算法与计算理论

课程内容

数据结构

概述	数组与广义表	图	外部排序
线性表	串	查找	
栈与队列	树	内部排序	

算法与计算理论

概述	贪心	……	计算模型
分治	回溯	……	可计算理论
动态规划	……	……	计算复杂性

Contents

本章内容

有限自动机

图灵机

教材

[S] 唐常杰等译, Sipser著, 计算理论导引(第3版), 机械工业.

参考资料:

[L] Lewis等著, 计算理论基础, 清华大学.

第1章 有限自动机

0. 引论--语言--什么是问题

1. 确定性有限自动机

1943年McCulloeh和Pitts建立了模拟神经网络的自动机

2. 非确定性有限自动机

(1959, 英国Dana Stewart Scott,Michael O. Rabin)

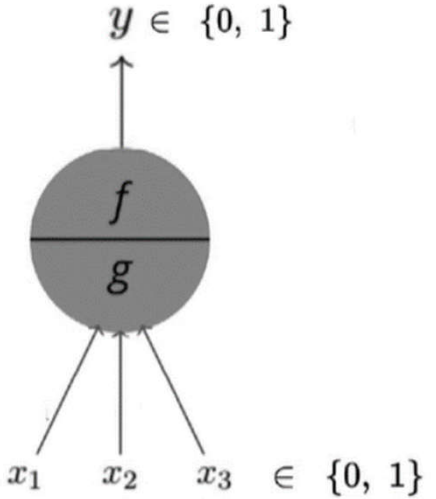
3. 正则表达式

（1951年，数学家Stephen Cole Kleene）

4. 正则语言的泵引理

1961年由Y. Bar-Hillel、 M. Perles和E. Shamir首次发表的

MP模型



第1章 有限自动机

如何表示全体问题?
如何表示算法?
问题与算法哪个多?

问题与决定性问题

判定性问题(Decision Prob): 只需回答是与否的问题

“某数是否是偶数”, “某串长度是否是2的幂次”

“某图是否连通”, “某图是否有k团”, “某数是否素数”

功能问题:

排序, 最大流, 最大团问题

本书只研究判定性问题:

1. 判定性问题能统一描述
2. 功能问题总能转化为判定性问题

例: 最大团问题如何转化为判定性问题?

“最大团”与“图是否有k团”

团: 完全子图, 即所有节点对都有边相连的子图.

两个问题目前都没有快速算法

若“最大团”有快速算法, 则“图是否有k团”也有:

对图G运行最大团算法, 得最大团的节点数m

若m≥k, 则有k团; 否则没有k团.

若“图是否有k团”有快速算法, 则“最大团”也有:

利用“图是否有k团”，二分搜索最大团节点数m.

1. left=0; right=n;

2.令k=(r-l)/2，执行“G是否有k团”.

有则令left=k+1;没有则令right=k-1继续第2步.

直到left>right

判定性问题与字符串集合

判定性问题(Decision Prob): 只需回答是与否的问题

“某数是否是偶数” ----------{ 以0结尾的01串 }

“串长度是否是2的幂次” ---{ 0^{2^n} : $n \geq 0$ }

“图是否连通” ----------{ <G> | G是连通图 }

其中<G>是图G编码成的字符串.

“图是否有k团” ----------{<G> | 图G有k团}

判定性问题与字符串集合

给定有限字母表Σ, 例如{0,1}

每个输入是一个01串, 任意01串都可以是输入

“判定性问题”——对应“字符串集合”

字符串与语言

字母表: 任意一个有限集. 常用记号Σ, Γ(读音:Gamma).

符号: 字母表中的元素： Σ={0, 1}， Γ={a, b, c, d, …, z, 空格}

字符串: 字母表中符号组成的有限序列

如： x=0011, y=love, z=math， 通俗地说即单词

串的长度： 序列的长度, 例: |x|=|y|=4

串的连接, 例: yz=lovemath

串的反转R, 例: $(z)^R$=htam

空词(空串): 记为ε, |ε|=0， 长度为0

子串： th是math的子串

语言与Σ^*

语言：给定字母表Σ，称Σ上一些字符串的集合为Σ上的语言.

例. 令字母表 $\Sigma = \{0,1\}$, Σ上的语言举例

A = {0,00,0000}, B = {0,1,01,000,001,...}

Σ^*={x| x是Σ上全体有限长度的字符串}

Σ上的任意语言A都是Σ^*的子集，即：$A \subseteq \Sigma^*$.

空语言：$\varnothing$

空串语言：$\{\varepsilon\}$

判定性问题与{0,1}上的语言一一对应

P(A)集合A的幂集. 例: P({a,b})={∅,{a},{b},{a,b}}

$P(\Sigma^*)=P(\{0,1\}^*)$: 全体判定性问题

Σ^*的标准序

Σ^*的标准序:长度按从小到大, 同长度按数从小到大排列

例1： $\Sigma_1 = \{0,1\}$

$\Sigma_1^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

例2： $\Sigma_2 = \{a, b, c, d, \ldots, z\}$

$\Sigma_2^* = \{\varepsilon, a, b, \ldots, z, aa, ab, \ldots, az, ba, bb, \ldots, bz, \ldots, zz, aaa, \ldots, zzz, \ldots, aaaa, \ldots\}$

因此Σ^*是可数集合，与N等势，其基数为$\aleph_0$

（$\aleph$ 是希伯来文中的Aleph，通常不发音 :)）

$\{0,1\}^N$

Σ={0, 1}，注意与Σ^*的区别， Σ^*是Σ上所有有限长度串集合

Σ^N： Σ上所有无限长度串记为Σ^N, 即$\{(x_i)_{i=1}^{\infty}: x_i \in \Sigma\}$

例: 0001100..., 010111...,

Σ^N中任一串$(x_i)_{i=1}^{\infty}$可以看做一个映射f:N$\rightarrow\Sigma_i$

其中N是自然数集, f(i) = x_i

串0001100...的映射

N	0	1	2	3	4	5	6	...
f	0	0	0	1	1	0	0	

定理 $\{0,1\}^N$不可数

$\{0,1\}^N$是全体无限长的01串

证明: 假设$\{0,1\}^N$可数, 即可以排成一列($f(i)$)

按下面方法在$\{0,1\}^N$中取一点 x ,

x的第i位与$f(i)$的第i位相反

n	f(n)
1	**1** 1 1 0 1 …
2	0 **0** 0 0 0 …
3	0 1 **1** 1 1 …
4	1 1 1 **0** 0 …
…	…
x	0 1 0 1 …

x与列表每个数不同

x不在列表中

所以$\{0,1\}^N$不可数.

Σ^N与R等势

其基数为$\aleph_1$

{0,1}上的语言与$\{0,1\}^N$一一对应

任取Σ上的语言A（Σ^*的一个子集），如下表示：

对Σ^*字典序下第i个字符串w,

若 $w \in A$ 令 $x_i = 1$; 若 $w \notin A$ 令 $x_i = 0$,

$(x_i)_{i=1}^{\infty} \in \Sigma^N$.

所以，Σ上的语言与Σ^N一一对应.

全体语言$P(\Sigma^*)$与Σ^N是等势的。

Σ^*	ε	0	1	00	01	10	11	000	001	…
A	×	0	×	00	01	×	×	000	001	…
g(A)	0	1	0	1	1	0	0	1	1	…

计算理论研究对象:语言

全体程序是$\{0,1\}^*$的子集, 至多可数
全体判定性问题与$\{0,1\}^N$等势, 不可数
程序可数, 问题不可数

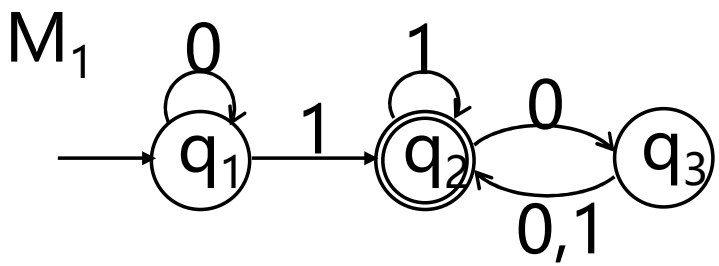
数学的研究对象有数, 函数, 函数空间等
计算理论的研究对象: 问题 即 语言 即 字符串集合

第1章 有限自动机

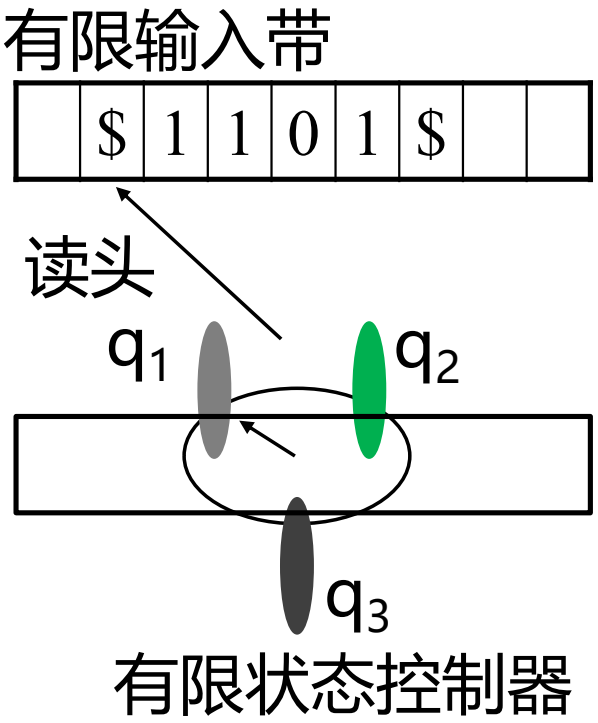
北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

0. 引论--语言--什么是问题
1. 确定有限自动机
   - 有限自动机定义
   - 有限自动机举例
   - 有限自动机的设计
   - 正则运算
2. 非确定有限自动机
3. 正则表达式
4. 正则语言的泵引理

有限自动机(Finite Automaton)



读头不能改写，且只能右移



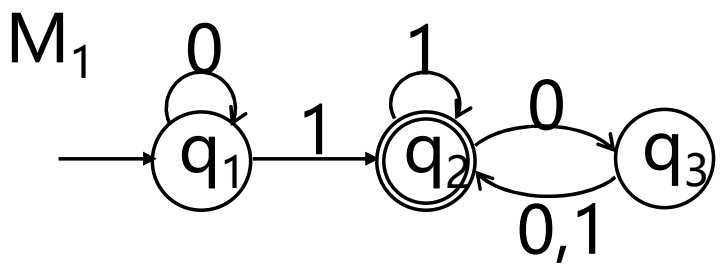
- 状态图

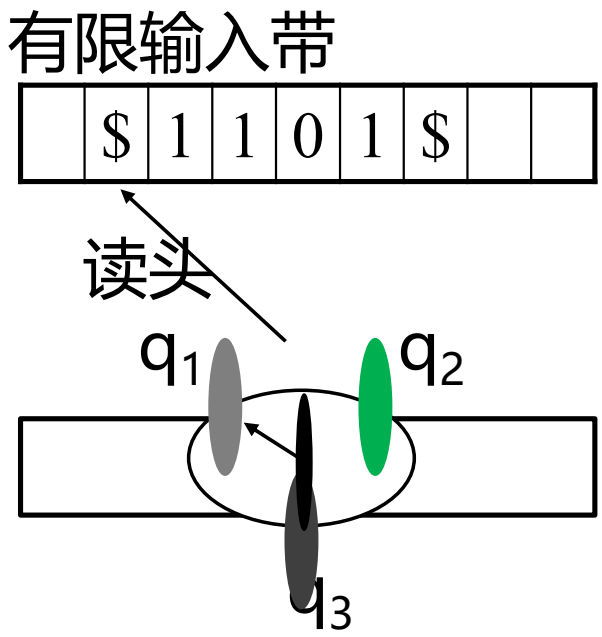
状态: q_1, q_2, q_3　　起始状态q_1

接受状态q_2　　转移: 箭头

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

有限自动机(Finite Automaton)





有限状态控制器

- **运行:**

  从起始状态开始沿转移箭头进行.

- **输出:**

  输入读完处于接受状态则接受, 否则拒绝.

- 接受: 1, 11, 100, 101, 1101, ...
- 拒绝: ε, 0, 10, 110, 1010, ...

