Chapter 2: Finite Automaton

Dr. Yuan Yao

University of Nottingham Ningbo China (UNNC)

Learning Outcomes

Learning outcomes

At the conclusion of this chapter, the students are expected to be able to:

- Define the components of a deterministic finite automaton (DFA).
- State whether an input string is accepted by a DFA.
- Describe the language accepted by a DFA.
- Construct a DFA to accept a specific language.
- Show that a particular language is regular.
- Describe the differences between deterministic and nondeterministic finite automata (NFA).
- State whether an input string is accepted by an NFA.
- Construct an NFA to accept a specific language.
- Transform an arbitrary NFA to an equivalent DFA.

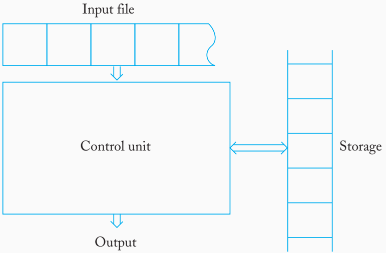
Introduction

Introduction to Finite Automata

有限状态自动机

In this lecture, we discuss our first simple automata, a finite state automaton (also known as a finite state accepter).

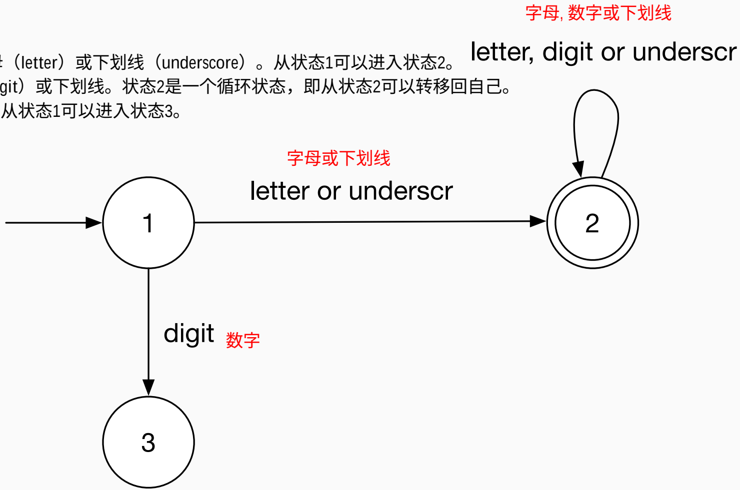
- An automaton with finite number of internal states and no other memory.
- Its output is restricted to either accepting or rejecting the input (string).
- It is useful for simple pattern recognition tasks.



Example: Finite Automata

Can you guess the set of strings accepted by the following Finite Automaton mean?

状态 1：初始状态，接受字母（letter）或下划线（underscore）。从状态1可以进入状态2。
状态 2：接受字母、数字（digit）或下划线。状态2是一个循环状态，即从状态2可以转移回自己。
状态 3：接受数字（digit）。从状态1可以进入状态3。



Deterministic Finite Automata

Deterministic Finite Automata

确定有限自动机

- A **deterministic finite automaton (DFA)** is defined by a 5-tuple:

$$M = (Q, \Sigma, \delta, q_0, F)$$

  where

  - Q is a finite set of **internal states**. 内部状态
  - Σ is a finite set of symbols called the **input alphabet**. 输入字母表
  - $\delta : Q \times \Sigma \to Q$ is a total function called the **transition function**. 转移函数
  - $q_0 \in Q$ is the **initial state**. 初始状态
  - $F \subseteq Q$ is the set of **final states**. 最终状态

简而言之，DFA是一个具有有限数量状态、明确转移规则的自动机模型。它的行为完全由输入和当前状态决定，没有任何歧义。

Example: DFA

- An example DFA:

Q

$$M = (\underline{\{q_0, q_1, q_2\}}, \{0, 1\}, \delta, q_0, \{q_1\})$$

where δ is given by:

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

- $\delta(q_0, 0) = q_0$：在状态 q_0 下，如果输入为 0，系统保持在状态 q_0。
- $\delta(q_0, 1) = q_1$：在状态 q_0 下，如果输入为 1，系统转移到状态 q_1。
- $\delta(q_1, 0) = q_0$：在状态 q_1 下，如果输入为 0，系统转移回状态 q_0。
- $\delta(q_1, 1) = q_2$：在状态 q_1 下，如果输入为 1，系统转移到状态 q_2。
- $\delta(q_2, 0) = q_2$：在状态 q_2 下，如果输入为 0，系统保持在状态 q_2。
- $\delta(q_2, 1) = q_1$：在状态 q_2 下，如果输入为 1，系统转移到状态 q_1。

- Is the following strings accepted by this DFA? 01
  - 10
  - 01
  - 110
  - 010

1. **字符串 “10”**：
   - 从 q_0，读取 “1” 进入 q_1。
   - 从 q_1，读取 “0” 进入 q_0。
   - 最终状态是 q_0，不是接受状态（接受状态是 q_1），所以 **不被接受**。
2. **字符串 “01”**：
   - 从 q_0，读取 “0” 进入 q_0。
   - 从 q_0，读取 “1” 进入 q_1。
   - 最终状态是 q_1，这是接受状态，所以 **被接受**。
3. **字符串 “110”**：
   - 从 q_0，读取 “1” 进入 q_1。
   - 从 q_1，读取 “1” 进入 q_2。
   - 从 q_2，读取 “0” 进入 q_2。
   - 最终状态是 q_2，不是接受状态，所以 **不被接受**。
4. **字符串 “010”**：
   - 从 q_0，读取 “0” 进入 q_0。
   - 从 q_0，读取 “1” 进入 q_1。
   - 从 q_1，读取 “0” 进入 q_0。
   - 最终状态是 q_0，不是接受状态，所以 **不被接受**。

Example: DFA

- An example DFA:

$$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$$

  where δ is given by:

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

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

  $\delta(q_2, 0) = q_2, \quad \delta(q_2, 1) = q_1.$

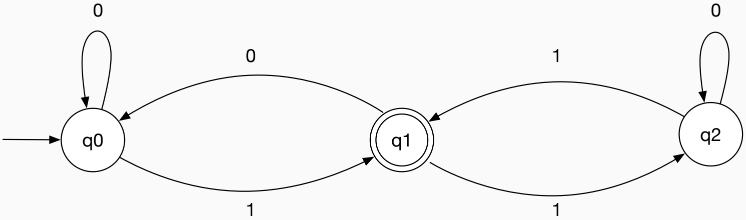
- Is the following strings accepted by this DFA?
  - 10: No ($q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_0$)
  - 01: Yes ($q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1$)
  - 110: No ($q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_2$)
  - 010: No ($q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_0$)

- A DFA can be represented as a transition graph.

$$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$$

where δ is given by:

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



JFLAP

- JFLAP is software for experimenting with formal languages topics including nondeterministic finite automata, nondeterministic pushdown automata, multi-tape Turing machines.
- The software is available on Moodle together with an brief introduction video by Prof. Amin Farjudian.

Deterministic Finite Automata

1. **为什么这些自动机被称为“确定性”？**

   - 转移函数是定义在状态集 Q 和输入字母表 Σ 上的**总函数**，即对于任何给定的状态和输入符号，自动机只有一个唯一的转移选择。换句话说，在任何状态下，给定输入符号，自动机只有一个明确的下一个状态，没有任何歧义或选择余地。

- Why these automaton are termed as “**Deterministic**”?
  - The transition functions are total function over Q and Σ, i.e., for any given state and input symbol, the automaton has one and only one option.
- We use DFAs to define a certain type of language, which we call *regular language*.
  - This is the first time we establish a connection between automata and languages (we will see more later in this module).

Processing Input with a DFA

- A DFA starts with the leftmost input symbol with its control in the initial state.
- Its next state is determined by the transition functions, the current state and the input symbol.
- The DFA continues processing input symbols until the end of the input string is reached or (unrecognised symbols occurs).
- The input string is accepted if the automaton is in a final state after the last symbol is processed. Otherwise, the string is rejected.

