形式语言与自动机理论

有穷自动机

王春宇

计算机科学与技术学院
哈尔滨工业大学

有穷自动机

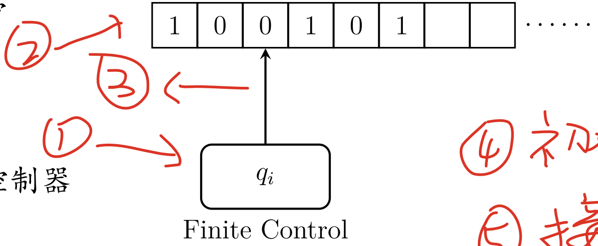
- 确定的有穷自动机
  - 形式定义
  - DFA 的设计举例
  - 扩展转移函数与 DFA 的语言
- 非确定有穷自动机
- 带有空转移的非确定有穷自动机

有穷状态系统

- 有限状态机: Moore Machine, Mealy Machine
- 数字电路设计
- 电脑游戏的 AI 设计
- 各种通讯协议: TCP, HTTP, Bluetooth, Wifi
- 文本搜索, 词法分析

确定的有穷自动机

- 一条输入带
- 一个读头
- 一个有穷控制器



④ 初始 status

⑤ 接受状态

例 1. 用有穷自动机识别 $\{w \in \{0,1\}^* \mid w$ 的长度$|w|$是偶数.$\}$

确定的有穷自动机的形式定义

定义

确定的有穷自动机(DFA, Deterministic Finite Automaton) A 为五元组

$$\boxed{A = (Q, \Sigma, \delta, q_0, F)}$$

❶ Q : 有穷状态集;

❷ Σ : 有穷输入符号集或字母表;

❸ $\delta : Q \times \Sigma \to Q$, 状态转移函数;

❹ $q_0 \in Q$: 初始状态;

❺ $F \subseteq Q$: 终结状态集或接受状态集.

$\delta(q, a) = p$

例 2. 请设计 DFA, 在任何由 0 和 1 构成的串中, 接受含有 01 子串的全部串.

❶ 未发现 01, 即使 0 都还没出现过; q_1

❷ 未发现 01, 但刚刚读入字符是 0; q_2

❸ 已经发现了 01. q_3

因此 DFA A 的可定义为:

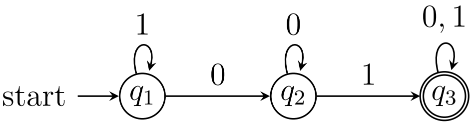
$$A = (\{q_1, q_2, q_3\}, \{0, 1\}, \delta, q_1, \{q_3\})$$

其中 δ 为:

$$\begin{array}{lll} \delta(q_1, 1) = q_1 & \delta(q_2, 1) = q_3 & \delta(q_3, 1) = q_3 \\ \delta(q_1, 0) = q_2 & \delta(q_2, 0) = q_2 & \delta(q_3, 0) = q_3 \end{array}$$

状态转移图

1. 每个状态 q 对应一个节点，用圆圈表示；
2. 状态转移 $\delta(q,a)=p$ 为一条从 q 到 p 且标记为字符 a 的有向边；
3. 开始状态 q_0 用一个标有 start 的箭头表示；
4. 接受状态的节点，用双圆圈表示.



状态转移表

❶ 每个状态 q 对应一行，每个字符 a 对应一列；

❷ 若有 $\delta(q,a)=p$，用第 q 行第 a 列中填入的 p 表示；

❸ 开始状态 q_0 前，标记箭头 $\rightarrow$ 表示；

❹ 接受状态 $q\in F$ 前，标记星号 $*$ 表示.

Σ

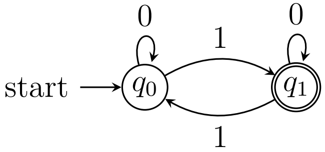
	Q	0	1
q_0	$\rightarrow q_1$	q_2	q_1
	q_2	q_2	q_3
F	$*q_3$	q_3	q_3

δ

典型问题

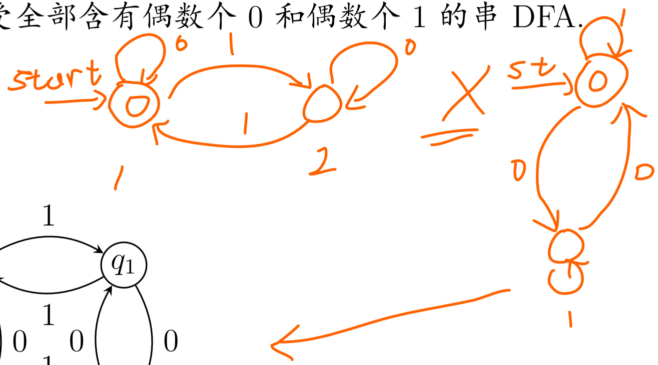
设计 DFA 使其接受且仅接受给定的语言 L.

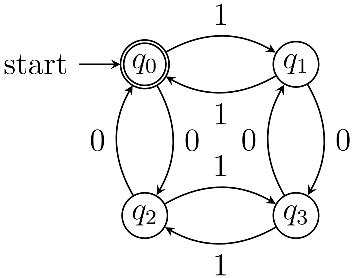
例 3. 若 $\Sigma = \{0, 1\}$, 给出接受全部含有奇数个 1 的串 DFA.



例 4. 若 $\Sigma = \{0,1\}$, 给出接受全部含有偶数个 0 和偶数个 1 的串 DFA.