有限自动机

定义: 有限自动机(Deterministic Finite Automata)

是一个5元组(Q,Σ,δ,s,F),

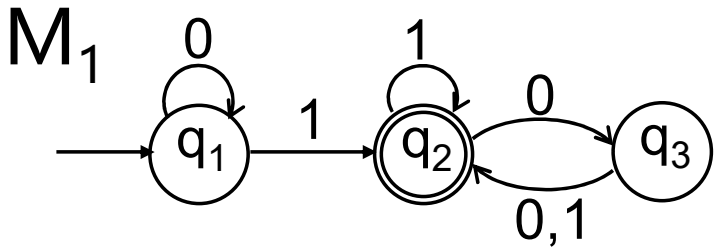
1) Q是有限集, 称为状态集;

2) Σ是有限集, 称为字母表;

3) δ: $Q\times\Sigma\rightarrow Q$是转移函数;

4) $s\in Q$是起始状态;

5) $F\subseteq Q$是接受状态集;



状态图等价于形式定义

δ读音: delta

$Q=\{q_1,q_2,q_3\}$, 状态集

$\Sigma=\{0,1\}$, 字母表

$s=q_1$, 起始状态

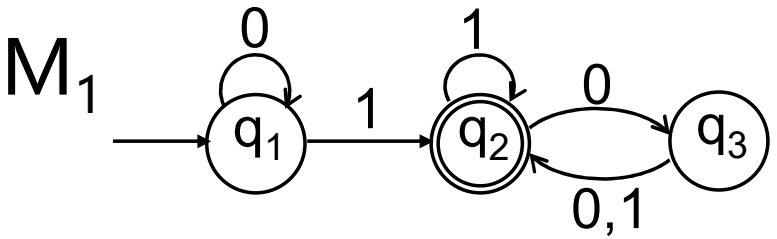
$F=\{q_2\}$接受状态集

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

DFA计算的形式定义

- 设M=(Q,Σ,δ,s,F)是一个DFA,
  $w=w_1w_2\ldots w_n$是字母表Σ上的一个字符串.
  若存在Q中的状态序列$r_0,r_1,\ldots,r_n$, 满足
  1) $r_0 = s$;
  2) $r_{i+1}=\delta(r_i, w_{i+1})$;
  3) $r_n \in F$

  则M接受w，记为$\delta(r_0,w)\in F$



$$s \xrightarrow{w_1} r_1 \xrightarrow{w_2} r_2 \cdots r_{n-1} \xrightarrow{w_n} r_n$$

有限自动机的语言:正则语言

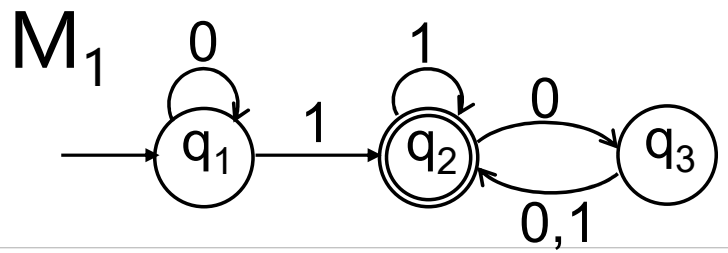
对有限自动机M, 若 $A = \{ w \in \Sigma^* \mid M接受w \}$,

即A是有限自动机M的语言, 记为L(M)=A, 也称M识别A.

注: M的语言唯一. M不识别任何其它语言.

正则语言：若存在DFA识别语言A, 则称A是正则语言.

等价：若两个有限自动机的语言相同，则称它们等价.



有限自动机的语言:正则语言

分析M_1:

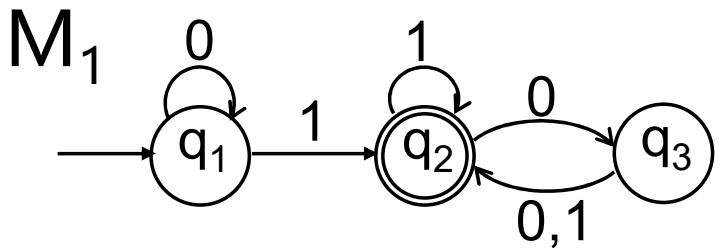
在任何状态, 读到1后一定会进入接受状态q_2.

在q_3状态下，读入0或1都进入接受状态

因此$L(M_1)=\{w \mid w\in\{0,1\}^*$, w至少含一个1,

且最后一个1后面含有偶数个0 $\}$

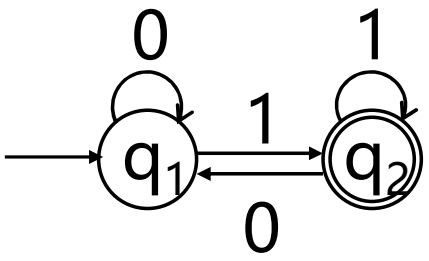
任何其它语言都不是M_1的语言.



有限自动机举例

$M_2=(\{q_1,q_2\},\{0,1\},\delta,s=q_1,F=\{q_2\})$

M_2



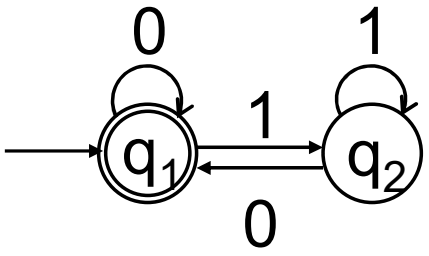
δ	0	1
q_1	q_1	q_2
q_2	q_1	q_2

$L(M_2)=\{w \mid w\in\{0,1\}^*$, w是以1结束的非空串 $\}$

有限自动机举例

$M_3=(\{q_1,q_2\},\{0,1\},\delta,s=q_1,F=\{q_1\})$

M_3

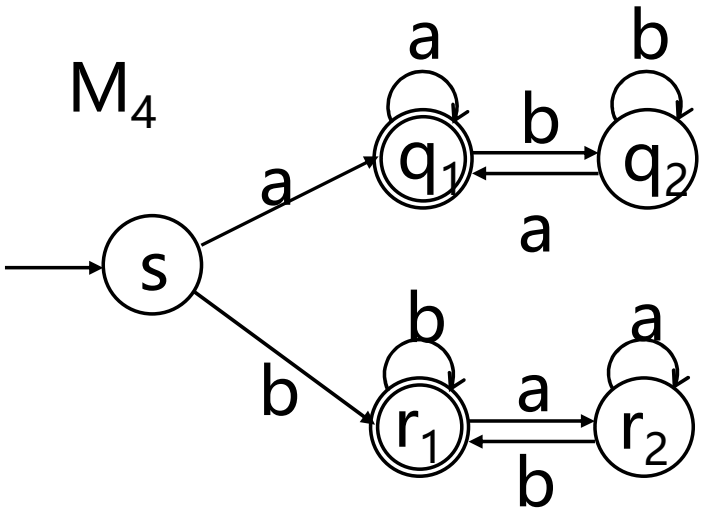


δ	0	1
q_1	q_1	q_2
q_2	q_1	q_2

$L(M_3)=\{w \mid w\in\{0,1\}^*$, w为空或以0结束 $\}$

有限自动机举例

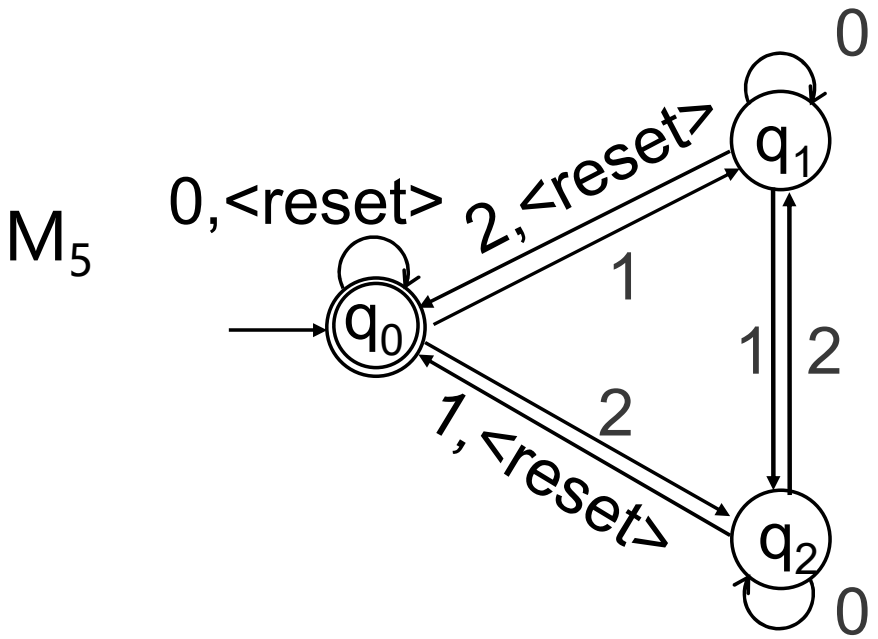
$M_4=(\{q_1,q_2,r_1,r_2\},\{a,b\}, \delta,s,F=\{q_1,r_1\}))$



$L(M_4)=\{w \mid w\in\{a,b\}^*$, w首尾字母相同的非空串 $\}$

有限自动机举例

$M_5=(\{q_0,q_1,q_2\}, \{0,1,2,\text{reset}\}, \delta_i, s=q_0, F=\{q_0\})$



$L(M_5)=\{w \mid w$满足在最后一个reset之后的所有数字之和为3的倍数$\}$

有限自动机举例

$\Sigma=\{0,1,2,reset\}$

$A_i=\{w \mid w$满足在最后一个reset之后的所有数字之和为i的倍数$\}$

$L(B_i)=A_i$. 设计自动机B_i。

$B_i=(Q_i, \Sigma, \delta_i, s=q_0, F=\{q_0\})$

$Q_i=\{q_0,q_1,q_2,\ldots q_{k-1}\}$

$\delta_i(q_k,0)= q_k$.

$\delta_i(q_k,1)= q_{(k+1) \bmod i}$.

$\delta_i(q_k,2)= q_{(k+2) \bmod i}$.

$\delta_i(q_k,reset)= q_0$.

有限自动机举例

$\Sigma=\{0,1,2,reset\}$

$A_i=\{w \mid w$满足在最后一个reset之后的所有数字之和为i的倍数 $\}$

$L(B_i)=A_i$. 设计自动机B_i。

$B_i=(Q_i, \Sigma, \delta_i, s=q_0, F=\{q_0\})$

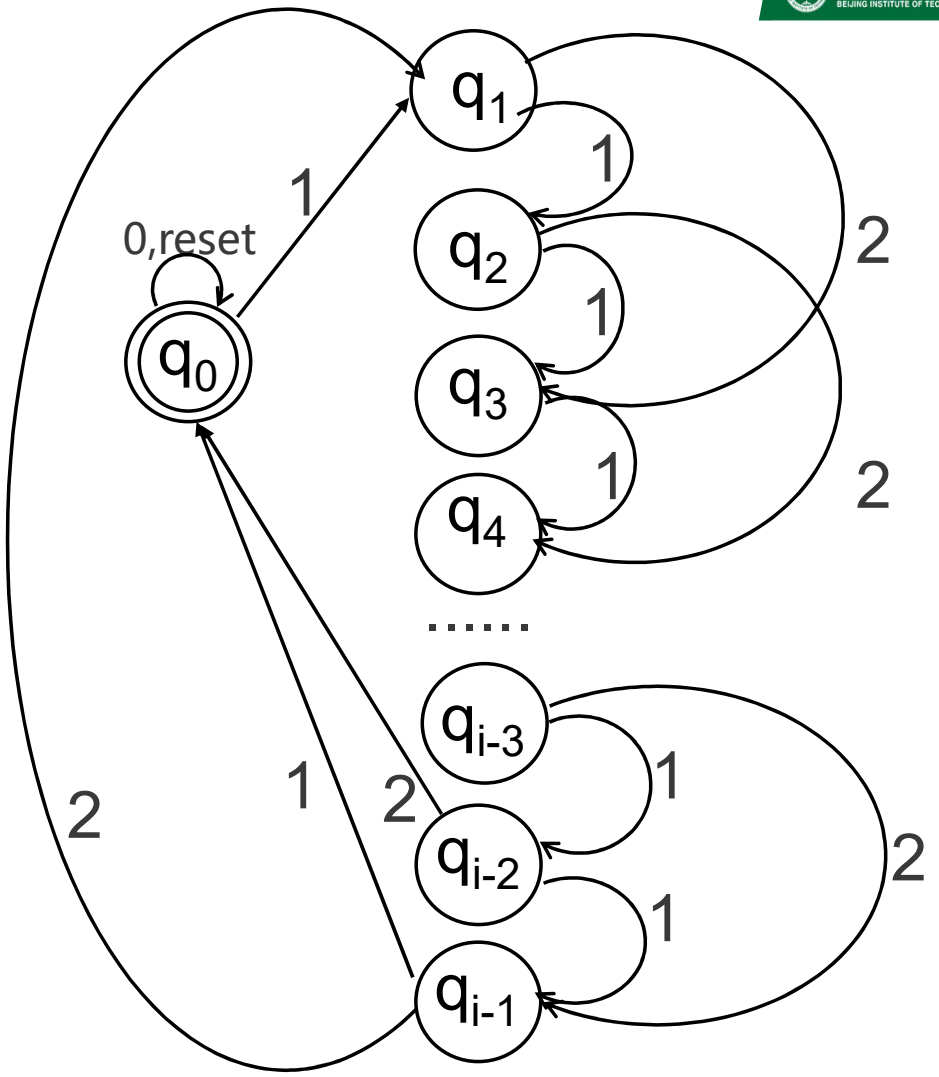
$Q_i=\{q_0,q_1,q_2,\ldots q_{i-1}\}$

$\delta_i(q_0,0)= q_k$

$\delta_i(q_j,1)= q_{(j+1) \bmod i}$.

$\delta_i(q_j,2)= q_{(j+2) \bmod i}$.

$\delta_i(q_0,reset)= q_0$.



