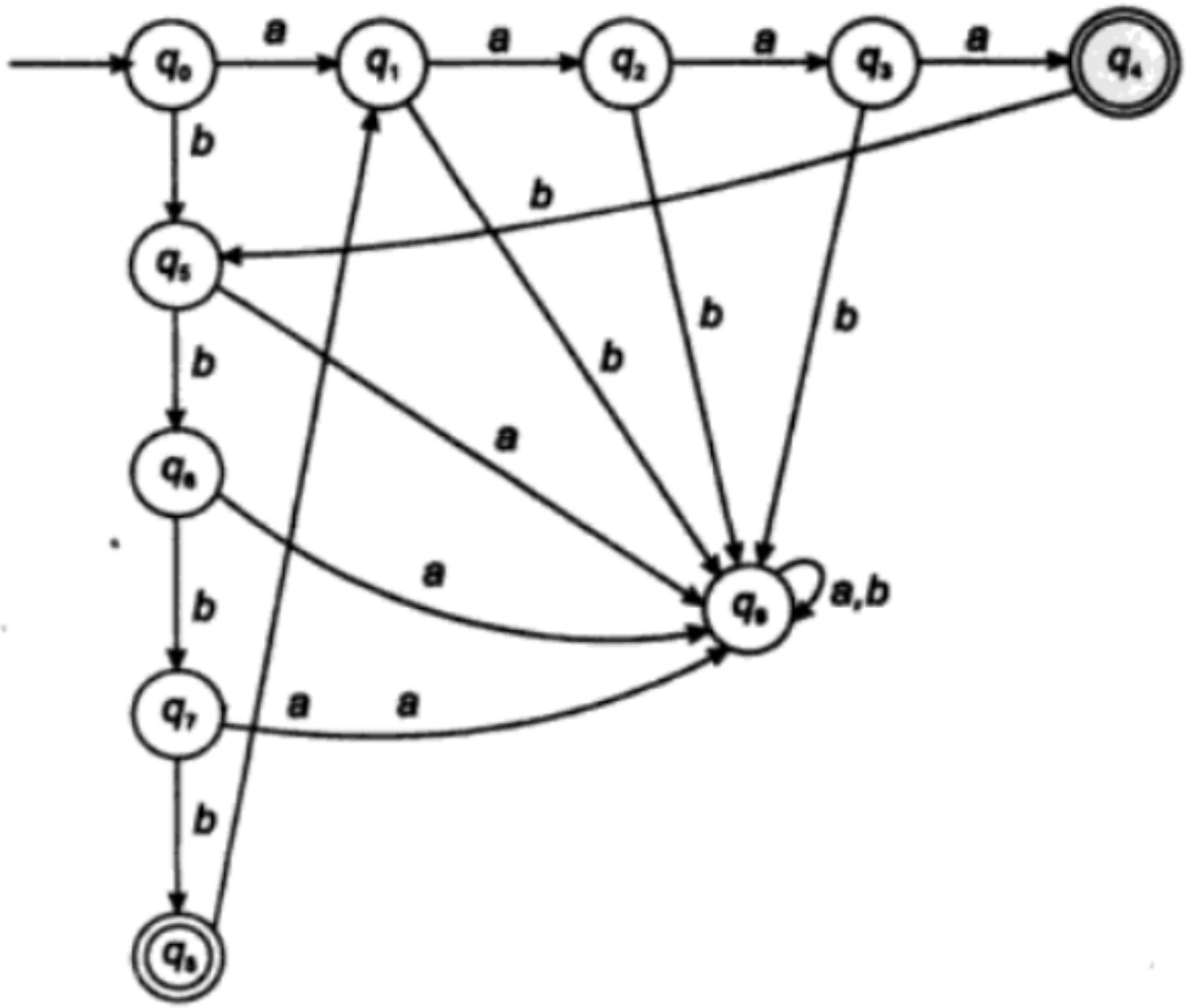
Example (substring)



