LANGUAGES

Definition

- A system suitable for the expression of certain ideas, facts, or concepts, including a set of symbols and rules for their manipulation
- Not concise….
- We start with a finite, nonempty set $\sum$ of symbols , called the alphabet
- We then construct strings from finite sequences of symbols from the alphabet.
- For example, if the alphabet $\sum$ ={a,b}, then abab and aaabbba are strings on $\sum$

Terms

Concatenation

- appending e.g w=a1a2…an and v=b1b2….bn the concatenation of w and v, denoted by wv, is

$$wv = a1a2 \cdots anb1b2 \cdots bm.$$

Reverse

- The reverse of a string is obtained by writing the symbols in reverse order; if w is a string w = a1a2 ⋯an, then its reverse wR is

$$wR = an \cdots a2a1$$

Terms

Length

- The length of a string w, denoted by $|w|$, is the number of symbols in the string. We will frequently need to refer to the empty string, which is a string with no symbols at all. It will be denoted by λ.
- The following simple relations hold for all w.

$$|\lambda| = 0, \lambda w = w\lambda = w$$

Terms
Length

- Any string of consecutive symbols in some w is said to be a substring of w. If

$$w = vu,$$

then the substrings v and u are said to be a prefix and a suffix of w, respectively.

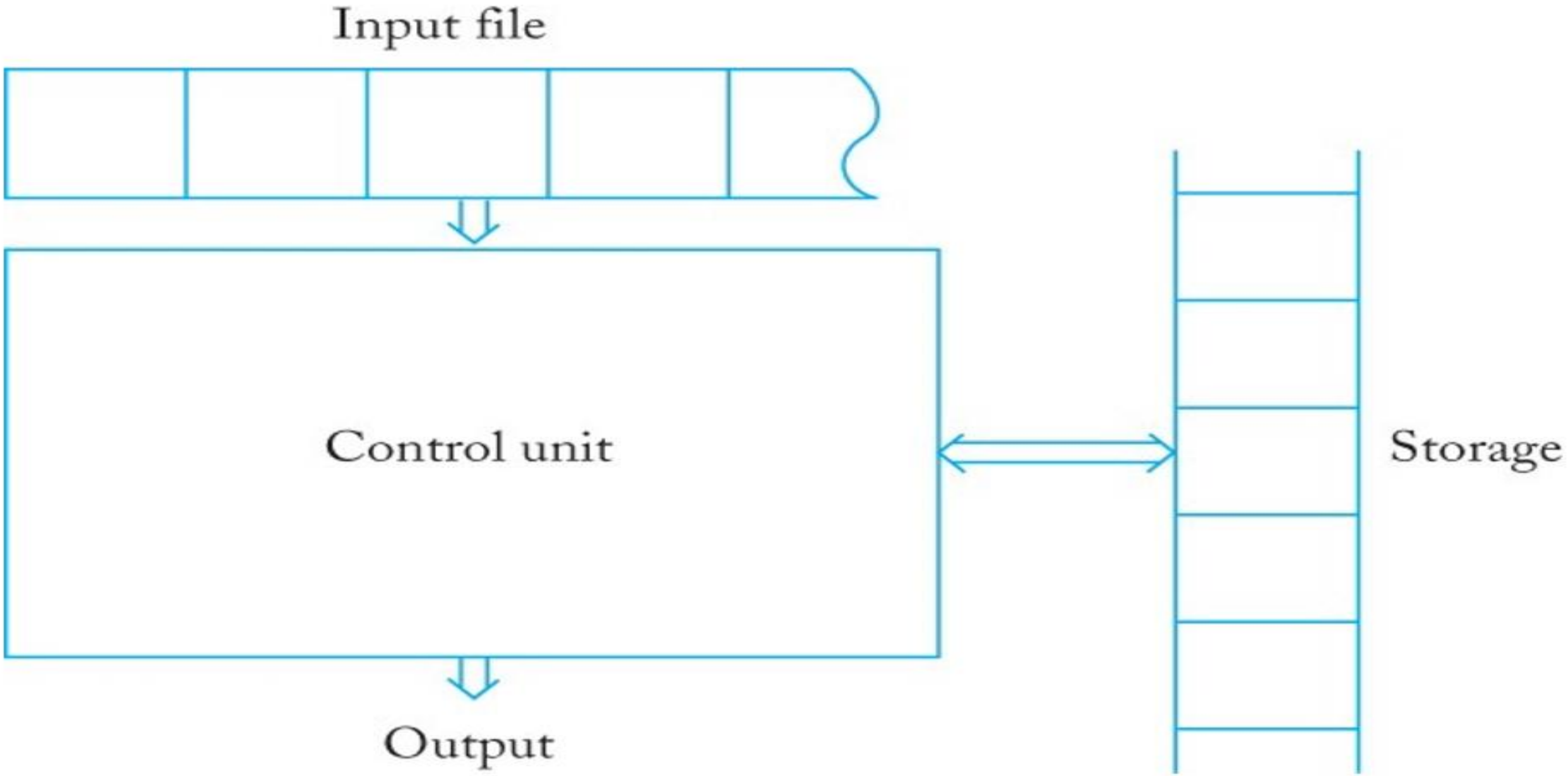
- For example, if w = abbab, then {λ, a, ab, abb, abba, abbab} is the set of all prefixes of w, while bab, ab, b are some of its suffixes.
- The length of the concatenation of two strings is the sum of the individual lengths, that is,