有限自动机的设计(难点)

原则：自己即自动机

寻找需要记录的关键信息：

步骤1：确定状态

步骤2：确定转移

设计识别下列语言的DFA:

例1：{ w∈{0,1}* | w从1开始, 以0结束 }

例2：{ w∈{0,1}* | w含有子串1010 }

例3：{ w∈{0,1}* | w的倒数第2个符号是1 }

例4：{ 0^k | k是2或3的倍数 }

有限自动机的设计

例1：{ w∈{0,1}* | w从1开始, 以0结束 }

M_6

Σ={0,1}

步骤1：根据关键信息确定状态

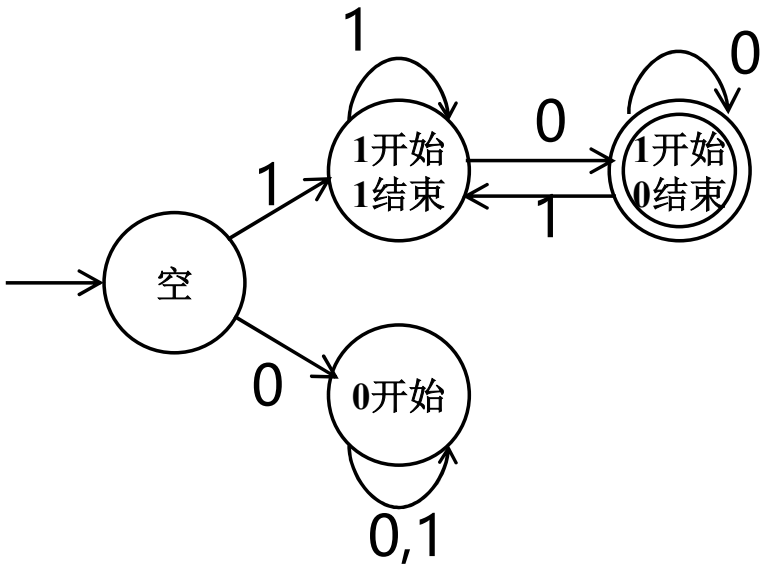
空-->不接受

以0开始-->不接受

以1开始以0结束-->接受

以1开始以1结束-->不接受

步骤2：确定转移函数

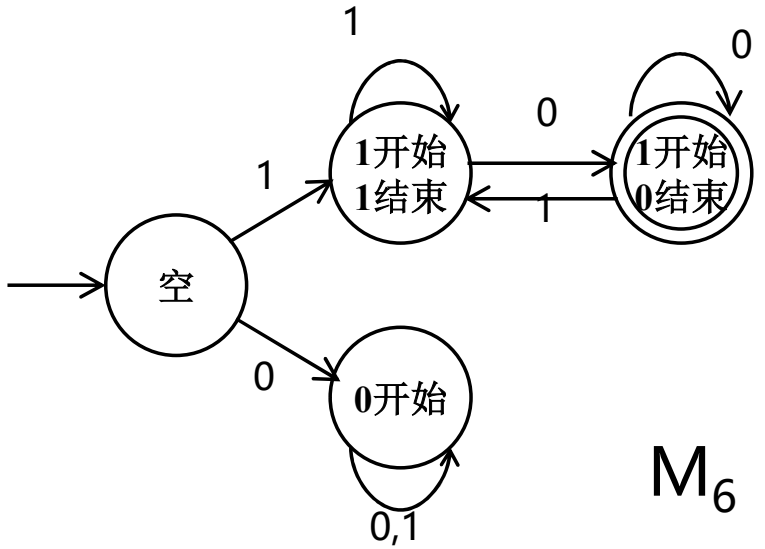


有限自动机的设计

例1：{ w∈{0,1}* | w从1开始, 以0结束 }

运行举例: 1100, 101

对应自动机算法:
1. 当前为初始状态
2. 当有输入, 根据转移函数转移当前状态
3. 若当前处于接受状态, 返回真, 否则返回假



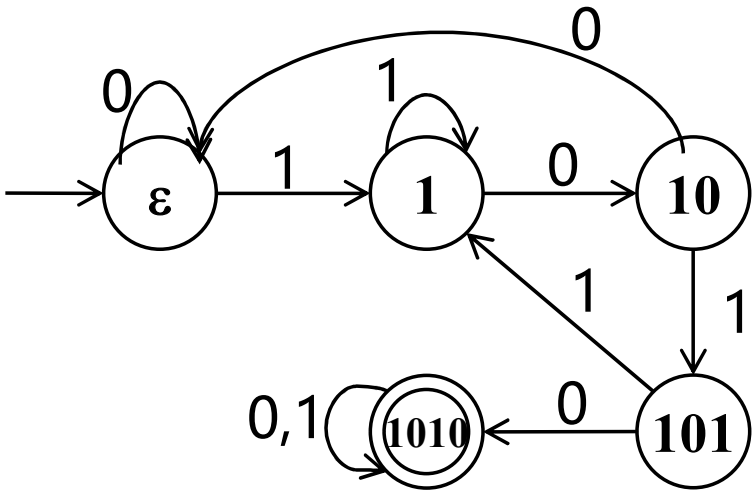
M_6

有限自动机的设计

例2：{ w∈{0,1}* | w含有子串1010 }

Σ={0,1}

关键信息: ε, 1, 10, 101, 1010



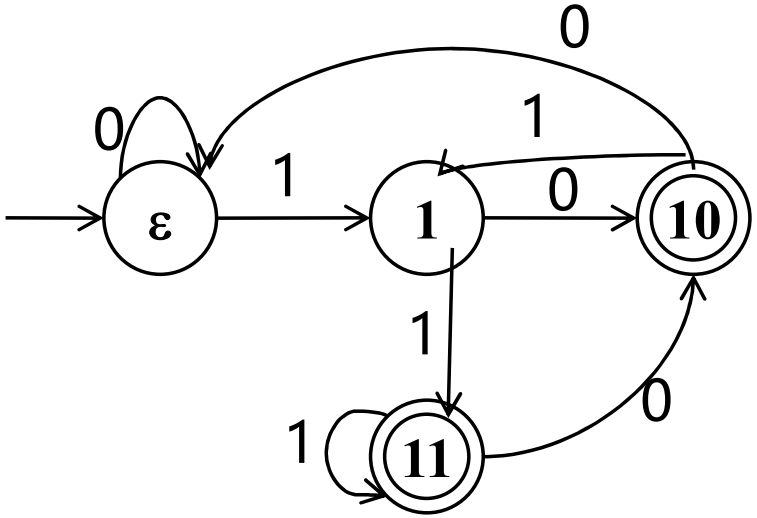
有限自动机的设计

例3：{ w∈{0,1}* | w倒数第2个符号是1 }

只需关注最后两个符号

Σ={0,1}, 关键信息: ε, 0, 1, 00, 01, 10, 11

关键信息改进: ε, 1, 10, 11



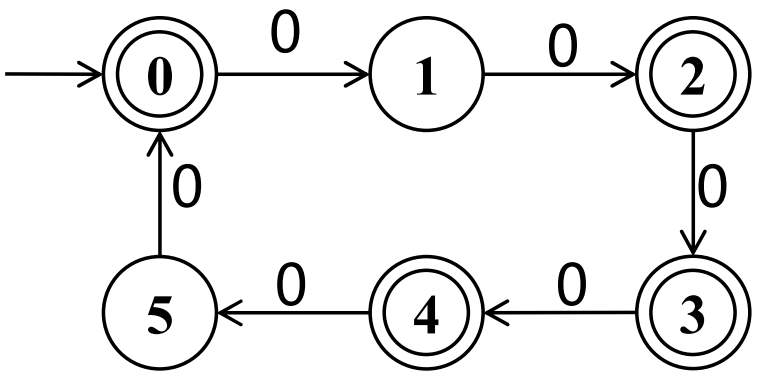
有限自动机的设计

例4：{ 0^k | k是2或3的倍数 }

Σ={0}

关键信息: ε,0^1,0^2,0^3,0^4,0^5.

记为: 0,1,2,3,4,5

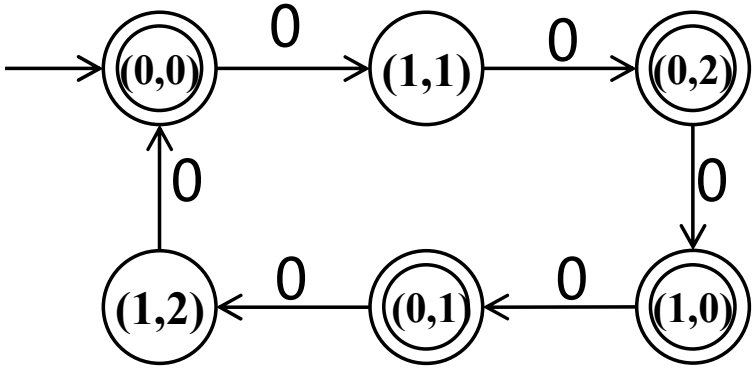


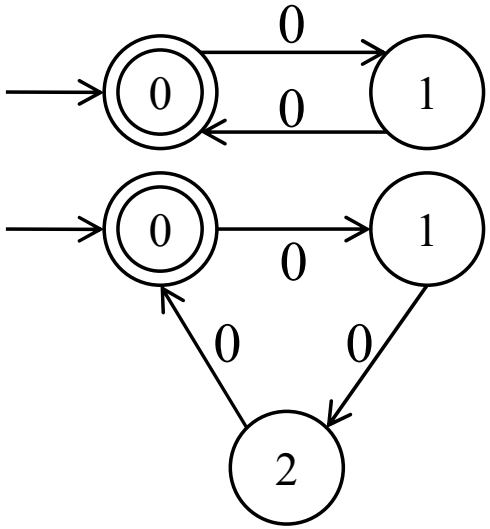
有限自动机的设计

例4：{ 0^k | k是2或3的倍数 }

Σ={0}, 关键信息: ε,0^1,0^2,0^3,0^4,0^5,

记为: 0,1,2,3,4,5 或 (0,0), (1,1), (0,2), (1,0), (0,1), (1,2)





{0^k|k是2或3的倍数} = {0^k|k是2倍数} ∪ {0^k|k是3的倍数}

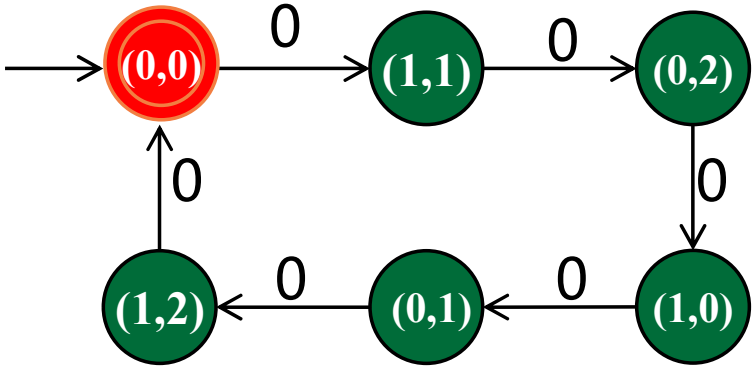
{0^k|k是2和3的倍数} = {0^k|k是2倍数} ∩ {0^k|k是3的倍数}?

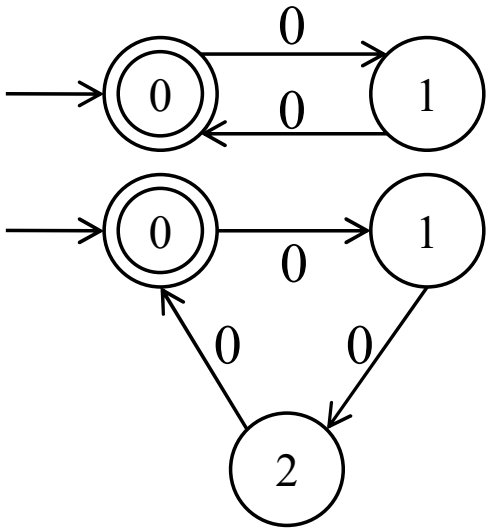
有限自动机的设计

{ 0^k | k是2和3的倍数 }

Σ={0}, 关键信息: ε,0^1,0^2,0^3,0^4,0^5,

记为: 0,1,2,3,4,5 或 (0,0), (1,1), (0,2), (1,0), (0,1), (1,2)





{0^k|k是2和3的倍数} = {0^k|k是2倍数} ∩ {0^k|k是3的倍数}

课堂练习

试着完成下列的DFA模型：

1. $\{ w \in \{a,b\}^* \mid$ w中的每一个a的前面都是一个b$\}$
2. $\{ w \in \{a,b\}^* \mid$ w含有子串abab$\}$
3. $\{ w \in \{a,b\}^* \mid$ w不包含子串aa和bb$\}$
4. $\{ w \in \{a,b\}^* \mid$ w有奇数个a和偶数个b$\}$
5. $\{ w \in \{a,b\}^* \mid$ w含有子串ab和ba$\}$

正则运算

设A, B都是Σ上的正则语言, 正则运算:

并 $A \cup B = \{x \mid x \in A \text{ 或 } x \in B\}$

连接 $AB = \{xy \mid x \in A, y \in B\}$

星号 $A^* = A^0 \cup A^1 \cup A^2 \cup A^3 \cup \ldots$

$= \{\varepsilon\} \cup A \cup AA \cup AAA \cup \ldots$

$= \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ 且每一个 } x_i \in A\}$

补 $A^c = \{x \mid x \in \Sigma^* - A\}$

定理：正则语言对于正则运算是封闭的。

正则运算举例

设字母表Σ由标准的26个字母组成

A={good, bad}, B={boy, girl}, 则

A∪B={ good, bad, boy, girl }

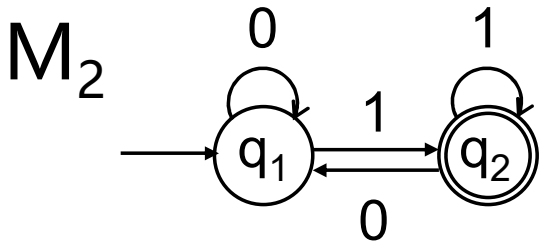
A∘B={ goodboy, goodgirl, badboy, badgirl }