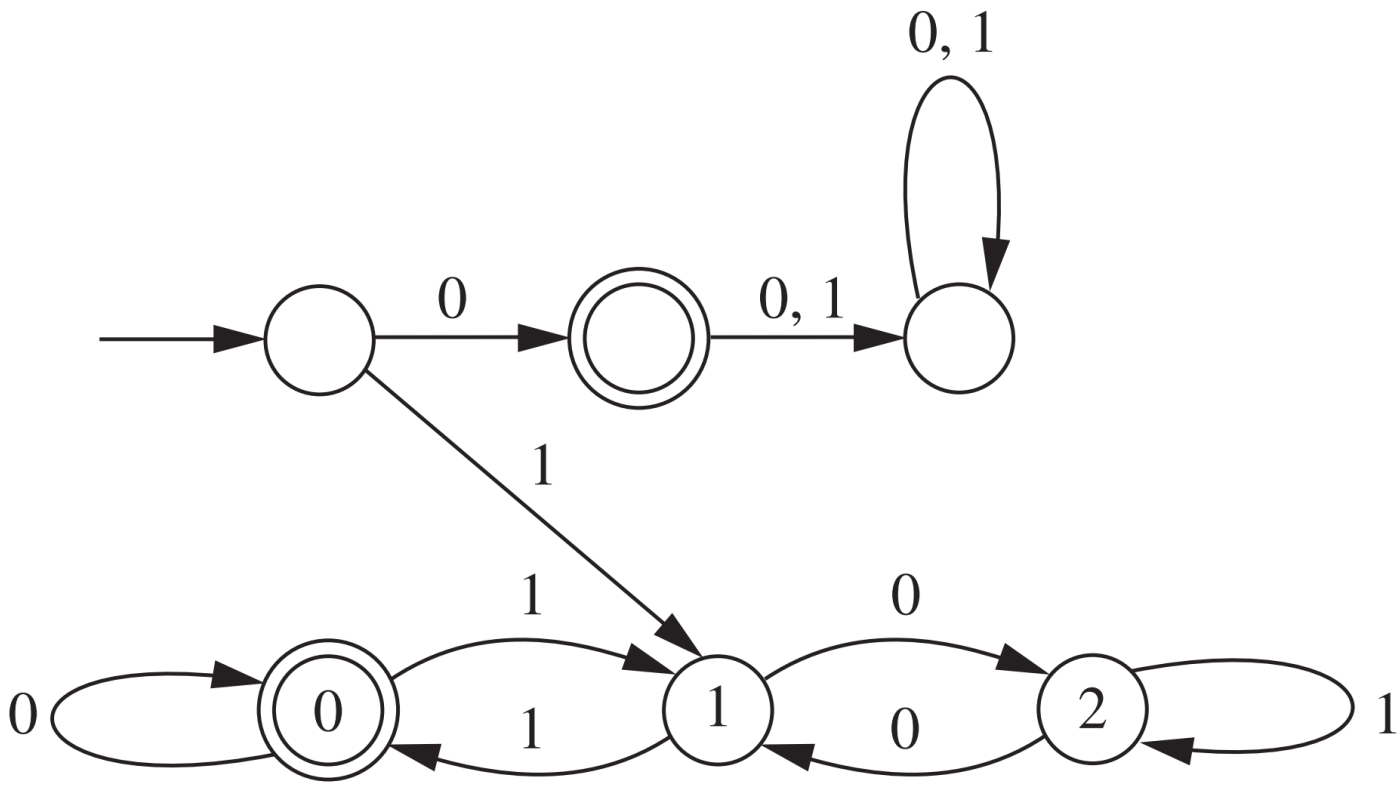
Example (strings that start and end with the same symbol)



Example (no runs of length less than four)

