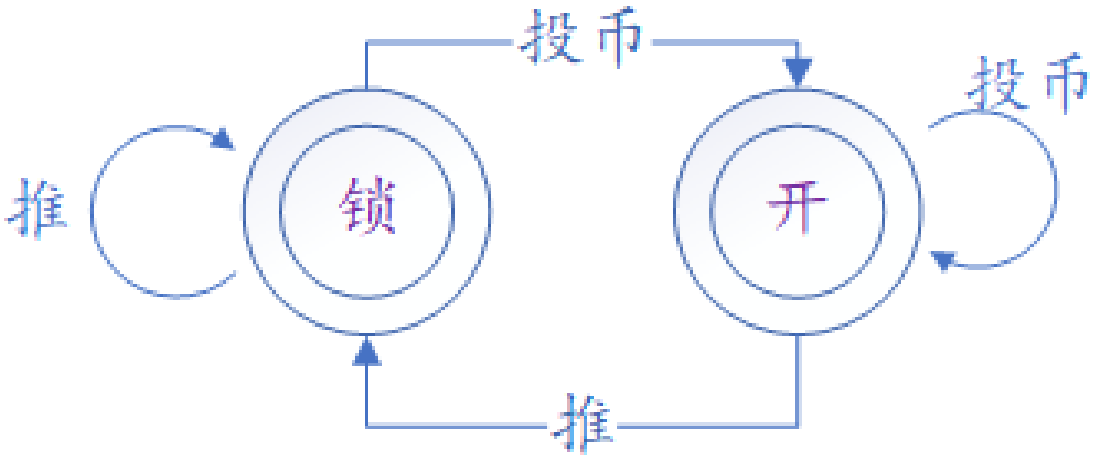
有限状态机

有限状态机(Finite-State Machine, FSM)是表示有限个状态以及在这些状态之间的转移和动作等行为的计算模型。



投入硬币,闸机打开。推闸机进人后,重新锁上。 闸机未锁时投币,或推锁着的闸机,闸机状态不变。

一个确定性有限状态机(Deterministic Finite Automata, DFA) 是一个五元组$(\Sigma, S, s_0, \delta, F)$,其中:

- Σ 是输入字符集(有限、非空).
- S是有限非空的状态集。
- s_0是初始状态,属于集合S.
- δ是状态转移函数 $\delta: S \times \Sigma \to S$.
- F是终止状态,也属于集合S.

正则表达式(Regular Expression, RE)由一组字符集Σ通过连接、并、克莱尼(Kleene)闭包等运算得到。RE用来描述正则语言(regular languages, RL)或正则集合。

正则表达式的正式定义如下：

1. 空集ϕ是RE,对应于空语言。
2. 空串ϵ是RE,没有实际字符,仅包含一个空字符。对应于空字符串语言。
3. 每一个字符$a \in \Sigma$是RE,是单字符串。对应单字符串语言。
4. 如果r和s是RE,那么(rs), $(r+s)$, 和(r^*)也是RE.对应$R \cdot S$, $R \cup S$和R^*语言。

克莱尼闭包:

{“a”,“bc”}*={ϵ,“a”,“bc”,“abc”,“bca”,“bcbc”,“aa”,“bcbcbc”,“abcbc”,…}

语言(languages)，无论是正则语言还是非正则语言，通常都用某种形式的二进制串来表示。

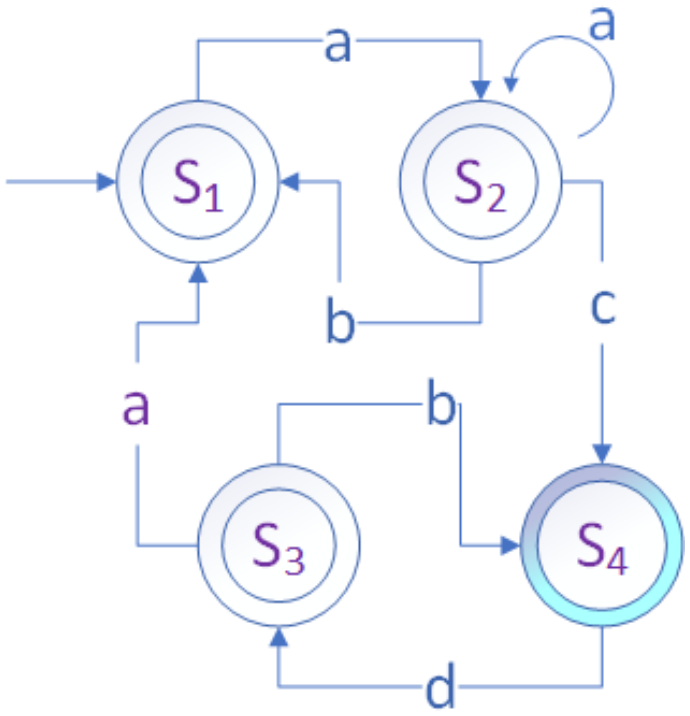
例如：第一个字符是0的语言，可以包括001, 0111, 01001等符合该语言的字符串。而110, 10001, 1等字符串不属于该语言。