$$|uv| = |u| + |v|$$

Automata

- An automaton is an abstract model of a digital computer. As such, every automaton includes some essential features.
- It has a mechanism for reading input.
- It will be assumed that the input is a string over a given alphabet, written on an input file, which the automaton can read but not change.

Automata



Automata - Input

- The input file is divided into cells, each of which can hold one symbol.
- The input mechanism can read the input file from left to right, one symbol at a time.
- The input mechanism can also detect the end of the input string (by sensing an end-of-file condition).

Automata - Output

- The automaton can produce output of some form.
- It may have a temporary storage device, consisting of an unlimited number of cells, each capable of holding a single symbol from an alphabet (not necessarily the same one as the input alphabet).
- The automaton can read and change the contents of the storage cells

Automata - Process

- Finally, the automaton has a control unit, which can be in any one of a finite number of internal states, and which can change state in some defined manner

FINITE AUTOMATA

- We begin with finite accepters, which are a simple, special case of the general scheme introduced in the last chapter.
- This type of automaton is characterized by having no temporary storage.
- Since an input file cannot be rewritten, a finite automaton is severely limited in its capacity to "remember" things during the computation.

FINITE AUTOMATA…

- A finite amount of information can be retained in the control unit by placing the unit into a specific state.
- But since the number of such states is finite, a finite automaton can only deal with situations in which the information to be stored at any time is strictly bounded.

Deterministic Accepters

- A deterministic finite accepter or dfa is defined by the quintuple

  $M = (Q, \Sigma, \delta, q0, F)$,

- where
  - Q is a finite set of internal states,
  - Σ is a finite set of symbols called the input alphabet,
  - $\delta : Q \times \Sigma \rightarrow Q$ is a total function called the transition function,
  - $q0 \in Q$ is the initial state,
  - $F \subseteq Q$ is a set of final states.