迪卡尔积

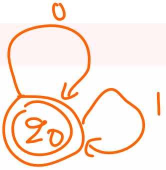




思考题

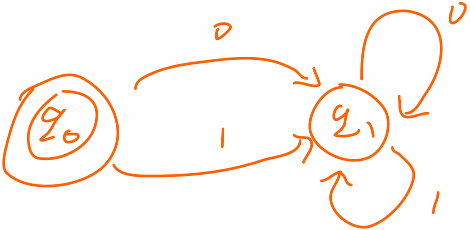
若 $\Sigma = \{0, 1\}$

❶ 如何设计接受 $\emptyset$ 的 DFA?



❷ 如何设计接受 Σ^* 的 DFA?

❸ 如何设计接受 $\{\varepsilon\}$ 的 DFA?



扩展转移函数

定义

delta height

扩展 δ 到字符串，定义扩展转移函数 $\hat{\delta}: Q \times \Sigma^* \to Q$ 为

recursive

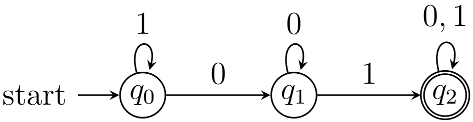
$$\hat{\delta}(q, w) = \begin{cases} q & w = \varepsilon \\ \delta(\hat{\delta}(q, x), a) & w = xa \end{cases}$$

其中 $a \in \Sigma$，$w, x \in \Sigma^*$.

那么，当 $w = a_0 a_1 \cdots a_n$，则有

$$\begin{aligned} \hat{\delta}(q, w) &= \delta(\hat{\delta}(q, a_0 a_1 \cdots a_{n-1}), a_n) \\ &= \delta(\delta(\hat{\delta}(q, a_0 a_1 \cdots a_{n-2}), a_{n-1}), a_n) = \cdots \\ &= \delta(\delta(\cdots \delta(\hat{\delta}(q, \varepsilon), a_0) \cdots, a_{n-1}), a_n) \end{aligned}$$

续例 2. 接受全部含有 01 子串的 DFA, $\hat{\delta}$ 处理串 0101 的过程.



$$
\begin{aligned}
\hat{\delta}(q_0, 0101) &= \delta(\hat{\delta}(q_0, 010), 1) \\
&= \delta(\delta(\hat{\delta}(q_0, 01), 0), 1) \\
&= \delta(\delta(\delta(\hat{\delta}(q_0, 0), 1), 0), 1) \\
&= \delta(\delta(\delta(\delta(\hat{\delta}(q_0, \varepsilon), 0), 1), 0), 1) \\
&= \delta(\delta(\delta(\delta(q_0, 0), 1), 0), 1) \\
&= \delta(\delta(\delta(q_1, 1), 0), 1) \\
&= \delta(\delta(q_2, 0), 1) = \delta(q_2, 1) = q_2
\end{aligned}
$$

思考题

❶ 扩展转移函数 $\hat{\delta}$ 必须从开始状态 q_0 处理字符串吗? $\hat{\delta}(q',w)$ ✓

❷ 对任意的串 w, $\hat{\delta}$ 能保证一定会跳转到某个状态吗? ✓

DFA: every input 都有跳转

例 5. 对任何状态 q 及字符串 x 和 y, 证明 $\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$.

证明: 对 y 使用归纳法. by induction

❶ 当 $y = \varepsilon$ 时

$$\begin{aligned} \hat{\delta}(\hat{\delta}(q, x), \varepsilon) &= \hat{\delta}(q, x) && \hat{\delta}\text{的定义} \\ &= \hat{\delta}(q, x\varepsilon) \end{aligned}$$

~~ats~~ assume $y=w$ correct

❷ 当 $y = wa$ 时

$$\begin{aligned} \hat{\delta}(q, xwa) &= \delta(\hat{\delta}(q, xw), a) && \hat{\delta}\text{和连接的定义} \\ &= \delta(\hat{\delta}(\hat{\delta}(q, x), w), a) && \text{归纳假设} \\ &= \hat{\delta}(\hat{\delta}(q, x), wa) \quad \square && \hat{\delta}\text{的定义} \end{aligned}$$

DFA 的语言与正则语言

定义

若 $D = (Q, \Sigma, \delta, q_0, F)$ 是一个*DFA*, 则 D 接受的语言为

$$\boxed{\mathbf{L}(D)} = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}.$$

定义

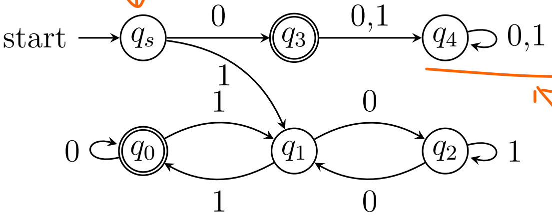
如果语言 L 是某个 *DFA* D 的语言, 即 $L = \mathbf{L}(D)$,
则称 L 是$\boxed{\text{正则语言}}$.

DFA可以识别的语言

- $\emptyset$, $\{\varepsilon\}$ 都是正则语言
- 若 Σ 是字母表, Σ^*, Σ^n 都是 Σ 上的正则语言

例 6. 设计 DFA 接受 $\{0,1\}$ 上的字符串 w, 且 w 是 3 的倍数的二进制表示.

$\{\varepsilon\}$



从0开头

~~ta~~ death status

	$\times 2$: 0	$\times 2+1$: 1
* $/3 = 0 \longrightarrow$ 0	0	1
1 $\longrightarrow$ 1	2	0
2	1	2