不同的表达式有可能表达的是同一个语言。

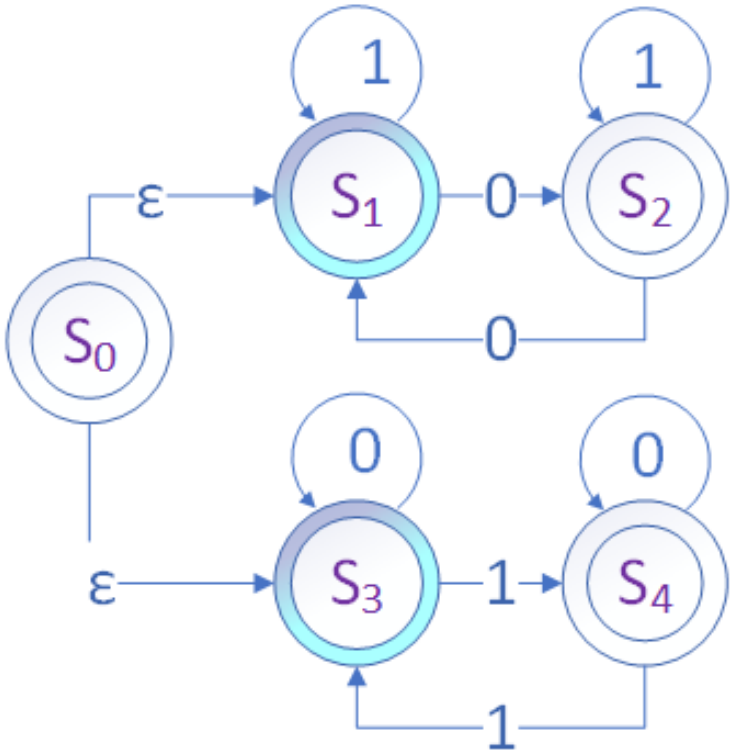
例: $((0^*)(1))((1)(0^*)(1)+0)^*$和 $0^*1(10^*1+0)^*$ 表达的语言相同。

有限状态机可以产生或接受(Accept)正则语言。

这里的字符串中的字符相当于之前例子中的“投币”和“推”。



这个有限状态机可以接受哪些字符串？ aaaaaac, abac, aaaacd, abacdaac.



这个有限状态机不能产生哪个字符串？ 0, 100, 1000, 01001, 101111011.