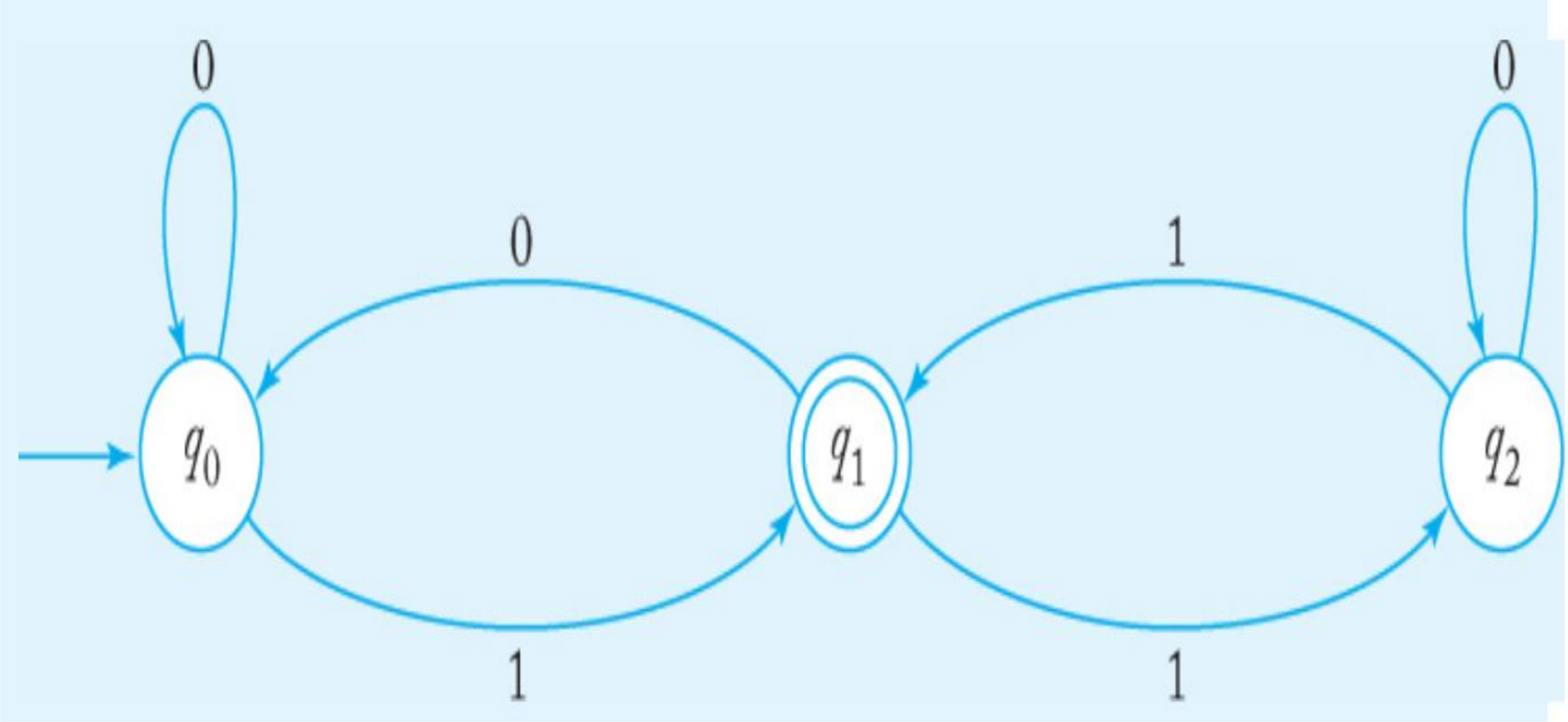
Deterministic Accepters: Operation

- At the initial time, it is assumed to be in the initial state q0, with its input mechanism on the leftmost symbol of the input string.
- During each move of the automaton, the input mechanism advances one position to the right, so each move consumes one input symbol.
- When the end of the string is reached, the string is accepted if the automaton is in one of its final states, otherwise the string is rejected.

Deterministic Accepters: Operation

- The input mechanism can move only from left to right and reads exactly one symbol on each step.
- The transitions from one internal state to another are governed by the transition function
- δ. For example, if $\delta(q0, a) = q1$, then if the dfa is in state q0 and the current input symbol is a, the dfa will go into state q1.

Deterministic Accepters: Transition Graphs



Deterministic Accepters: Transition Graphs

- To visualize and represent finite automata, we use transition graphs, in which the vertices represent states and the edges represent transitions.
- The labels on the vertices are the names of the states, while the labels on the edges are the current values of the input symbol. For example, if q0 and q1 are internal states of some dfa M, then the graph associated with M will have one vertex labeled q0 and another labeled q1.