A^*={ε, good, bad, goodgood, goodbad, … }

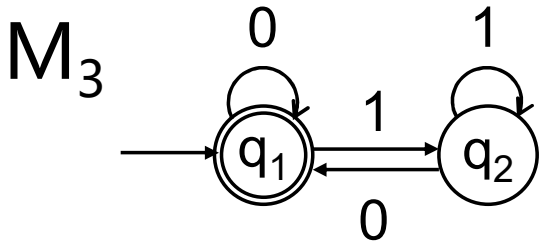
正则语言对补运算封闭

定理：正则语言对补运算封闭

证明思路：



$L(M_2)=\{w \mid w\in\{0,1\}^*, w是以1结束的非空串\}$



$L(M_3)=\{w \mid w\in\{0,1\}^*, w为空或以0结束\}$

正则语言对补运算封闭

定理：正则语言对补运算封闭

证明：构造性证明

假设正则语言L=L(M);

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

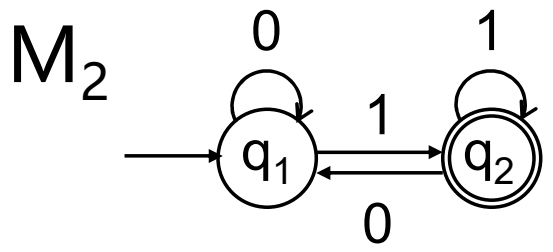
$M' = (Q,\Sigma,\delta,q_0,F')$，令$F'=Q-F$

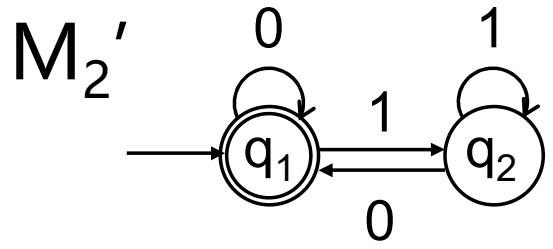
$\forall x,\ x\in L(M) \Leftrightarrow \delta(q_0,x)\in F$

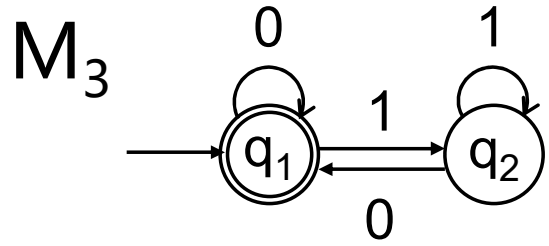
$\Leftrightarrow \delta(q_0,x)\notin F' \Leftrightarrow x\notin L(M')$

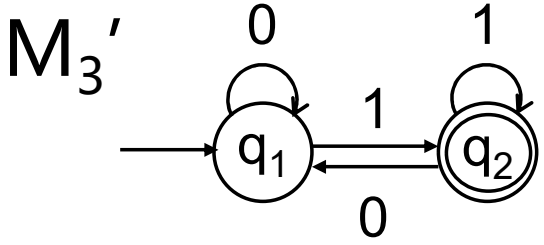
所以$L(M') = \Sigma^* - L(M) = L(M)^C$.

证毕.







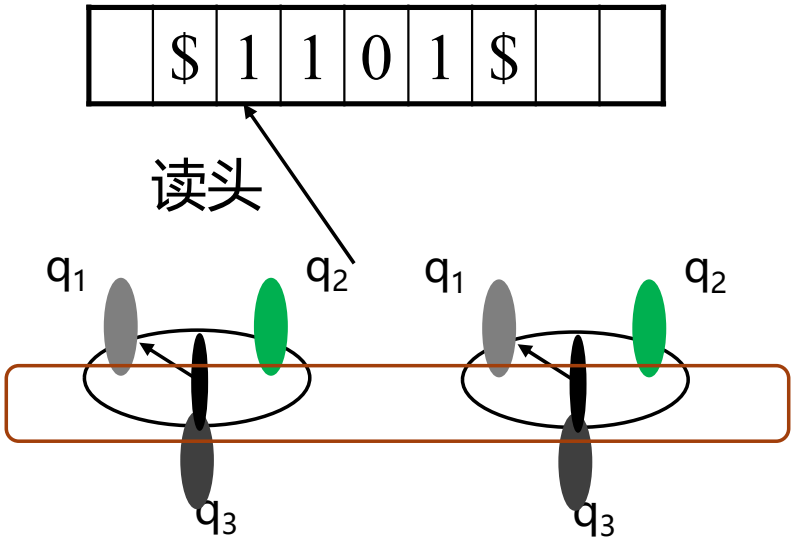


正则语言对并运算封闭

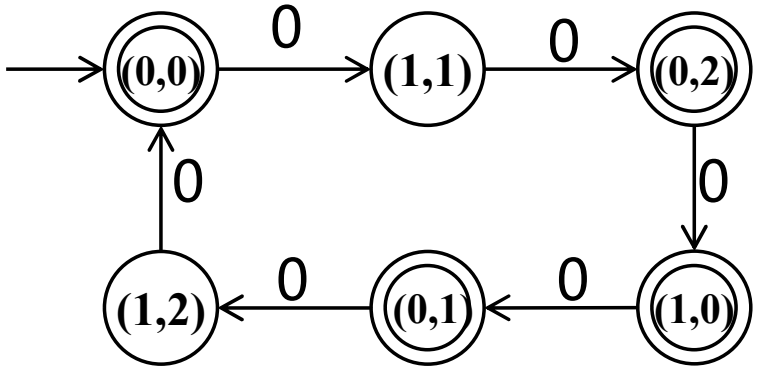
定理: 设A, B都是Σ上的正则语言, 则A∪B也是正则语言.

证明思路：构造性证明

构造一个有限自动机，同步更新两个有限自动机



有限状态控制器



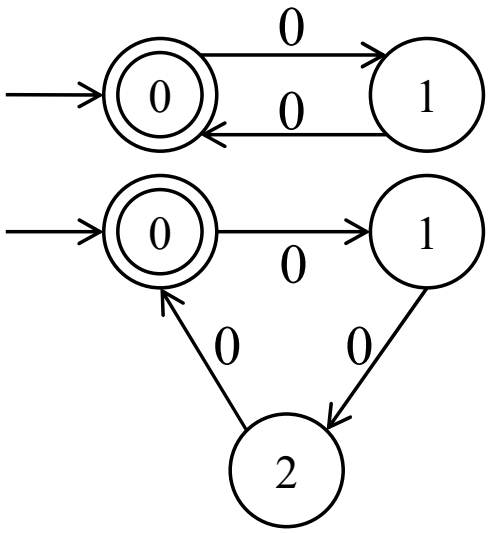
{ 0^k | k是2或3的倍数 }

正则语言对并运算封闭

定理: 设A, B都是Σ上的正则语言, 则A∪B也是正则语言.

证明思路：构造性证明

构造一个有限自动机，同步更新两个有限自动机



{ 0^k | k是2或3的倍数 }

正则语言的并是正则语言

定理: 设A, B都是Σ上的正则语言, 则$A \cup B$也是正则语言.

证明: 设$M_1=(Q_1,\Sigma,\delta_1,s_1,F_1)$和$M_2=(Q_2,\Sigma,\delta_2,s_2,F_2)$是有限自动机,

且 $L(M_1)=A$, $L(M_2)=B$.

令$M=(Q,\Sigma,\delta,s,F)$

其中 $Q=Q_1\times Q_2$, $s=(s_1,s_2)$, $F = F_1\times Q_2 \cup Q_1\times F_2$,

δ: $Q\times\Sigma\to Q$, $\forall a\in\Sigma$, $r_1\in Q_1$, $r_2\in Q_2$,

$\delta((r_1,r_2), a) = (\delta_1(r_1,a), \delta(r_2,a))$,

即第1个分量按M_1的转移函数变化，第2个分量按M_2的转移函数变化。 则 $\forall x\,(x\in L(M) \leftrightarrow x\in A \cup B)$

即 $L(M) = A \cup B$. 证毕

正则语言的交是正则语言

定理: 设A,B都是Σ上的正则语言, 则$A \cap B$也是正则语言.

证明: 设$M_1=(Q_1,\Sigma,\delta_1,s_1,F_1)$和$M_2=(Q_2,\Sigma,\delta_2,s_2,F_2)$是有限自动机,

且 $L(M_1)=A$, $L(M_2)=B$,

令$M=(Q,\Sigma,\delta,s,F)$,

其中$Q=Q_1\times Q_2$, $s=(s_1,s_2)$, $F=F_1\times F_2$,

δ: $Q\times\Sigma\to Q$, $\forall a\in\Sigma$, $r_1\in Q_1$, $r_2\in Q_2$,

$\delta((r_1,r_2) , a) = (\delta_1(r_1,a), \delta(r_2,a))$,

则 $\forall x (x\in L(M) \leftrightarrow x\in A \cap B)$

即 $L(M) = A \cap B$. 证毕

正则语言对相对补和对称差运算封闭

定理：正则语言对补运算封闭

定理: 设A, B都是Σ上的正则语言, 则A∪B也是正则语言.

定理: 设A,B都是Σ上的正则语言, 则A∩B也是正则语言.

推论：正则语言对相对补和对称差运算封闭。

相对补　$A-B = A\cap\sim B$

对称差　$A\oplus B = (A-B)\cup(B-A)$

第1章 有限自动机

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

- 0. 引论--语言--什么是问题
- 1. 确定有限自动机
- 2. 非确定有限自动机
  - ¶ 非确定型机器
  - ¶ NFA的形式定义
  - ¶ NFA计算的形式定义
  - ¶ NFA举例
  - ¶ NFA设计
  - ¶ NFA和DFA等价
- 3. 正则表达式
- 4. 正则语言的泵引理

非确定型机器

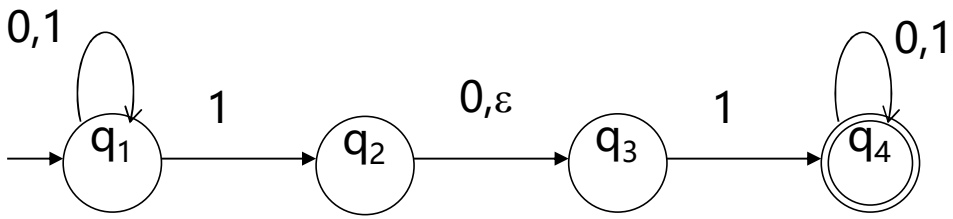
确定型有限自动机(DFA)：Deterministic Finite Automaton

δ: Q× Σ→Q, 下一个状态是唯一确定的

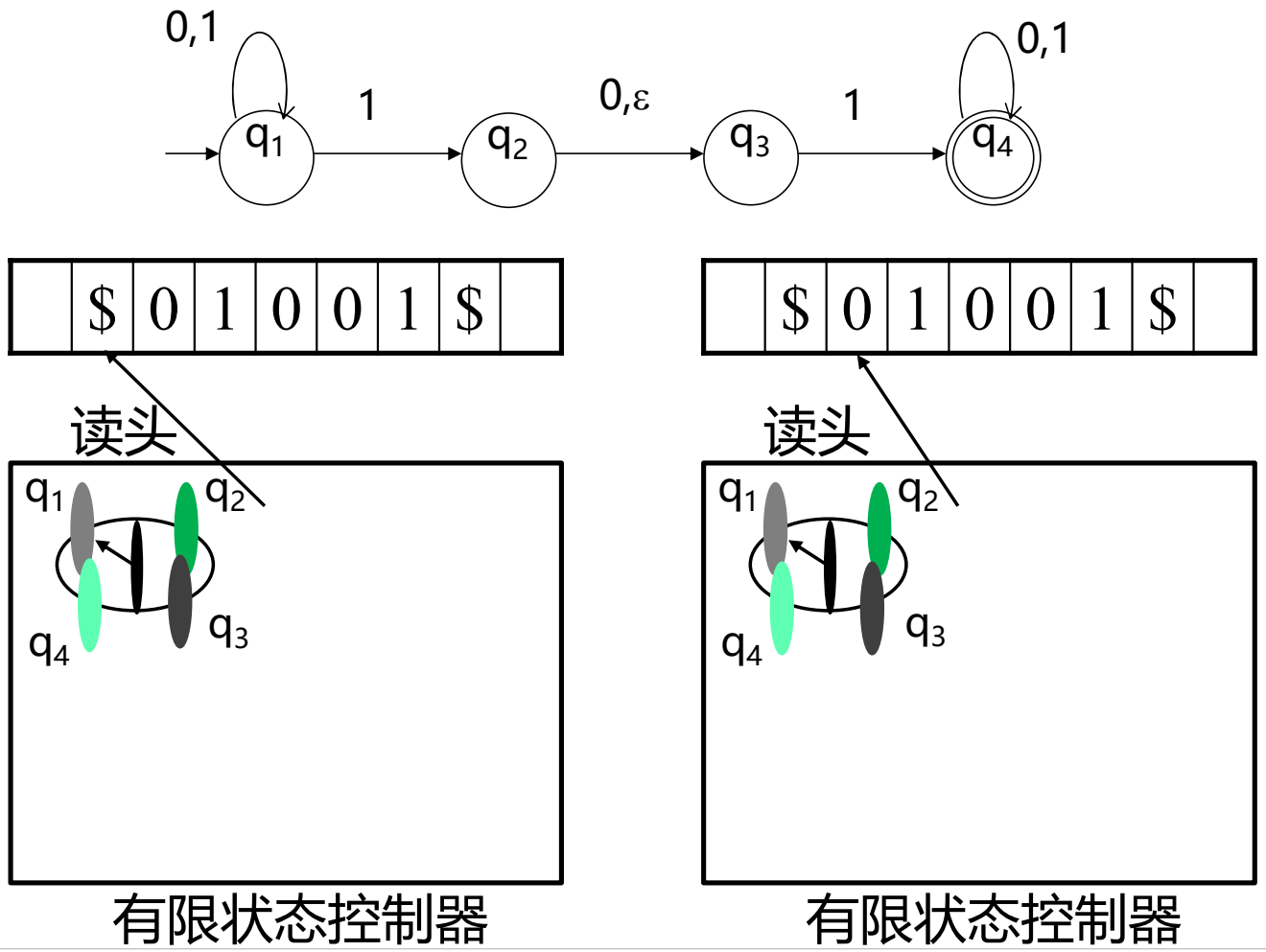
非确定型有限自动机(NFA)：Nondeterministic Finite Automaton