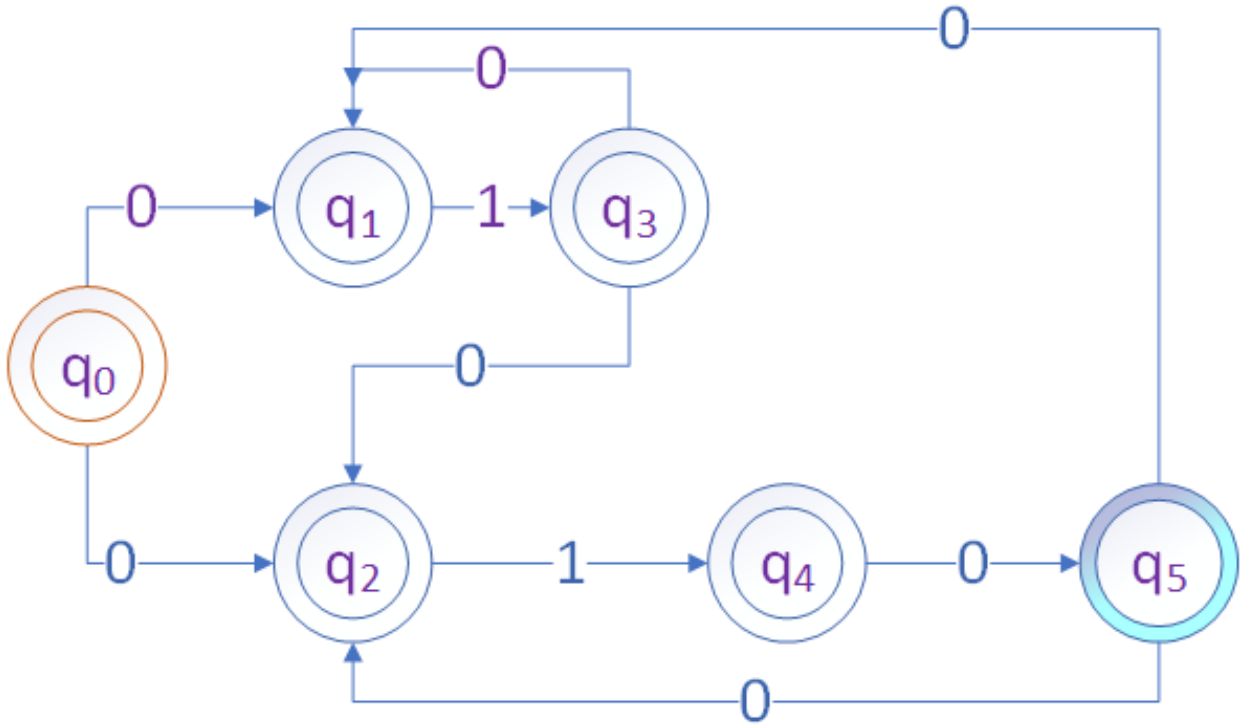
有两个办法来建立有限状态机与语言的联系。

- 反复从起始状态运行到终止状态，看看此有限状态机能生成哪些串？
- 给定一类字符串，看看有限状态机是否能按照字符串走到终止状态。

特别需要注意的是: 并非所有的语言都是正则语言。

例如: $L_1 = \{0^p 1^p \mid p \geq 1\}$和$L_2 = \{a^n b a^n \mid n \geq 1\}$不是正则语言。

通常,有两种类型的有限状态机:确定性(Deterministic)有限状态机和非确定性(Non-Deterministic)有限状态机(NDFAs) 。



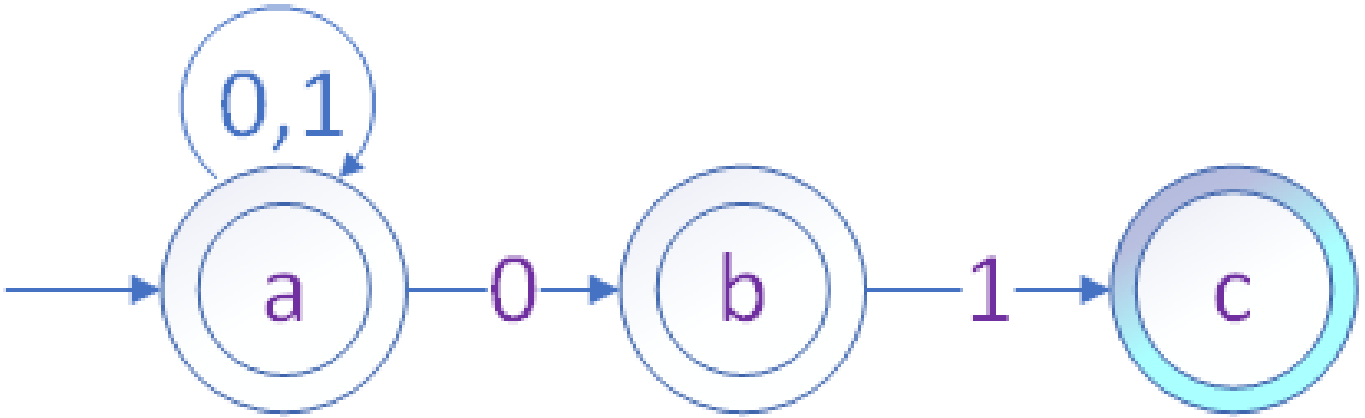
NDFA示例: q_3在输入为0时,可能进入q_1或q_2状态,不唯一。

事实上非确定性有限状态机与确定性有限状态机(DFAs)是等价的。

DFA可以看作NDFA的一个子集, 所以DFA转换成NDFA不成问题。

NDFA转换成DFA: 如果NDFA用到n个状态,转化后的DFA需要2^n个状态。

例如NDFA有3个状态， DFA就可以使用$\{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$来穷尽这些状态。



此NDFA转换成DFA后,{a}根据输入0,走向{a,b}.

在非确定性状态机中$\delta: S \times \Sigma \to \mathcal{P}(S)$, 例如$\delta$可能对于相同的输入进入不同的下一个状态。

有限状态机和图灵机的关系：

1. 有限状态机可以看作是一个受限的图灵机：
   - 只能读不能写
   - 只能往单方向走
2. 有限状态机也是图灵机的一部分。