DFA & NFA

Deterministic Finite Automata

定义

DFA is a finite-state machine that accepts or rejects a given string of symbols, by running through a state sequence uniquely determined by the string.

DFA是一个有限状态机器，它通过运行一个由字符串唯一确定的状态序列来接受或拒绝一个给定的符号字符串。

基本要素

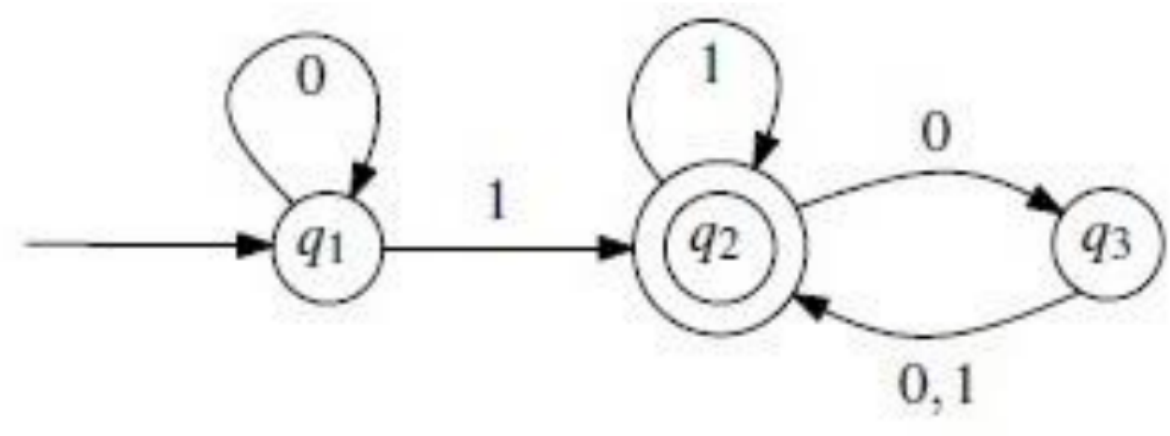
Start state(起始状态)

下面例子中的q1

Accept state(接受状态)

下面例子中的q2

举例：



Question.

Can the input string 0101010 be accepted by this machine?

Which kind of strings can be accepted by this machine?

答案：

1. 不能，这个字符串会导致该机器运行时停留在q3, 然而q2才是接受状态（最终必须停留在q2）
2. 以1结束的字符串或者有偶数个0接在最右边的1的后面的字符串可以被接受

用途

它的发明是为了识别一种特殊的正式语言，并有许多实际应用：

1. 词汇分析——从头到尾扫描输入程序，并将它划分为标识符、常量和关键字等标记，并删除注释和空白（指定编程语言的标记）。
2. 模型检查，对系统进行推理，以证明它们满足有用的性质。
3. 用于分析生物序列和文本序列的统计模型。

组成部分（5-tuple)

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

1. Q is a finite set of **states**,
2. Σ is a finite set of symbols, called the **alphabet** of the automaton,
3. $\delta: Q \times \Sigma \to Q$ is a function, called the **transition function**,
4. $q \in Q$ is called the **initial state**,
5. $F \subseteq Q$ is a set of **accepting/terminal states**.

State of a machine tells you something about the prefix that has been read so far. If the string is a member of the language of interest, the state reached when the whole string has been scanned will be an accepting state (a member of F).

机器的状态会告诉你一些关于迄今为止已经读取的前缀的信息。如果字符串是感兴趣语言的成员，则扫描整个字符串时达到的状态将是接受状态（F的成员）。

Transition function δ tells you how state should change when an additional letter is read by the DFA.