每步可以0至多种方式进入下一步

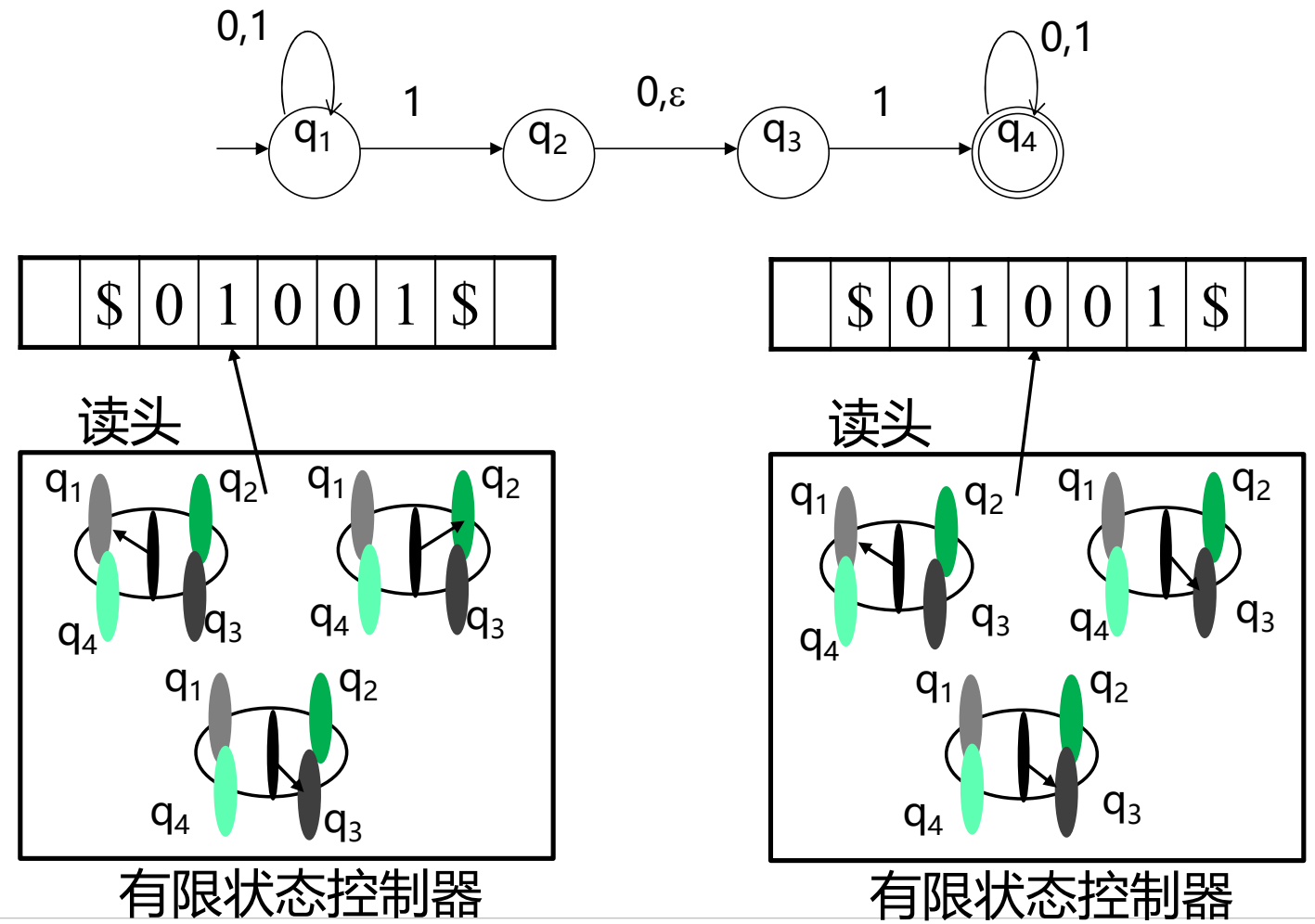
转移箭头上的符号可以是空串ε，表示不读任何输入就可以转移过去



非确定型机器计算示例

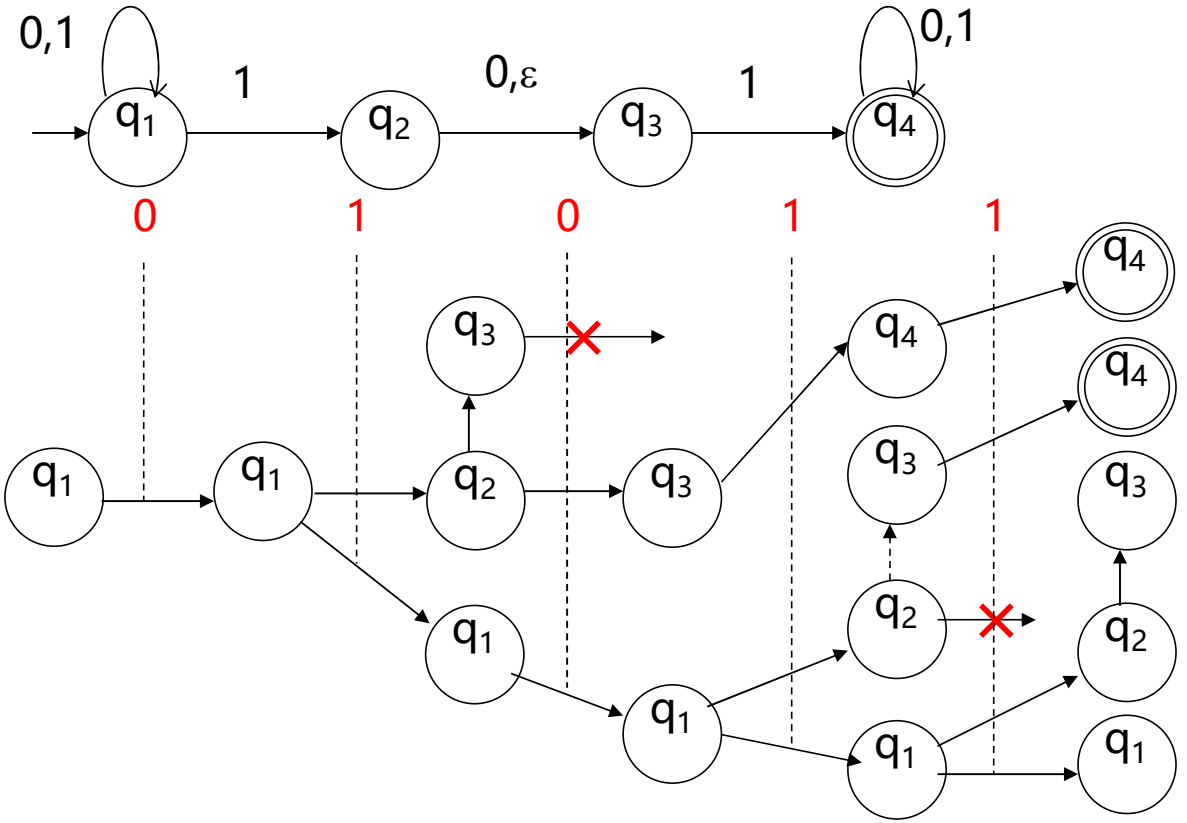

0,1
1
0,ε
1
0,1
q1
q2
q3
q4
$ 0 1 0 0 1 $
读头
q1
q2
q3
q4
有限状态控制器
$ 0 1 0 0 1 $
读头
q1
q2
q3
q4
有限状态控制器

非确定型机器计算示例



非确定型计算

输入
01011



NFA的计算方式

1.设读到符号s, 对(每个副本)机器状态q,若q有多个射出s箭头，则机器把自己复制为成多个副本.

2. 对每个副本的状态,若其上有射出标ε的箭头,

则不读任何输入, 机器复制出相应副本.

3. 读下一个输入符号, 若有符号则转1.

若无输入符号, 计算结束, 并且,

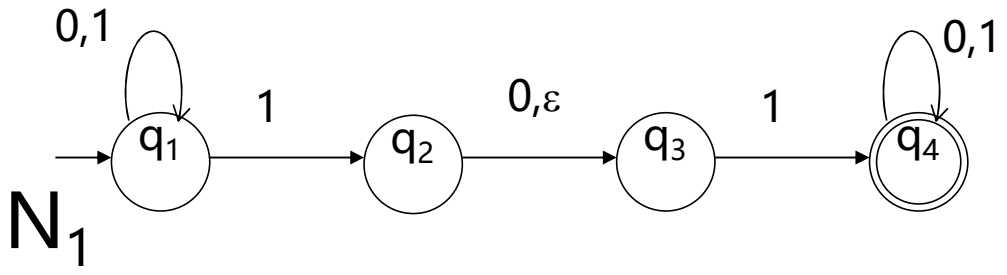
若此时有一个副本处于接受状态,则接受,

否则拒绝.

NFA的形式定义

定义: NFA是一个5元组(Q,Σ,δ,s,F),

- Q是状态集;
- Σ是字母表;
- δ: $Q\times \Sigma_\varepsilon \to P(Q)$是转移函数;
  - 其中 $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$
- $s\in Q$是起始状态;
- $F\subseteq Q$是接受状态集;



状态图
与
形式定义
包含
相同信息

试写出该状态图
对应的形式定义

$\delta(q_1,1) = \{q_1,q_2\}$
$\delta(q_2,\varepsilon) = \{q_3\}$
$\delta(q_2,1) = \varnothing$
$\delta(q_1,\varepsilon) = \varnothing$

NFA的形式定义与DFA的形式定义-对比

定义: NFA是一个5元组(Q,Σ,δ,s,F),

Q是状态集;

Σ是字母表;

δ: $Q\times \Sigma_\varepsilon \to P(Q)$是转移函数;

其中 $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

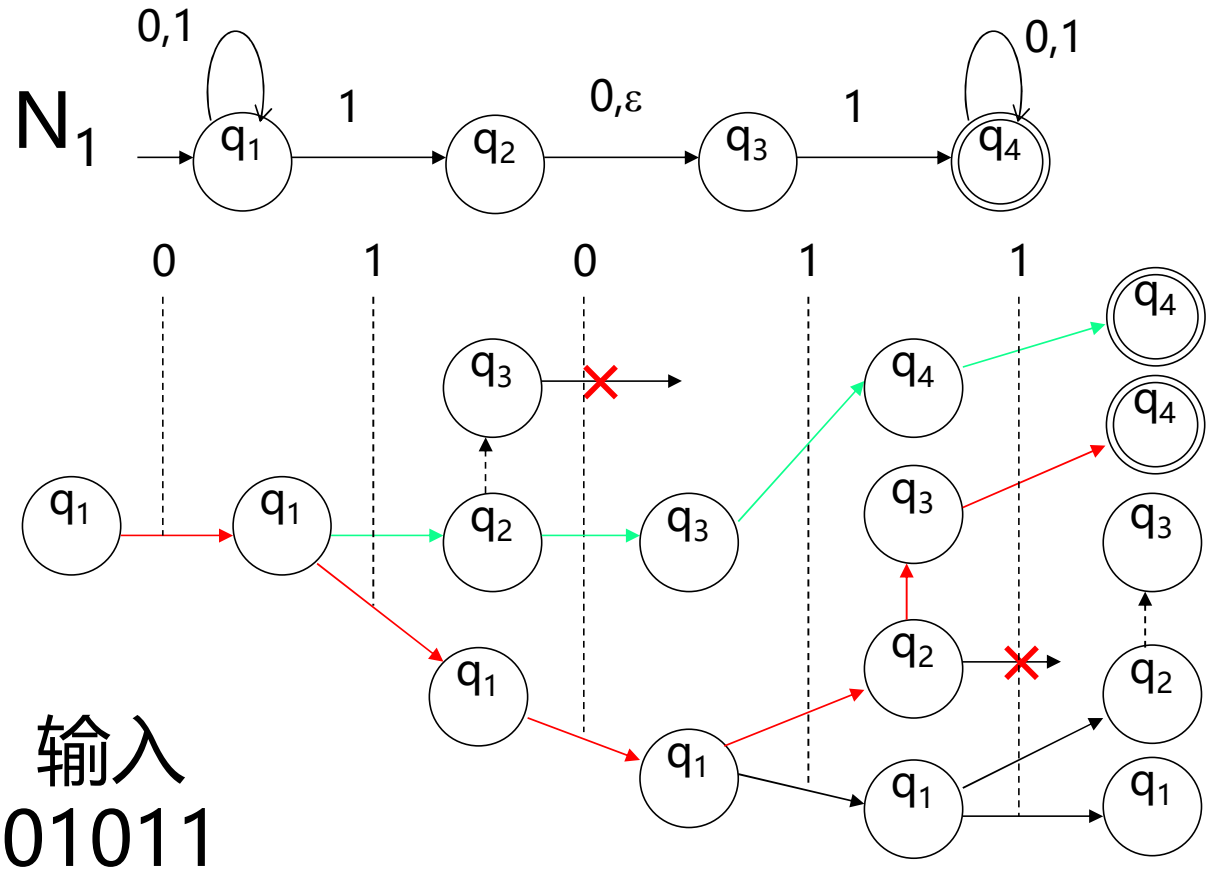
$s\in Q$是起始状态;