DFA（确定性有限自动机）处理输入的过程：

1.**DFA 从输入字符串的最左边开始**，并在初始状态下进行控制。也就是说，DFA 会从字符串的第一个符号开始处理。

2.**下一个状态的确定**：DFA 的下一个状态是由**转移函数**决定的。转移函数基于当前状态和输入符号的组合来确定下一个状态。

3.DFA 继续处理输入符号，直到达到输入字符串的末尾，或者遇到未识别的符号。换句话说，DFA 会处理每一个输入符号，直到它没有更多符号可以读取或者遇到无法识别的符号。

4.**接受条件**：如果在处理完所有输入符号后，DFA 最终处于一个接受状态（即最终状态在接受状态集合中），则该输入字符串被接受；否则，字符串会被拒绝。

总结：DFA 会从输入字符串的第一个符号开始，通过转移函数按照给定规则逐个处理字符，直到到达字符串的结尾。最终，DFA 会检查自己是否处于接受状态来决定是否接受该字符串。

Extended Transition Function: δ^*

- $\delta^* : Q \times \Sigma^* \to Q$ accepts **a string** as input and returns the state of the automaton after the string is processed.
- For example:
  - Suppose, we have $\delta(q_0, a) = q_1$ and $\delta(q_1, b) = q_2$.
  - Then, $\delta^*(q_0, ab) = q_2$
- Formally, $\delta^* : Q \times \Sigma^* \to Q$ can be recursively defined as follows:
  - For any given $q \in Q$, $w \in \Sigma^*$, $a \in \Sigma$:
  - $\delta^*(q, \lambda) = q$
  - $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$

The Language accepted by a DFA

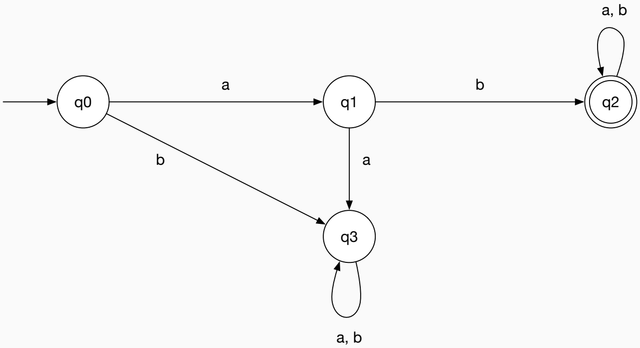
- The language accepted by a DFA M is the set of all strings accepted by M,
  - i.e., the set of all strings w such that $\delta^*(q_0, w)$ results in a final state

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

- What is the language accepted by the following DFA?

2. **DFA 图示**：

- 状态 q_0 是初始状态。
- 状态 q_1 和 q_2 都是接受状态，表示在这些状态结束的字符串会被接受。
- 转移规则如下：
  - 从 q_0 读取 “a” 或 “b” 转移到 q_1。
  - 从 q_1 读取 “a” 转移到 q_2，从 q_1 读取 “b” 回到 q_0。
  - 从 q_2 读取 “a” 或 “b” 循环回到 q_2。
  - 从 q_3 可以读取 “a” 或 “b”，并且保持在 q_3 中。



Exercise: DFA

- Give a simple and intuitive description of the language accepted by the following DFA:

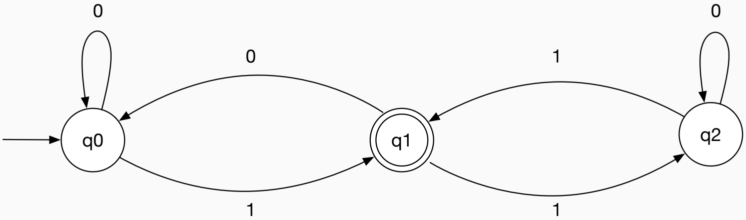


Table Representation

- While graphs are convenient for visualizing automata, other representations are also useful.
- For example, we can represent the transition function δ as a table.
- **Practice**: Assuming that the only final state is q_1, draw the graph for the DFA whose transition function is given by the table. Then, give a description of the language that it accepts.