转换函数δ告诉您，当DFA读取一个附加字母时，状态应该如何改变。

Example of transition function

Initially the state is i and if the input word is $w = a_1 a_2 \ldots a_n$ then, as each letter is read, the state changes and we get $q_1, q_2 \ldots q_n$ defined by:

$$\begin{aligned} q_1 &= \delta(i, a_1) \\ q_2 &= \delta(q_1, a_2) \\ q_3 &= \delta(q_2, a_3) \\ &\vdots \\ q_n &= \delta(q_{n-1}, a_n) \end{aligned}$$

Further extended Map

In the above notation, extend the map $\delta : Q \times \Sigma \to Q$ to $\delta^* : Q \times \Sigma^* \to Q$ by defining:

$$\delta^*(q, \epsilon) = q \qquad \text{for all } q \in Q$$

$$\delta^*(q, wa) = \delta(\delta^*(q, w), a) \qquad \text{for all } q \in Q;\ w \in \Sigma^*;\ a \in \Sigma$$

Example

$\delta(q_0, a) = q_1$ *and* $\delta(q_1, b) = q_2$

Then

$\delta^*(q_0, ab) = q_2$

Proof Steps:

$$\left.\begin{aligned} &\delta^*(q_0, ab) = \delta(\delta^*(q_0, a), b) \\ &\delta^*(q_0, a) = \delta(q_0, a) = q_1 \end{aligned}\right\} \quad \delta^*(q_0, ab) = \delta(q_1, b) = q_2$$

Let $M = (Q, \Sigma, \delta, q, F)$ be a finite automaton and let $w = w_0 w_1 \ldots w_n$ be a string over . Define the sequence $q_0, q_1, \ldots, q_n$ of states, in the following way:

- $q_0 = q$,
- $q_{i+1} = \delta(q_i, w_{i+1})$, for $i = 0, 1, \ldots, n - 1$.

1. If $q_n \in F$, then we say that M accepts w.

2. If $q_n \notin F$, then we say that M rejects w.

Symbolic description of the example DFA

Automaton $M = (Q, \Sigma, \delta, q, F)$

Set of states $Q = \{i, t, r\}$, $\Sigma = \{0, 1\}$, $F = \{t\}$ and the transition function δ is given by

$$\delta(i, 0) = r \qquad \delta(i, 1) = t$$

$$\delta(t, 0) = t \qquad \delta(i, 1) = t$$

$$\delta(r, 0) = r \qquad \delta(r, 1) = r$$

It is simpler to describe a transition function by a table of values. In this example we have:

	0	1
i	*r*	*t*
t	*t*	*t*
r	*r*	*r*

练习题

Exercise

DFA M =($\{q_0, q_1, q_2\}$, $\{0, 1\}$, δ, q_0, $\{q_1\}$), where δ is given as:

$$\delta(q_0, 0) = q_0 \qquad \delta(q_0, 1) = q_1$$

$$\delta(q_1, 0) = q_0 \qquad \delta(q_1, 1) = q_2$$

$$\delta(q_2, 0) = q_2 \qquad \delta(q_2, 1) = q_1$$

Can the input word 110100 be accepted/recognized by this M ?

Exercise

How about the following infinite language. Can you give a DFA that accepts the words:

bad, baad, baaad, baaaad, ...?

Exercise

Given the symbolic description of a DFA, can you draw its corresponding diagram?

δ	a	b	c
Q_1	Q_2	Q_1	Q_3
Q_2	Q_4	Q_1	Q_2
Q_3	Q_2	Q_3	Q_1
Q_4	Q_4	Q_2	Q_3

Language defined by DFA(to build a DFA)

Suppose we have a DFA M. A word $w \in \Sigma^*$ is said to be accepted or recognized by M if $\delta^*(q_0, w) \in F$, otherwise it is said to be rejected. The set of all words accepted by M is called the language accepted by M and will be denoted by $\boldsymbol{L}(M)$. Thus

$$\boldsymbol{L}(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}$$

Any finite language is accepted by some DFA

A language $\mathbf{A}$ is called **regular**, if there exists a finite automaton M such that $\mathbf{A} = \boldsymbol{L}(M)$