Deterministic Accepters: Transition Graphs

- An edge (q0, q1) labeled **a** represents the transition **δ (q0, a) = q1**.
- The **initial state** will be identified by an incoming unlabeled arrow not originating at any vertex.
- **Final states** are drawn with a double circle.
- More formally, if M = (Q, Σ, δ, q0, F) is a deterministic finite accepter, then its associated transition graph GM has exactly |Q| vertices, each one labeled with a different qi ∈ Q.

Deterministic Accepters: Transition Graphs

- For every transition rule δ (qi, a) = qj, the graph has an edge (qi, qj) labeled a.
- The vertex associated with q0 is called the initial vertex, while those labeled with qf ∈ F are the final vertices.
- It is a trivial matter to convert from the (Q, Σ, δ, q0, F) definition of a dfa to its transition graph representation and vice versa.

NONDETERMINISTIC FINITE ACCEPTERS

- In DFA’s, a unique transition is defined for each state and each input symbol
- 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.

Definition of NFA

- A nondeterministic finite accepter or nfa is defined by the quintuple

$$M = (Q, \Sigma, \delta, q0, F),$$

where Q, Σ, $q0$, F are defined as for deterministic finite accepters, but

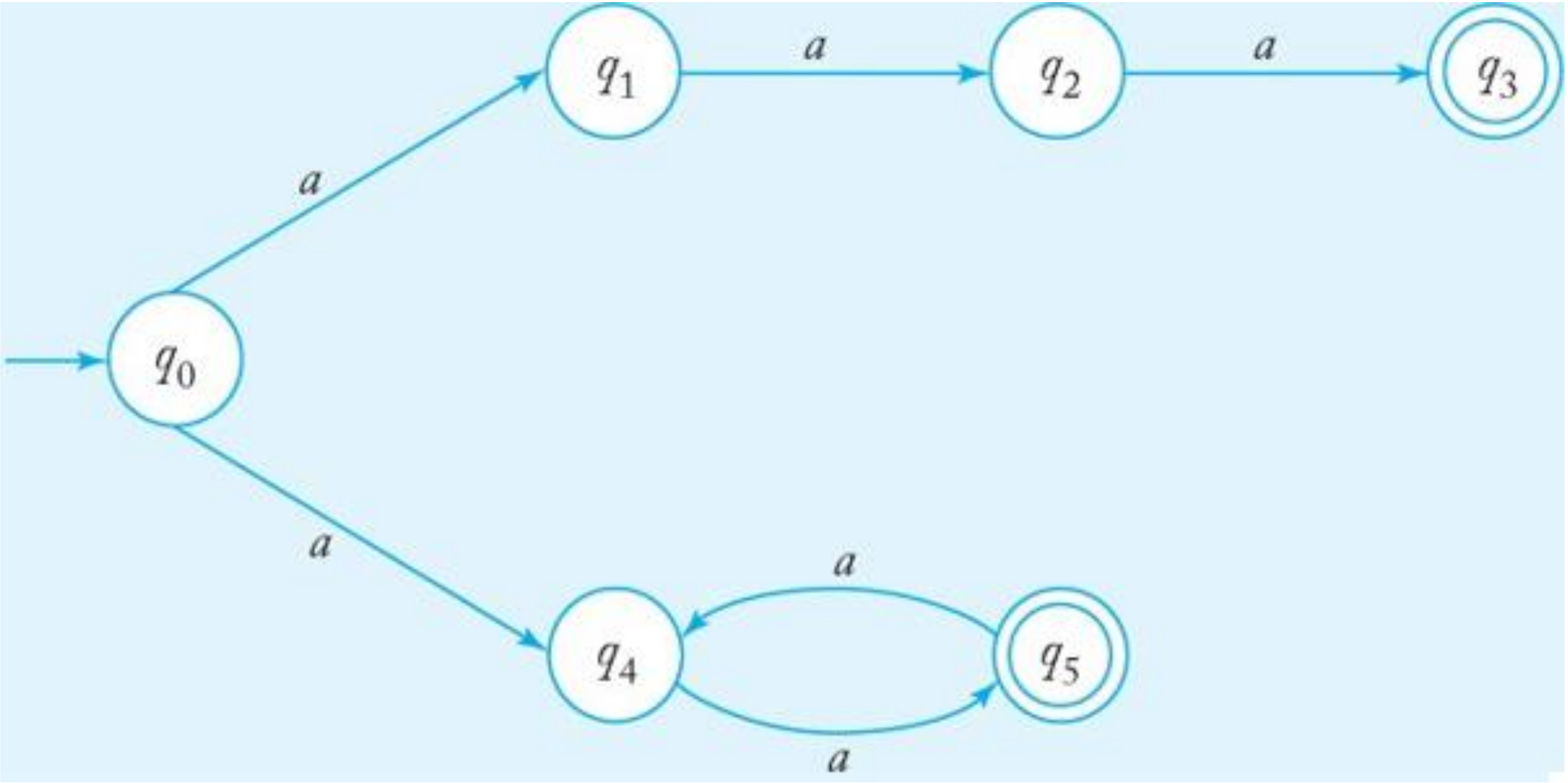
$\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^{Q}$.