	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

Table 1:

Exercise

- Find a DFA that accept all strings on $\Sigma = \{0, 1\}$, except those containing the substring 001

我们可以定义以下状态：

1. **状态** q_0：初始状态，表示尚未读取任何有问题的子串。
2. **状态** q_1：表示已经读取到一个 “0”。
3. **状态** q_2：表示已经读取到 “00”。
4. **状态** q_3：拒绝状态，表示已经读取到 “001” (即该字符串包含了子串 “001”)。

接下来，我们定义每个状态的转移规则：

- 从 q_0 开始：
  - 输入 “0” 转移到 q_1。
  - 输入 “1” 保持在 q_0。
- 从 q_1：
  - 输入 “0” 转移到 q_2。
  - 输入 “1” 转移回 q_0。
- 从 q_2：
  - 输入 “0” 保持在 q_2(继续保持状态，因为已经有了 “00”)。
  - 输入 “1” 转移到 q_3(表示已经读取到 “001”)。
- 从 q_3：
  - 无论输入什么，都会保持在 q_3，表示已经拒绝该字符串。

Notes on Finite Automata

- Finite automata are a simple special case of the general scheme introduced in the last chapter.
- Finite automata are characterised 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.
- 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.

Regular Language

Regular Language

- Finite automata accept a family of languages collectively known as **regular language**.
- A language L is called **regular** if and only if there exists some DFA M such that:

$$L = L(M)$$

- Therefore, to show that a language is regular, one must construct a DFA to accept it.
- **Regular languages are very useful in problems that involve scanning input strings in search of specific patterns.**
- Can you think of a language that is not regular?

Exercise: Regular Language

- Assume that $\Sigma = \{a, b\}$, and show that the following languages over Σ are regular, by constructing DFAs that accepts them:
  - $L_1 = \{(ab)^n a | n > 0\}$
  - $L_2 = \{w \in \Sigma^* | \ |w| \bmod 3 \neq 0\}$

Nondeterministic Finite Automaton (NFA)

Nondeterministic Finite Automaton (NFA) 非确定性有限自动机

- Nondeterminism means a choice of moves.
- Formally, a **nondeterministic finite automaton** is defined as a 5-tuple:

$$M = (Q, \Sigma, \delta, q_0, F)$$

where Q, Σ, q_0 and F are defined as for DFA, but

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

- Major differences between NFA and DFA:
    - In an NFA, the transition function returns **a subset** of Q rather than a single element in Q, e.g.,

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

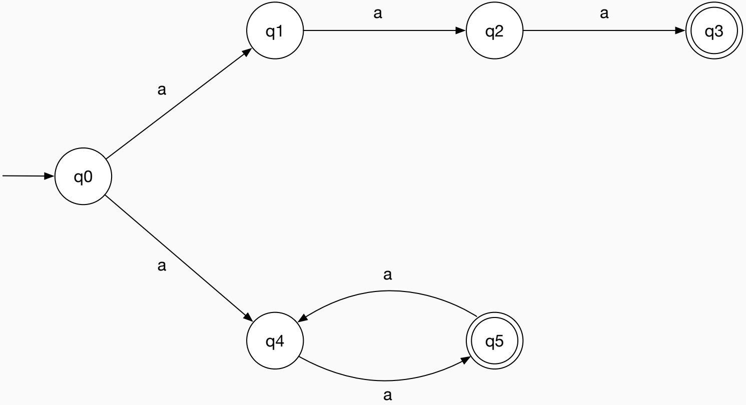
    - δ can be a **partial function**.
    - δ accepts λ as input, with which an NFA may change states without consuming input.

Example: NFA

- In this example, there are two transitions with label a from state q_0:

$$\delta(q_0, a) = \{q_1, q_4\}$$

这个例子帮助说明了 NFA 的一个重要特点：在某个状态下，同一个输入符号可以导致多个状态的转移，而不像 DFA 那样每次都有唯一的状态转移。



Example: NFA

$$M = (Q, \Sigma, \delta, q_0, F)$$

where

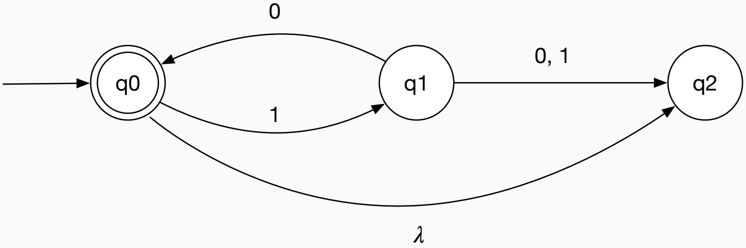
- **Q:** $\{q_0, q_1, q_2\}$
- **Σ:** $\{0, 1\}$
- δ: $\{\delta(q_0, 1) = \{q_1\}, \delta(q_0, \lambda) = \{q_2\}, \delta(q_1, 0) = \{q_0, q_2\}, \delta(q_1, 1) = q_2\}$
- F: $\{q_0\}$

Example: NFA

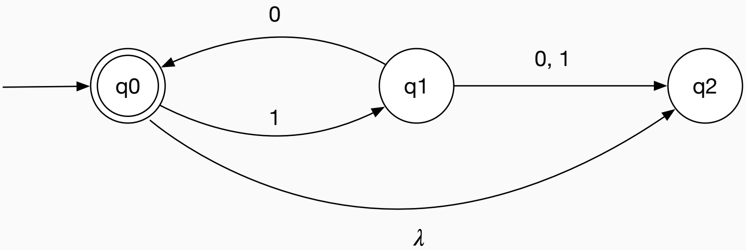
$$M = (Q, \Sigma, \delta, q_0, F)$$

where

- **Q:** $\{q_0, q_1, q_2\}$
- **Σ:** $\{0, 1\}$
- **δ:** $\{\delta(q_0, 1) = \{q_1\}, \delta(q_0, \lambda) = \{q_2\}, \delta(q_1, 0) = \{q_0, q_2\}, \delta(q_1, 1) = \{q_2\}\}$
- **F:** $\{q_0\}$



The Language Accepted by a NFA



Which of the followings are accepted by the given NFA?

- 10
- 1010
- 11
- λ

The Language Accepted by a NFA

- For a given NFA, the value of the extended transition function $\delta^*(q_i, w)$ is the set of all possible states for the control unit after processing w, having started in q_i
  - In other words, $\delta^*(q_i, w)$ contains q_j if and only if there is a walk in the transition graph from q_i to q_j labeled w.
- Sample values of δ^* for the NFA in the previous example:

$$\delta^*(q_0, 10) = \{q_0, q_2\}, \quad \delta^*(q_0, 101) = \{q_1\}$$

- A string w is accepted if $\delta^*(q_0, w)$ contains at least one final state.
- As is the case with DFA, the language accepted by an NFA M is the set of all accepted strings.

$$L(M) = \{w \in \Sigma^* \quad | \quad \delta^*(q_0, w) \cap F \neq \emptyset\}$$

- The machine in our previous example accepts $L = \{(10)^n : n \geq 0\}$

Equivalence of DFA and NFA

- **Question:** Does nondeterminism make it possible to accept languages that DFA cannot recognise? Is NFA more powerful than DFA?

Equivalence of DFA and NFA

- **Question:** Does nondeterminism make it possible to accept languages that DFA cannot recognise? Is NFA more powerful than DFA?
- **Definition:** Two finite automata M_1 and M_2 are said to be equivalent if

$$L(M_1) = L(M_2)$$

  that is, if they both accept the same language

Equivalence of DFA and NFA

- **Question:** Does nondeterminism make it possible to accept languages that DFA cannot recognise? Is NFA more powerful than DFA?
- **Definition:** Two finite automata M_1 and M_2 are said to be equivalent if

$$L(M_1) = L(M_2)$$

  that is, if they both accept the same language

  - First of all, it is clear that DFA is a restricted form of NFA, therefore, any language accepted by DFA is also accepted by some NFA.
  - How about the converse?

Equivalence of DFA and NFA

- The class of DFA and NFA are **equally** powerful: For every language accepted by some NFA there is a DFA that accept the same language.
- **Theorem:** Let $L(M_N)$ be the language accepted by a nondeterministic finite automaton $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$. Then there exists a deterministic finite automaton $M_D = (Q_D, \Sigma, \delta_D, q_0, F_D)$ such that

$$L(M_N) = L(M_D)$$

- How to prove it?