Example

$A = \{w : \ w \text{ is a binary string containing an odd number of 1s}\}$

- The set of states is $Q = \{q_e, q_o\}$. If the finite automaton is in state q_e, then it has read an even number of 1s; if it is in state q_o, then it has read an odd number of 1s.
- The alphabet is $\Sigma = \{0, 1\}$.
- The start state is q_e, because at the start, the number of 1s read by the automaton is equal to 0, and 0 is even.
- The set F of accept states is $F = \{q_o\}$.
- The transition function δ is given by the following table:

	0	1
q_e	q_e	q_o
q_o	q_o	q_e

Exercise

Show that the language $\boldsymbol{L}$ is regular:

$$\boldsymbol{L} = \{awa: w \in \{a,b\}^* \}$$

Can you design a DFA that accepts this language?

Regular operations on languages

Let $\mathbf{A}$ and $\mathbf{B}$ be two languages over the same alphabet.

The **union** of $\mathbf{A}$ and $\mathbf{B}$ is defined as:

$$\mathbf{A} \cup \mathbf{B} = \{w : w \in \mathbf{A} \text{ or } w \in \mathbf{B}\}$$

The concatenation of $\mathbf{A}$ and $\mathbf{B}$ is defined as:

$$\mathbf{AB} = \{ww' : w \in \mathbf{A} \text{ and } w' \in \mathbf{B}\}$$

The star of $\mathbf{A}$ is defined as:

$$\mathbf{A}^* = \{u_1 u_2 \ldots u_k : k \geq 0 \text{ and } u_i \in \mathbf{A} \text{ for all } i = 1, 2, \ldots, k\}$$

Example of $\mathbf{A}^*$

Given two languages $\mathbf{A} = \{0, 01\}$ and $\mathbf{B} = \{1, 10\}$. Then

$$\mathbf{A} \cup \mathbf{B} = \{0, 01, 1, 10\}$$

$$\mathbf{AB} = \{01, 010, 011, 0110\}$$

$$\mathbf{A}^* = \{\epsilon, 0, 01, 00, 001, 010, 0101, 000, 0001, 00101, \ldots\}$$

The set of regular languages is closed under the union operation, i.e., if **A** and **B** are regular languages over the same alphabet **Σ**, then **A** ∪ **B** is also a regular language.

规则语言集在联合操作下是关闭的，即，如果A和B是在相同的字母表Σ上的规则语言，那么A∪B也是一种规则语言。

Nondeterministic Finite Automata

定义

A finite automata is **nondeterministic**, if the machine allows for several or no choices to exist for the next state on a given symbol.

一个有限自动机是不确定的，如果机器允许在给定符号上存在下一个状态的选择。

This NFA is in a state with multiple ways to proceed, e.g. state $q1$ has two transition paths with 1.

The machine splits into multiple copies of itself (threads):

1. Each copy proceeds with computation independently of others.

每个副本都要独立于其他副本进行计算。

2. NFA may be in a set of states, instead of a single state.

NFA可能处于一组状态，而不是一个单一的状态。

3. NFA follows all possible computation paths in parallel.

NFA并行地遵循所有可能的计算路径

4. If a copy is in a state and next input symbol doesn’ t appear on any outgoing edge from the state, then the copy dies or crashes.

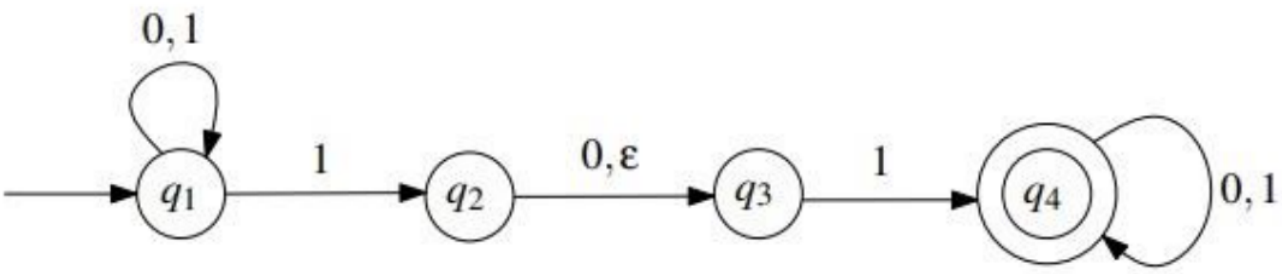
如果一个副本处于一种状态，并且下一个输入符号没有出现在该状态的任何输出边缘上，则该副本将死亡或崩溃。