$F\subseteq Q$是接受状态集;

定义: DFA是一个5元组(Q,Σ,δ,s,F),

1) Q是有限集, 称为状态集;
2) Σ是有限集, 称为字母表;
3) δ: $Q\times\Sigma\to Q$是转移函数;
4) $s\in Q$是起始状态;
5) $F\subseteq Q$是接受状态集;

如何定义NFA的计算



NFA计算的形式定义

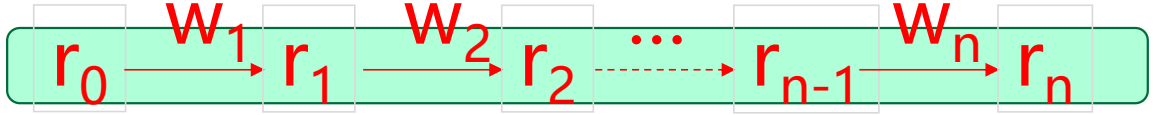
设$N=(Q,\Sigma,\delta,q_0,F)$是一台NFA, w是$\Sigma$上字符串

称 N接受w,

若 w能写作$w=w_1w_2...w_n$, $w_i\in\Sigma_\varepsilon$,且

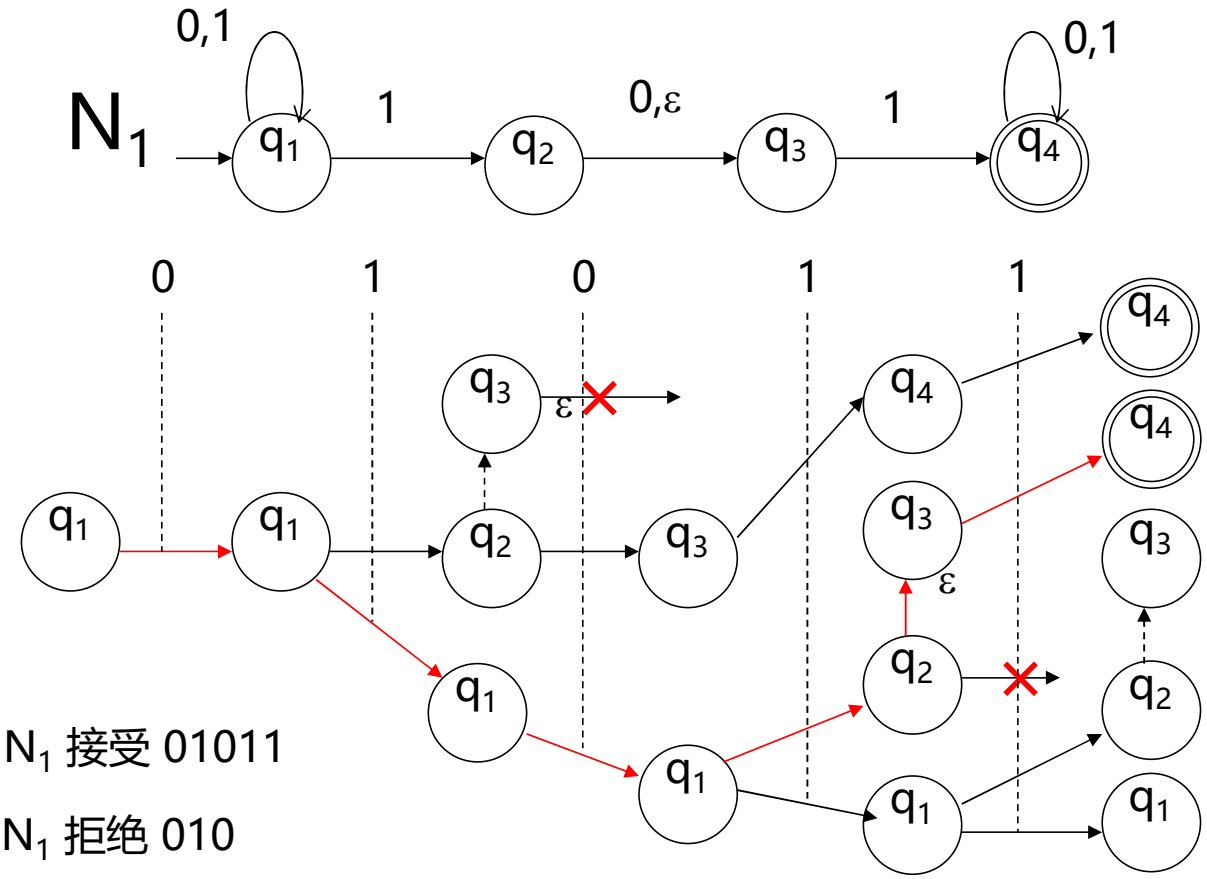
存在Q中的状态序列$r_0,r_1,...,r_n$, 满足

1) $r_0 = q_0$;
2) $r_{i+1}\in\delta(r_i, w_{i+1})$;
3) $r_n \in F$



对于输入, NFA计算的路径可能不唯一.

NFA计算形式定义举例



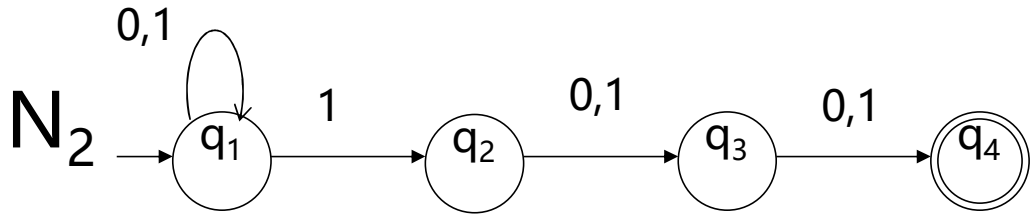
N_1 接受所有包含11或者101子串的字符串。

NFA举例

$L(N_2)=\{w \mid w$的倒数第三个字符是1$\}$

$\Sigma=\{0, 1\}$

NFA：猜测能力



NFA的设计(难点)

自己即自动机

寻找需要记录的关键信息

设计识别{0,1}上以下语言的NFA:

例1： $\{ w \in \{0,1\}^* \mid w$从1开始, 以0结束 $\}$

例2： $\{ w \in \{0,1\}^* \mid w$含有子串1010 $\}$

例3： $\{ w \in \{0,1\}^* \mid w$是倒数第2位是1 $\}$

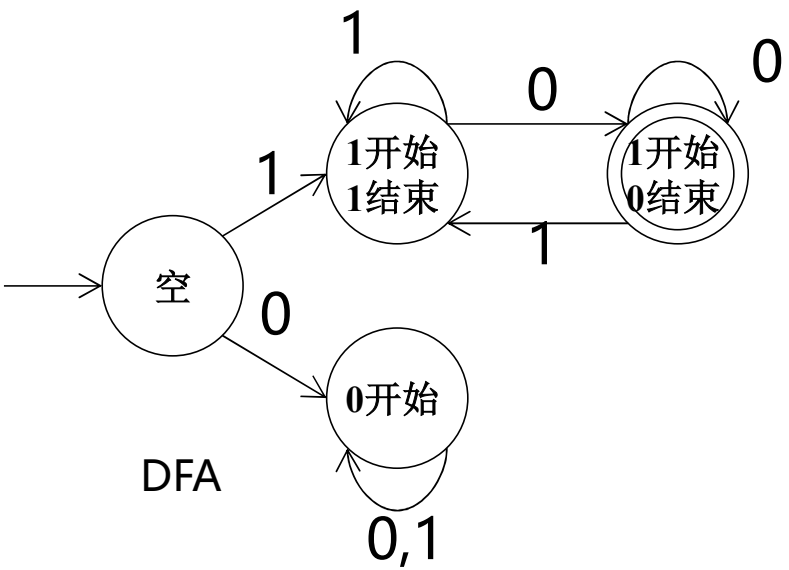
例4： $\{ 0^k \mid k$是2或3的倍数 $\}$

NFA的设计

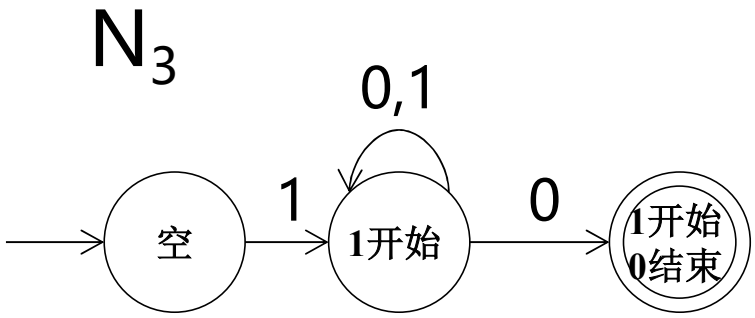
例1：{ w∈{0,1}* | w从1开始, 以0结束 }

Σ={0,1}, 根据关键信息设计状态,

空, 以0开始, 以1开始以1结束, 以1开始以0结束



DFA

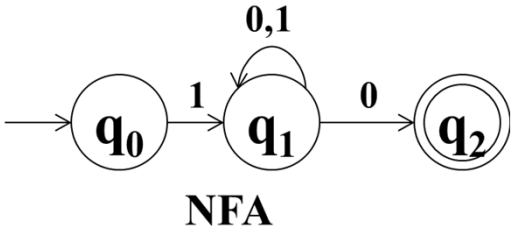


NFA

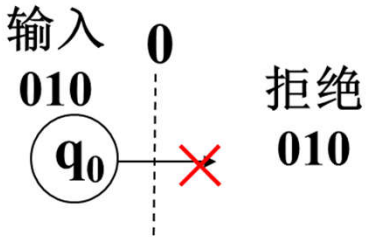
NFA计算举例

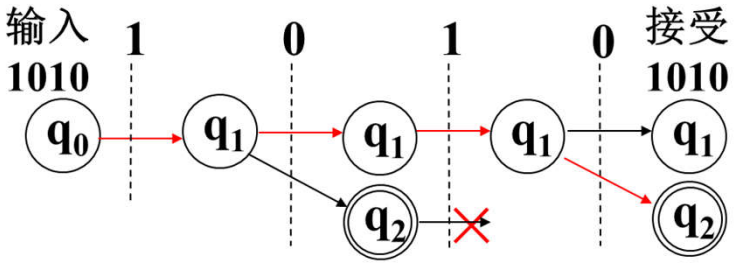
$A = \{ w \in \{0,1\}^* \mid w$ 从1开始, 以0结束 $\}$

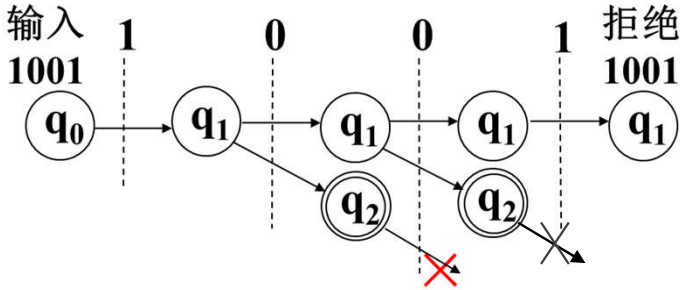
状态: (q_0, q_1, q_2)