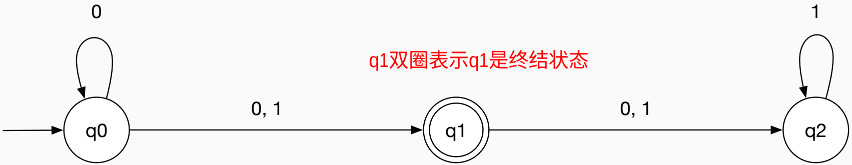
Equivalence of DFA and NFA

- Proof by construction, i.e., we could provide a way of converting any NFA into an equivalent DFA.
- Procedure: NFA-to-DFA
  1. Beginning with the initial state, define input transitions for the DFA as follows:
     - If the NFA input transition leads to a single state, replicate for the DFA.
     - If the NFA input transition leads to more than one state, create a new state in the DFA labeled $\{q_i, \ldots, q_j\}$, where $q_i, \ldots, q_j$ are all the states the NFA transition can lead to.
     - If the NFA input transition is not defined, the corresponding DFA transition should lead to a trap state.
  2. Repeat step 1 for all newly created DFA states, until no new states are created.
  3. Any DFA states containing an NFA final state in its label should be labeled as final.
  4. If the NFA accepts empty string, then the initial state should be labeled as final as well.

Example: NFA-to-DFA

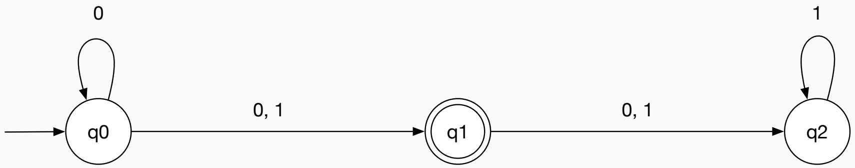
- $M_N = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta_N, q_0, \{q_1\})$ where δ is defined as follows:
  - $\delta(q_0, 0) = \{q_0, q_1\}$, $\delta(q_0, 1) = \{q_1\}$
  - $\delta(q_1, 0) = \{q_2\}$, $\delta(q_1, 1) = \{q_2\}$
  - $\delta(q_2, 0) = \emptyset$, $\delta(q_2, 1) = \{q_2\}$

- $\delta(q_2, 0) = \emptyset$ (q_2 在接收 0 时没有有效的转换)

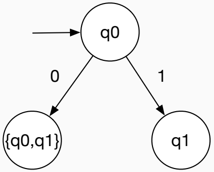


- **NFA 允许多条路径**，但 DFA 必须是**确定性的**（即每个状态在每个输入字符上只能有唯一的转换）。
- 使用**子集构造法（Subset Construction）**，我们可以将这个 NFA 转换成等价的 DFA：
  - 每个 **DFA 的状态**代表 NFA 中的**一个或多个状态的集合。**
  - 通过**计算 NFA 的 ε-闭包和状态转换**，构造 DFA 状态转换表。

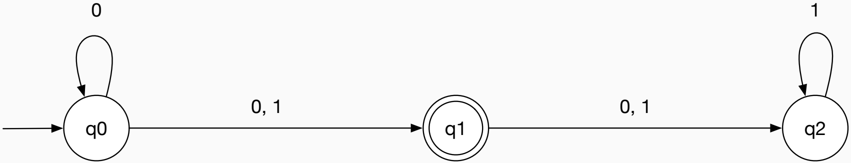
Example: NFA-to-DFA



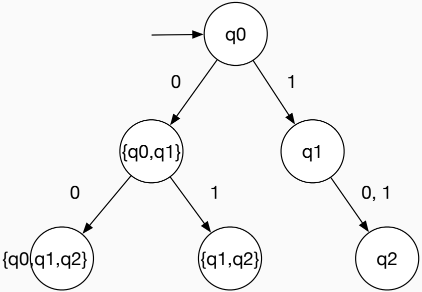
- We start from q_0, with 0, the transition leads to $\{q_0, q_1\}$, with 1, the transition leads to $\{q_1\}$