The NFA accepts the input string, if any copy ends in an accept state after reading the entire string.

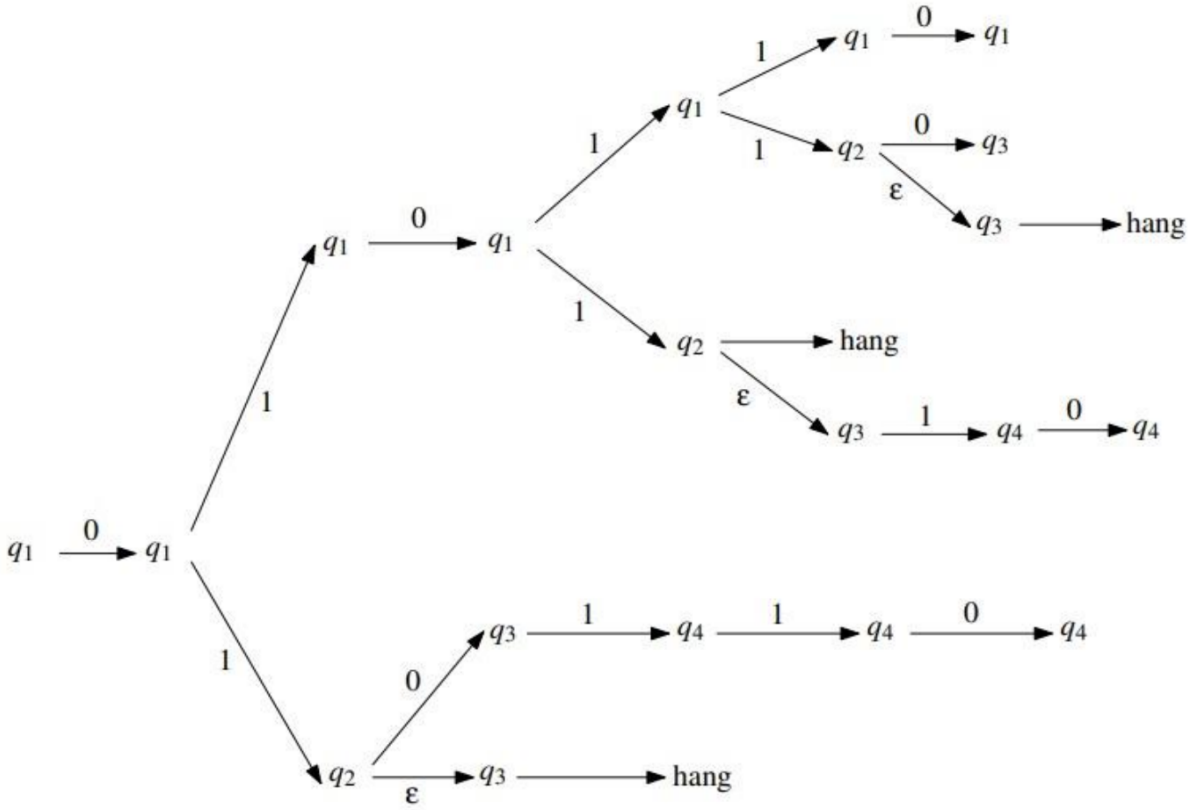
The NFA rejects the input string, if no copy ends in an accept state after reading the entire string.