Definition of NFA

Three major differences

- the range of δ is in the powerset 2^Q, so that its value is not a single element of Q, but a subset of it
- Since allow λ as the second argument of δ. This means that the nfa can make a transition without consuming an input symbol
- the set $\delta(q_i, a)$ may be empty, meaning that there is no transition defined for this specific situation

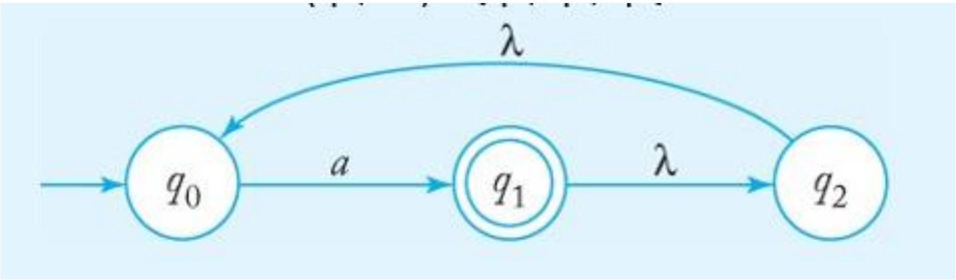
NFA Transition Graphs



Definition of NFA (2)

- For an nfa, the extended transition function is defined so that $\delta^*(q_i, w)$ contains q_j if and only if there is a walk in the transition graph from q_i to q_j labeled w. This holds for all $q_i, q_j \in Q$, and $w \in \Sigma^*$.

- Example



Definition of NFA (3)

- The language L accepted by an nfa $M = (Q, \Sigma, \delta, q0, F)$ is defined as the set of all strings accepted in the above sense.
- Formally, $L(M) = \{w \in \Sigma^* : \delta^*(q0, w) \cap F \neq \emptyset\}$.
- In words, the language consists of all strings w for which there is a walk labeled w from the initial vertex of the transition graph to some final vertex.

Why Nondeterminism

- When several alternatives are possible in deterministic algorithms, we choose one and follow it until it becomes clear whether or not it was best. If not, we retreat to the last decision point and explore the other choices.
- A nondeterministic algorithm that can make the best choice would be able to solve the problem without backtracking, but a deterministic one can simulate nondeterminism with some extra work.

Why Nondeterminism

- For this reason, nondeterministic machines can serve as models of search-and-backtrack algorithms.
- Nondeterminism is an effective mechanism for describing some complicated languages concisely.

Why Nondeterminism

- Notice that the definition of a grammar involves a nondeterministic element. In

$S \rightarrow aSb \mid \lambda$

- we can at any point choose either the first or the second production.
- This lets us specify many different strings using only two rules.

Why Nondeterminism

- Finally, there is a technical reason for introducing nondeterminism. As we will see, certain theoretical results are more easily established for nfa’s than for dfa’s.
- Our next major result indicates that there is no essential difference between these two types of automata.
- Consequently, allowing nondeterminism often simplifies formal arguments without affecting the generality of the conclusion.