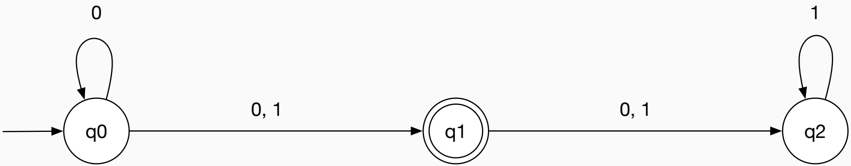
Example: NFA-to-DFA



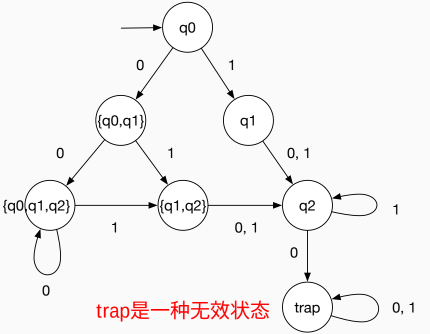
- $\delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1, q_2\}$, $\delta(q_0, 1) \cup \delta(q_1, 1) = \{q_1, q_2\}$
- $\delta(q_1, 0) = \{q_2\}$, $\delta(q_1, 1) = \{q_2\}$



Example: NFA-to-DFA

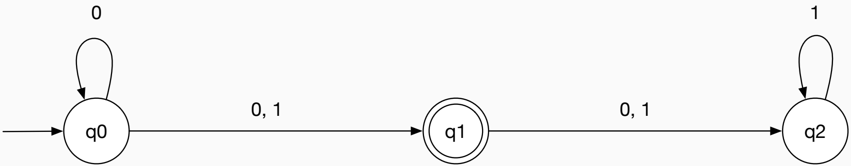


- $\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0) = \{q_0, q_1, q_2\}$, $\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1) = \{q_1, q_2\}$
- $\delta(q_1, 0) \cup \delta(q_2, 0) = \{q_2\}$, $\delta(q_1, 1) \cup \delta(q_2, 1) = \{q_2\}$
- $\delta(q_2, 0) = \emptyset$, $\delta(q_2, 1) = \{q_2\}$



trap是一种无效状态

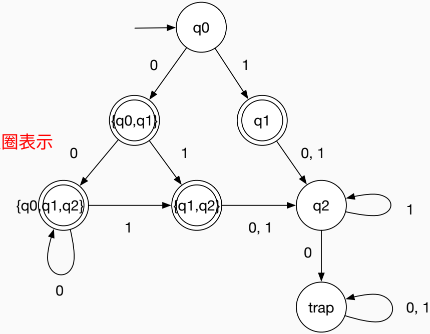
Example: NFA-to-DFA



- Final States:

  $\{q_1\}, \{q_0, q_1\}, \{q_0, q_1, q_2\}, \{q_1, q_2\}$

因为q1是NFA的终结状态，所以任何到达q1的输入都会被NFA接受，所以终结状态用双圈表示

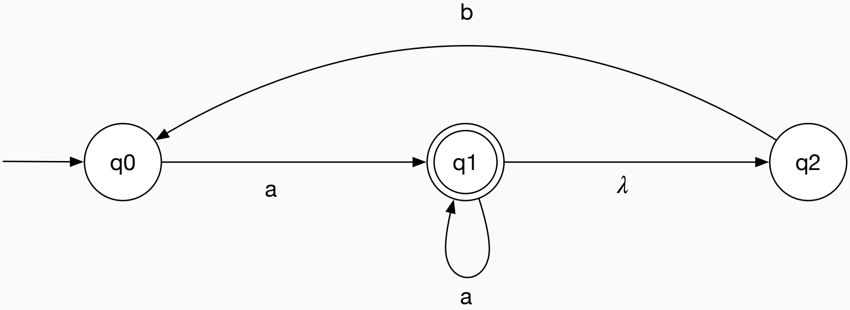


Exercise

- Convert the following NFA to an equivalent DFA.

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

where the transition graph is represented as the following one.



- 输入 **λ (ε-空转换) → 转移到** q_2(**即** q_1 **可直接跳到** q_2)