如果任何副本在读取整个字符串后处于接受状态结束，则NFA接受输入字符串。

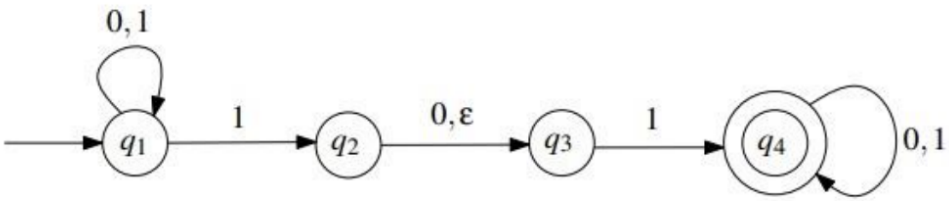
如果在读取整个字符串后没有复制结束时处于接受状态，则NFA将拒绝输入字符串。



What can this automaton do when it gets the string 010110 as input?



Question



How about 010?

组成要素

For any alphabet Σ, we define Σ_ϵ to be the set:

$$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$$

Recall the notion of a power set: For any set Q, the power set of Q, denoted by $P(Q)$, is the set of all subsets of Q:

$$P(Q) = \{R : R \subseteq Q\}$$

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

1. Q is a finite set of **states**,
2. Σ is a finite set of symbols, called the **alphabet** of the automaton,
3. $\delta : Q \times \Sigma_\epsilon \to P(Q)$ is a function, called the **transition function**,
4. $q \in Q$ is called the **initial/start state**,
5. $F \subseteq Q$ is a set of **accepting/terminal states**.

Let $M = (Q, \Sigma, \delta, q, F)$ be an NFA, and let $w \in \Sigma^*$. We say that M accepts w, if w can be written as $w = y_1 y_2 \ldots y_m$ where $y_i \in \Sigma_\epsilon$ for all i with $1 \leq i \leq m$, and there exists a sequence of states $r_1, r_2, \ldots, r_m$ in Q, such that:

- $r_0 = q$
- $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, 1, \ldots, m - 1$
- $r_m \in F$

Otherwise, we say that M rejects the string w.

DFA VS NFA

1. DFA has transition function $\delta: Q \times \Sigma \to Q$
2. NFA has transition function $\delta: Q \times \Sigma\boxtimes \to P(Q)$
3. Returns a set of states rather than a single state.
4. Allows for $\boxtimes$-transition because $\Sigma\boxtimes = \Sigma \cup \{\boxtimes\}$.
5. Note that every DFA is also an NFA.

Formal Definition of NFA

Extend the map δ to a map $Q \times \Sigma^* \rightarrow P(Q)$ by defining:

$$\delta(q, \epsilon) = \{q\} \quad \text{for all } q \in Q$$

$$\delta(q, wa) = \cup_{p \in \delta(q, w)} \delta(p, a) \quad \text{for all } q \in Q; w \in \Sigma^*; a \in \Sigma$$

Thus $\delta(q, w)$ is the set of all possible states that can arise when the input w is received in the state q. w is accepted provided that $\delta(q, w)$ contains an accepting state.

Accept/Rejected path（接受/拒绝路径）

假设，在DFA中，我们可以通过一个单词w的字母标记的转换从状态p到状态q。然后我们说状态p和q是由一个带有标签w的路径连接起来的。

If $w = abc$ and the 2 intermediate states are r_1 and r_2 we could write this as:

$$p \overset{a}{\rightarrow} r_1 \overset{b}{\rightarrow} r_2 \overset{c}{\rightarrow} q$$

In a NFA, if $\delta(p, a) = \{q, r\}$ we could write:

$$\{p\} \overset{a}{\rightarrow} \{q, r\}$$

and this would be an **accepting path** if any state on the RHS is an accepting state, otherwise it would be **rejecting path**.