1010
1001
010



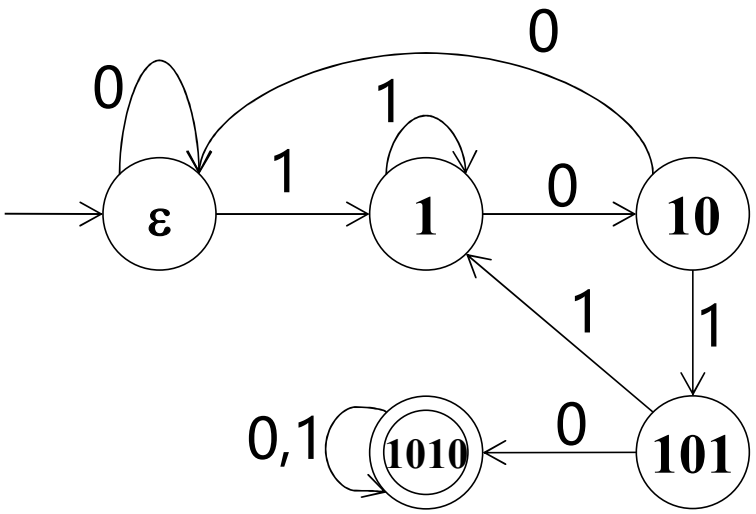




NFA的设计

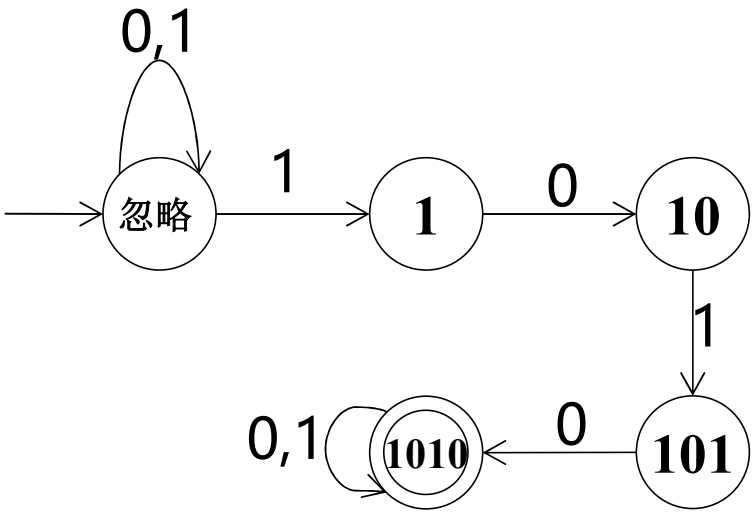
{ w∈{0,1}* | w含有子串1010 }

Σ={0,1}, 关键信息: 忽略(ε), 1, 10, 101, 1010



DFA

N_4

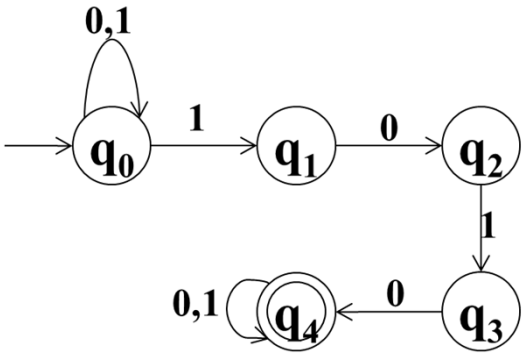


NFA

NFA计算举例

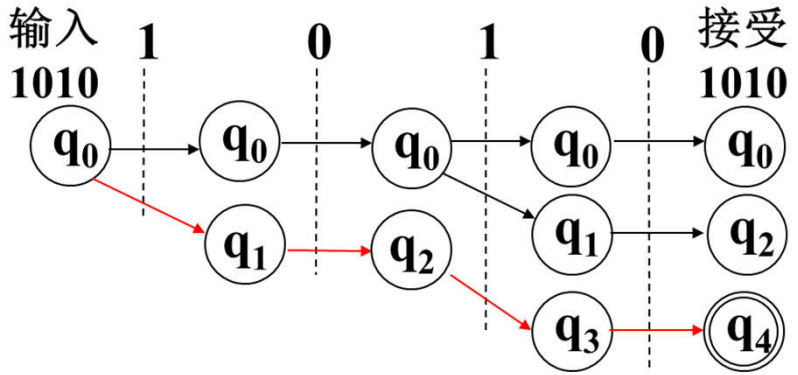
B = { w∈{0,1}* | w含有子串1010 }

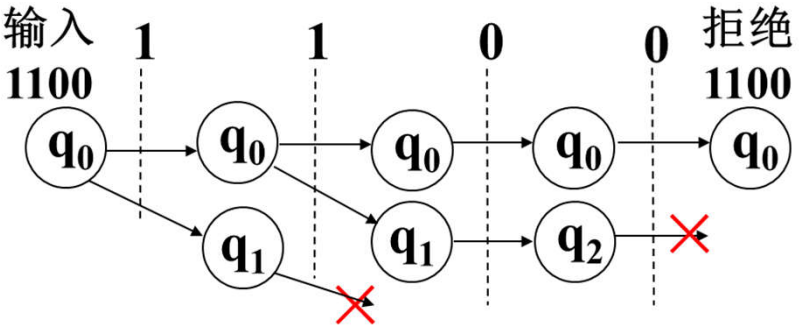
状态: (q_0,q_1,q_2,q_3,q_4) = (Ignore(ε), 1, 10, 101, 1010)



1010

1100

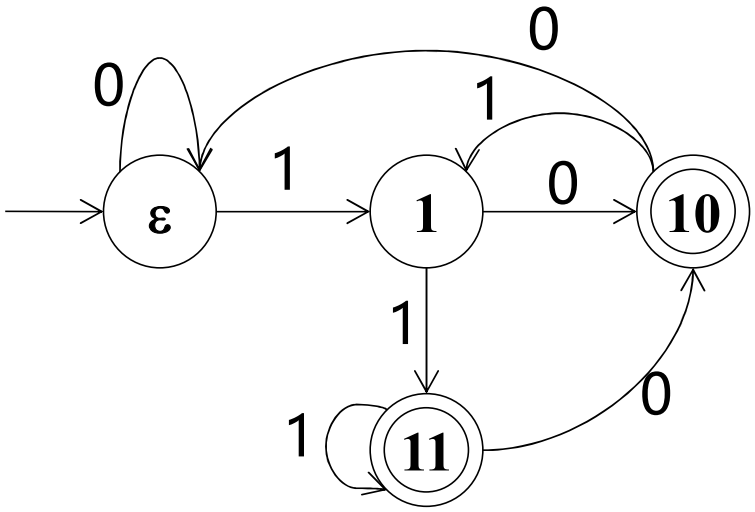




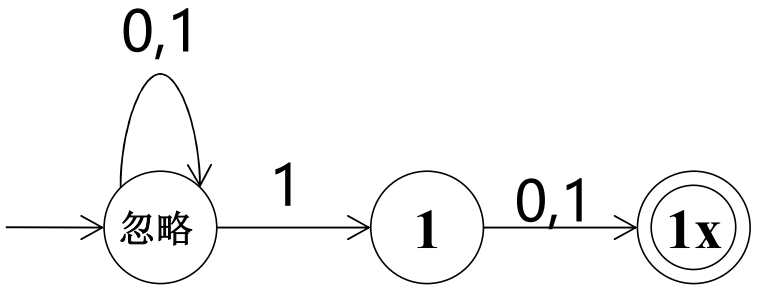
NFA的设计

{ w∈{0,1}* | w倒数第2个符号是1 }

Σ={0,1}, 关键信息: 忽略(ε), 1, 1x,



DFA

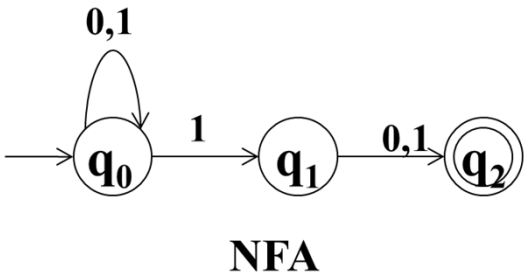


N_5 NFA

NFA计算举例

C = { w∈{0,1}* | w的倒数第2个符号是1 }

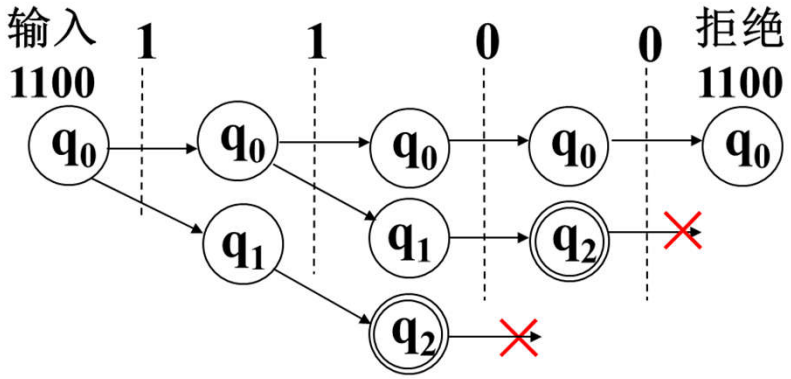
状态: (q_0,q_1,q_2) = (忽略(ε), 1, 1x)

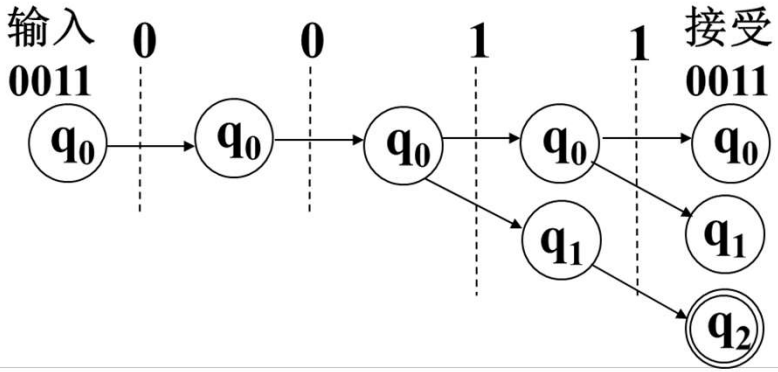


NFA

1100

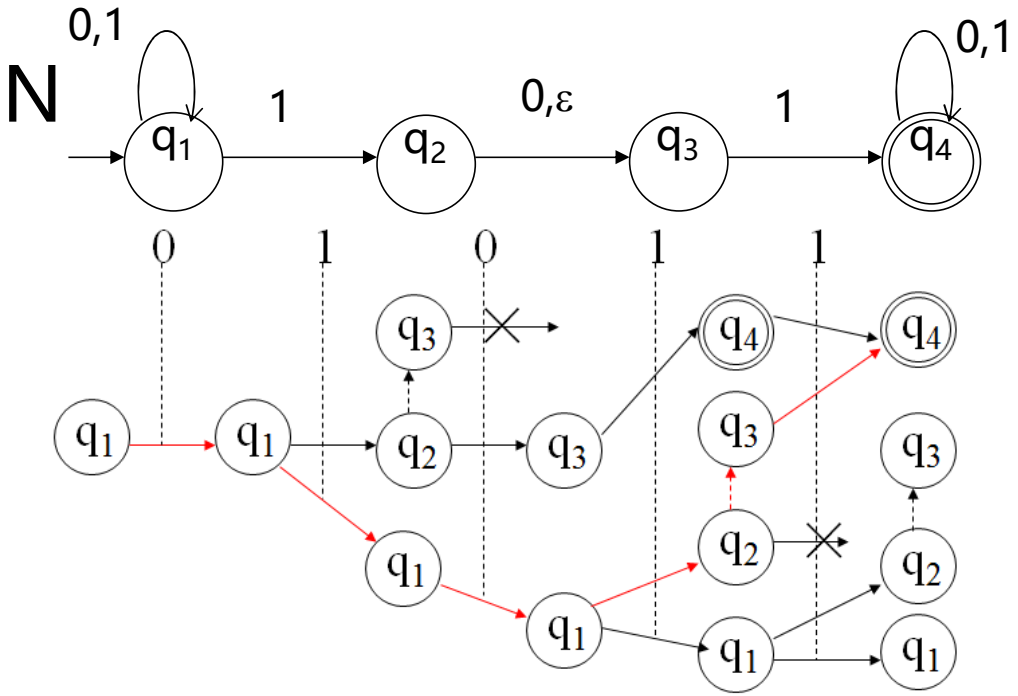
0011





NFA与DFA等价

定理: 每个NFA都有一台等价的DFA.



构造DFA关键信息: 副本状态的集合.

起始? 接受状态集?转移?

不确定:

在状态q_1读到1,

进入哪个状态?

确定:

在状态q_3读到1,

进入哪些状态?