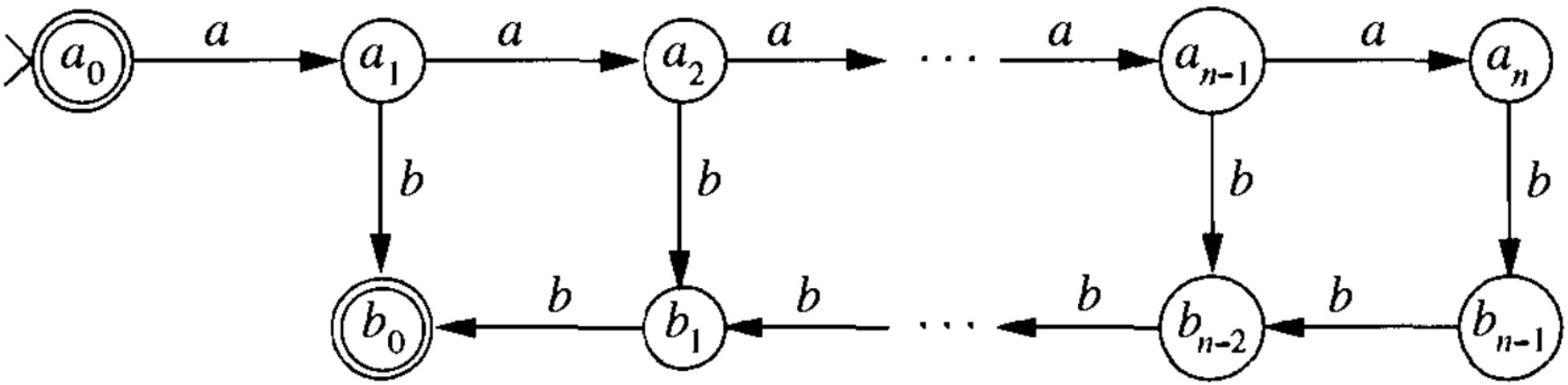


Example (an FA accepting binary representations of integers divisible by 3)



(We will disallow leading 0's in binary representations, except for the number 0 itself, and so we need one more state for the strings that start with 0 and have more than one digit.)

Example: The incompletely specified DFA defined by the state diagram given below accepts the language $\{a^i b^i \mid i \leq n\}$, for a fixed integer n. The states a_k count the number of a's, and then the b_k's ensure an equal number of b's.



This technique cannot be extended to accept $\{a^i b^i \mid i \geq 0\}$ since an infinite number of states would be needed. We will show that this language is not accepted by any finite automaton.

Formal Definition of Computation

We already have an informal idea of the way it computes, and we now formalize it mathematically.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and let $w = w_1 w_2 \cdots w_n$ be a string where each w_i is a member of the alphabet Σ. Then M accepts w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, *for* $i = 0, 1, \ldots, n-1$, *and*
3. $r_n \in F$.

We say that M recognizes language A if $A = \{w \mid M \text{ accepts } w\}$. A language is called a regular language if some finite automaton recognizes it.

The Regular Operations

We introduced and defined finite automata and regular languages. We now begin to investigate their properties. Doing so will help develop a toolbox of techniques for designing automata to recognize particular languages. The toolbox also will include ways of proving that certain other languages are nonregular (i.e., beyond the capability of finite automata).

In arithmetic, the basic objects are numbers and the tools are operations for manipulating them, such as $+$ and $\times$. In the theory of computation, the objects are languages and the tools include operations specifically designed for manipulating them. We define three operations on languages, called the regular operations, and use them to study properties of the regular languages.

Definition: *Let A and B be languages. We define the* ***regular operations*** ***union,*** ***concatenation,*** *and* ***star*** *as follows:*

- ☞ ***Union:*** $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
- ☞ ***Concatenation:*** $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$.
- ☞ ***Star:*** $A^* = \{x_1 x_2 \cdots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$.

The empty string ε is always a member of A^*, no matter what A is.

Example: ***Let the alphabet*** Σ ***be the standard 26 letters*** $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$**.**
If $A = \{\mathtt{good}, \mathtt{bad}\}$ ***and*** $B = \{\mathtt{boy}, \mathtt{girl}\}$**, then**

$$A \cup B = \{\mathtt{good}, \mathtt{bad}, \mathtt{boy}, \mathtt{girl}\},$$

$$A \circ B = \{\mathtt{goodboy}, \mathtt{goodgirl}, \mathtt{badboy}, \mathtt{badgirl}\},$$

and

$$A^* = \{\varepsilon, \mathtt{good}, \mathtt{bad}, \mathtt{goodgood}, \mathtt{goodbad}, \mathtt{badgood}, \mathtt{badbad},$$
$$\mathtt{goodgoodgood}, \mathtt{goodgoodbad}, \mathtt{goodbadgood}, \mathtt{goodbadbad}, \ldots\}.$$