Language accepted by NFA

Let $\mathrm{M} = (Q, \Sigma, \delta, q, F)$ be an NFA. The language $\mathrm{L(M)}$ accepted by M is defined as

$$\mathrm{L(M)} = \{w \in \Sigma^* : \mathrm{M} \text{ accepts } w\}.$$

Equivalence of DFAs and NFAs

如果两台机器（任何类型的）能够识别相同的语言，那么它们都是等价的

DFA是NFA的一种限制性形式：

1. 每个NFA都有一个等效的DFA
2. 我们可以将任意的NFA转换为接受相同语言的DFA
3. DFA与NFA具有相同的能力

DFA to NFA

The formal conversion of a DFA to an NFA is done as follows: Let M = $(Q, \Sigma, \delta, q, F)$ be a DFA. Recall that δ is a function $\delta : Q \times \Sigma \rightarrow Q$. We define the function $\delta' : Q \times \Sigma_\epsilon \rightarrow P(Q)$ as follows. For any $r \in Q$ and for any $a \in \Sigma_\epsilon$,

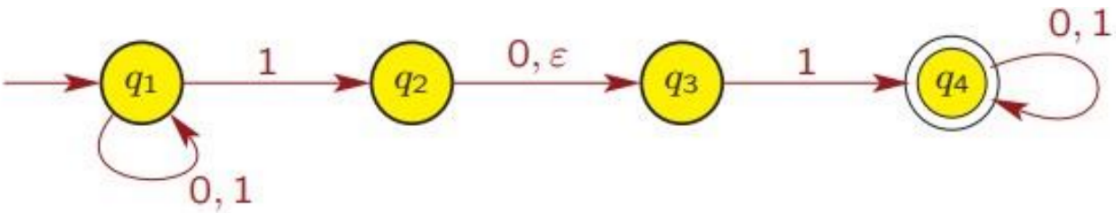
$$\delta'(r,a) = \begin{cases} \{\delta(r,a)\} & if\ a \neq \epsilon \\ \emptyset & if\ a = \epsilon \end{cases}$$

Then N = $(Q, \Sigma, \delta', q, F)$ is an NFA, whose behavior is exactly the same as that of the DFA M; the easiest way to see this is by observing that the state diagrams of M and N are equal. Therefore, we have $L(\mathrm{M}) = L(\mathrm{N})$.

NFA to DFA

Example

Convert NFA into equivalent DFA



NFA to DFA

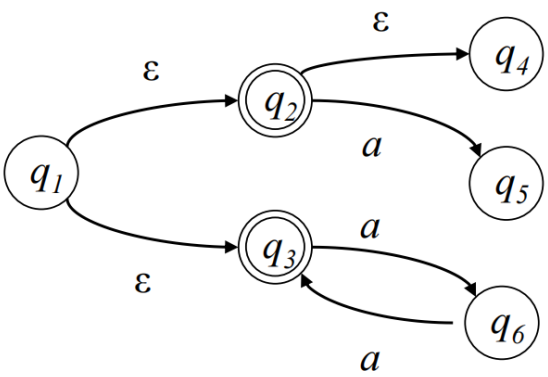
The ε-closure of a set of states $R \subseteq Q$:

$E(R) = \{ q \mid q$ can be reached from R by travelling over zero or more ε transitions $\}$.

Example

$$E(\{q_1, q_2\}) = \{q_1, q_2, q_3\}.$$

Question



$E(q_1) = ?$ $E(q_2) = ?$

NFA to DFA

Example

Consider the NFA M = (Q, Σ, δ, q, F), where $Q = \{1, 2, 3\}$, $\Sigma = \{\text{a}, \text{b}\}$, $q = 1$, $F = \{2\}$, and δ is given by the following table:

	a	b	ϵ
1	$\{3\}$	$\emptyset$	$\{2\}$
2	$\{1\}$	$\emptyset$	$\emptyset$
3	$\{2\}$	$\{2, 3\}$	$\emptyset$