进一步确定:

给定副本状态集,

读到符号1,

得到的副本状态集

每个NFA都有等价的DFA

NFA的确定化:子集法

(1)确定起始状态S′:将从 NFA N的起始状态S出发经过任意条ε弧所能到达的状态组成的集合作为确定化后的 DFA M的起始状态S′。

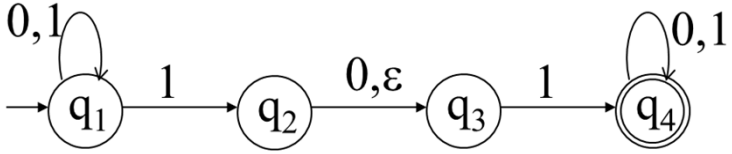
(2)确定其它状态：从S′出发，经过对任意输入符号a∈∑的状态转移所能到达的状态（包括读入输入符号a之后所有可能的ε转移所能到达的状态）所组成的集合作为M的新状态。

(3)如此重复，直到不再有新的状态出现为止。

(4)确定接受状态：在所产生的状态中，含有原NFA接受态的子集作为DFA的接受态。

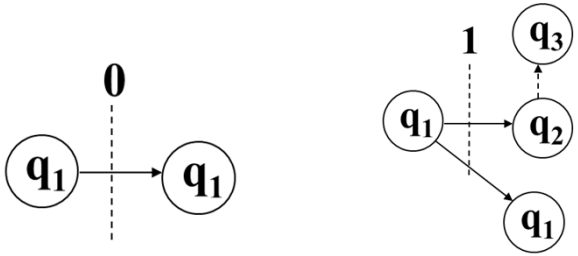
构造和N等价的DFA M

$N=(Q_1, \Sigma, \delta_1, q_1, \{q_4\})$



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

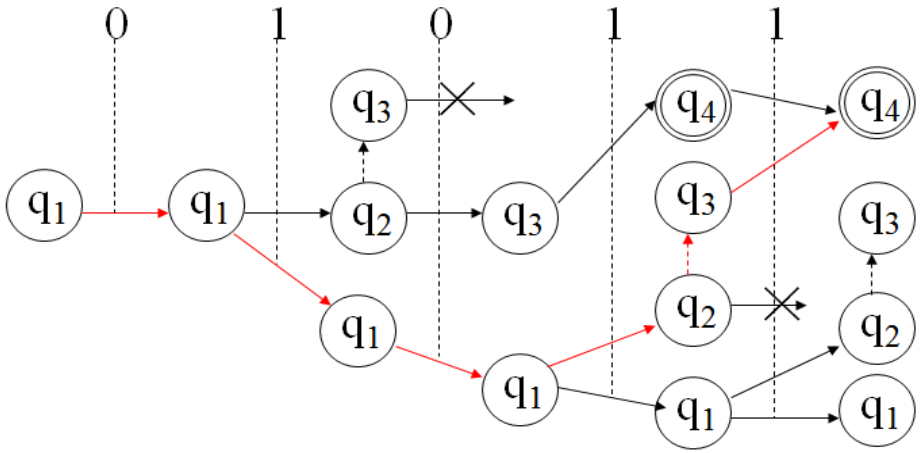
$Q=P(Q_1)$, 令$s = S_1 =\{q_1\}$



令 $S_2=\{q_1,q_2,q_3\}$. 则有

$\delta(S_1,0) = S_1$, $\delta(S_1,1) = S_2$,

$\delta(A, a)=E(\cup_{r\in A}\delta_1(r,a))$



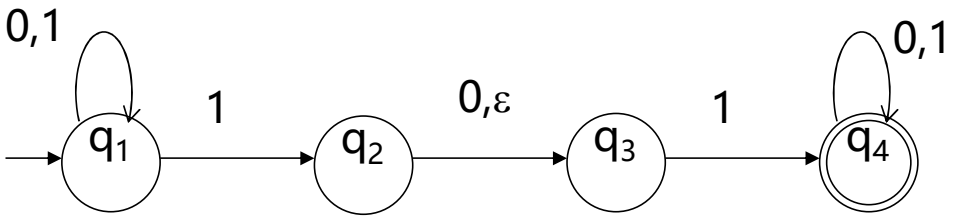
S_1--0--S_1--1--S_2

$Q = \{ S_1, S_2, \ldots \}$,

$s = S_1$,

$\delta(S_1,0) = S_1$, $\delta(S_1,1) = S_2$,

每个NFA都有等价的DFA

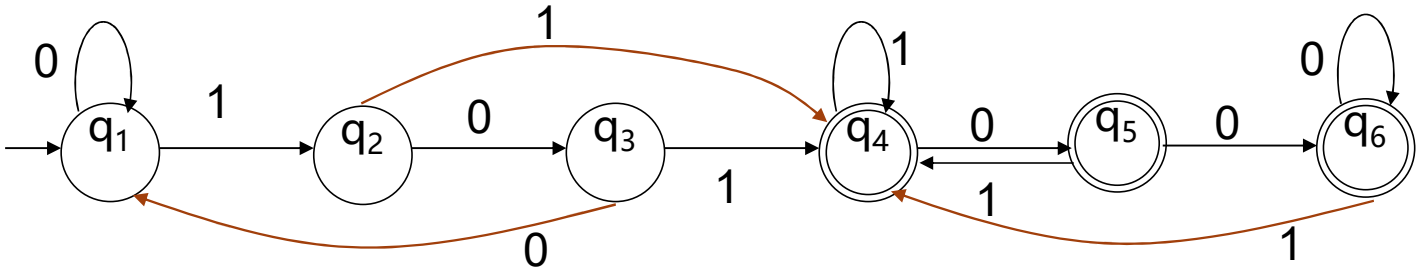


以原状态的子集为新机器的状态

编号	δ	0	1
1	{q_1} 1	{q_1}	{q_1, q_2, q_3}2
2	{q_1, q_2, q_3}	{q_1, q_3}3	{q_1, q_2, q_3, q_4}4
3	{q_1, q_3}	{q_1}	{q_1, q_2, q_3, q_4}
4*	{q_1, q_2, q_3, q_4}	{q_1, q_3, q_4}5	{q_1, q_2, q_3, q_4}
5*	{q_1, q_3, q_4}	{q_1, q_4}6	{q_1, q_2, q_3, q_4}
6*	{q_1, q_4}	{q_1, q_4}	{q_1, q_2, q_3, q_4}

N_1：包含11或者101子串的字符串

每个NFA都有等价的DFA



编号	δ	0	1
1	$\{q_1\}$ 1	$\{q_1\}$	$\{q_1, q_2, q_3\}$2
2	$\{q_1, q_2, q_3\}$	$\{q_1, q_3\}$3	$\{q_1, q_2, q_3, q_4\}$4
3	$\{q_1, q_3\}$	$\{q_1\}$	$\{q_1, q_2, q_3, q_4\}$
4*	$\{q_1, q_2, q_3, q_4\}$	$\{q_1, q_3, q_4\}$5	$\{q_1, q_2, q_3, q_4\}$
5*	$\{q_1, q_3, q_4\}$	$\{q_1, q_4\}$6	$\{q_1, q_2, q_3, q_4\}$
6*	$\{q_1, q_4\}$	$\{q_1, q_4\}$	$\{q_1, q_2, q_3, q_4\}$

N_1：包含11或者101子串的字符串

每个NFA都有等价的DFA

证明: 设$N=(Q_1, \Sigma, \delta_1, s_1, F_1)$是NFA,//构造一个DFA $M=(Q,\Sigma,\delta,s,F)$

令 $Q = P(Q_1)$,　//Q1的幂集

$F = \{ A \in Q : F_1 \cap A \neq \varnothing \}$,

$s = E(\{s_1\})$, $E(A) = \{ q : \exists r \in A$, r经0到多个ε箭头可达q $\}$

$\delta: Q \times \Sigma \to Q$,　$\forall a \in \Sigma, \forall A \in Q$,

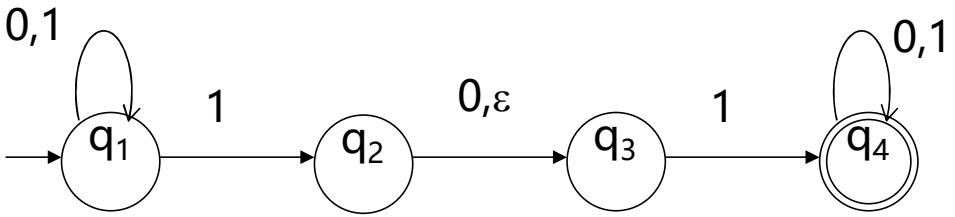
$$\delta(A, a) = E(\cup_{r \in A} \delta_1(r, a))$$

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

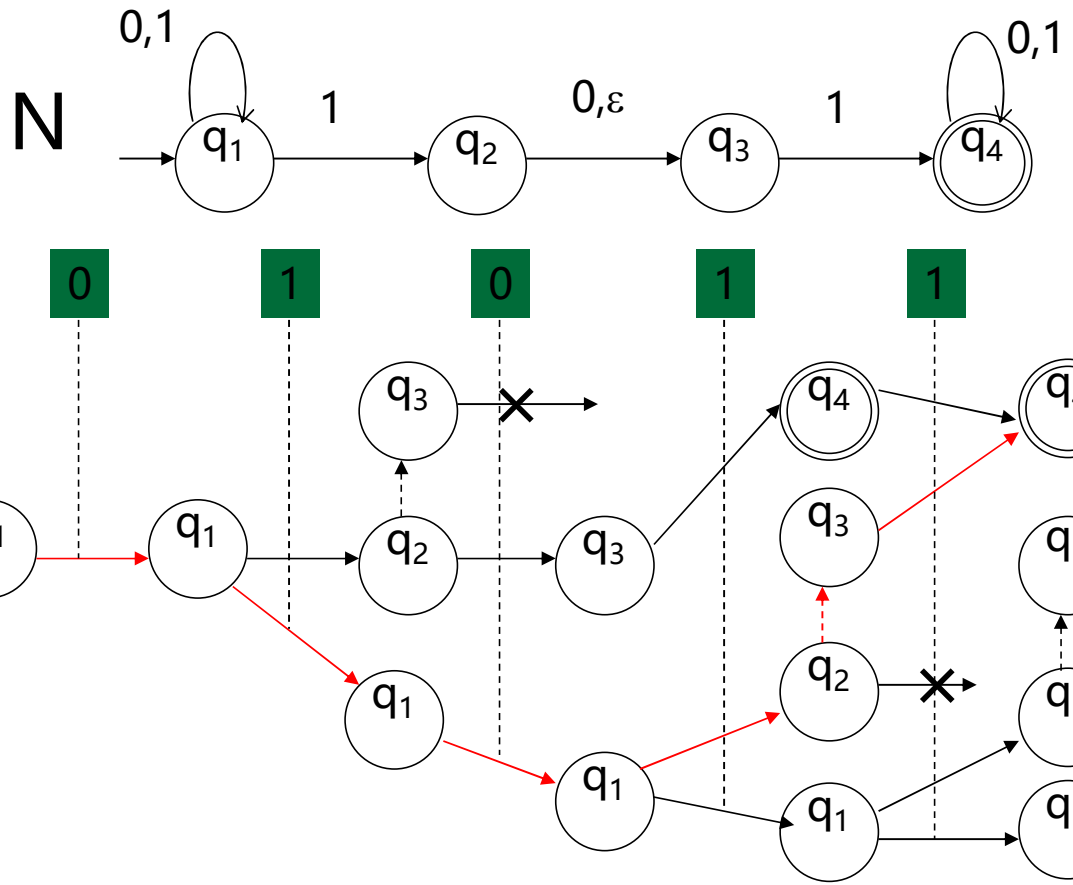
则 $\forall x (x \in L(M) \leftrightarrow x \in L(N))$,

即 $L(M) = L(N)$.

证毕.



$\forall x\ (\ x \in L(M) \leftrightarrow x \in L(N)\ )$



N接受w:

w=0101ε1

q_1-0-q_1-1-

q_1-0-q_1-1-

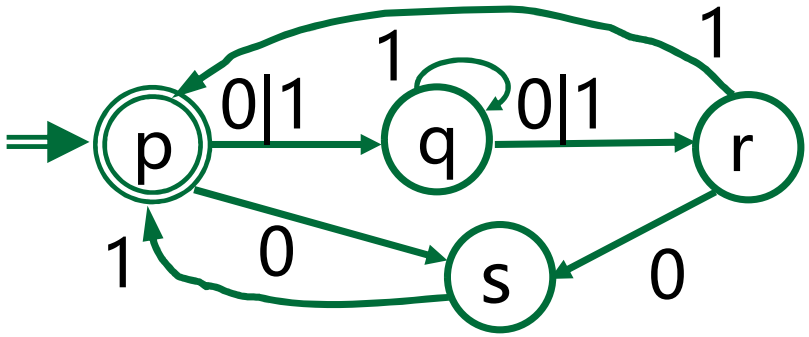
q_2-ε-q_3-1-

$q_4 \in F_1$

S_1--0--S_1--1--S_2--0--S_3--1--S_4--1--S_4　M接受w

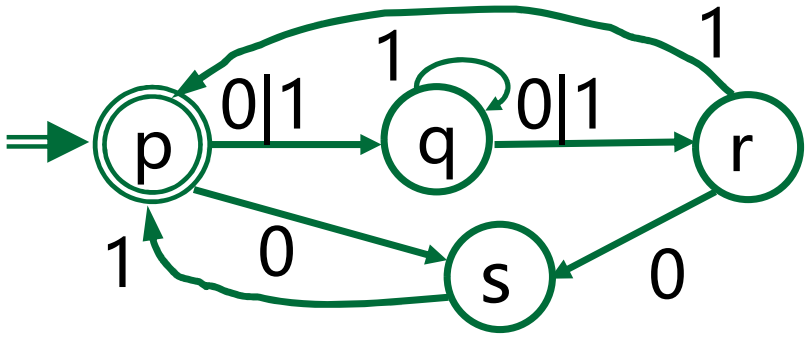
构造和N等价的DFA

课堂练习



构造和N等价的DFA

课堂练习



state	0	1
0*	1	2
1	3	4
2	3	5
3	6	0
4*	7	4
5	8	4
6		0
7	8	4
8	6	0

正则运算的封闭性

定理：每个NFA都有等价的DFA。

推论：一个语言是正则的，当且仅当有一个NFA识别它。

定理: 正则语言对并运算封闭.

定理: 正则语言对连接运算封闭.

定理: 正则语言对星号运算封闭.

证明方法: 构造一个NFA，画状态图.

证明：若A, B正则, 则A∪B正则

设DFA: $M_1=(Q_1,\Sigma,\delta_1,s_1,F_1)$, $L(M_1)=A$;

DFA:$M_2=(Q_2,\Sigma,\delta_2,s_2,F_2)$, $L(M_2)=B$,

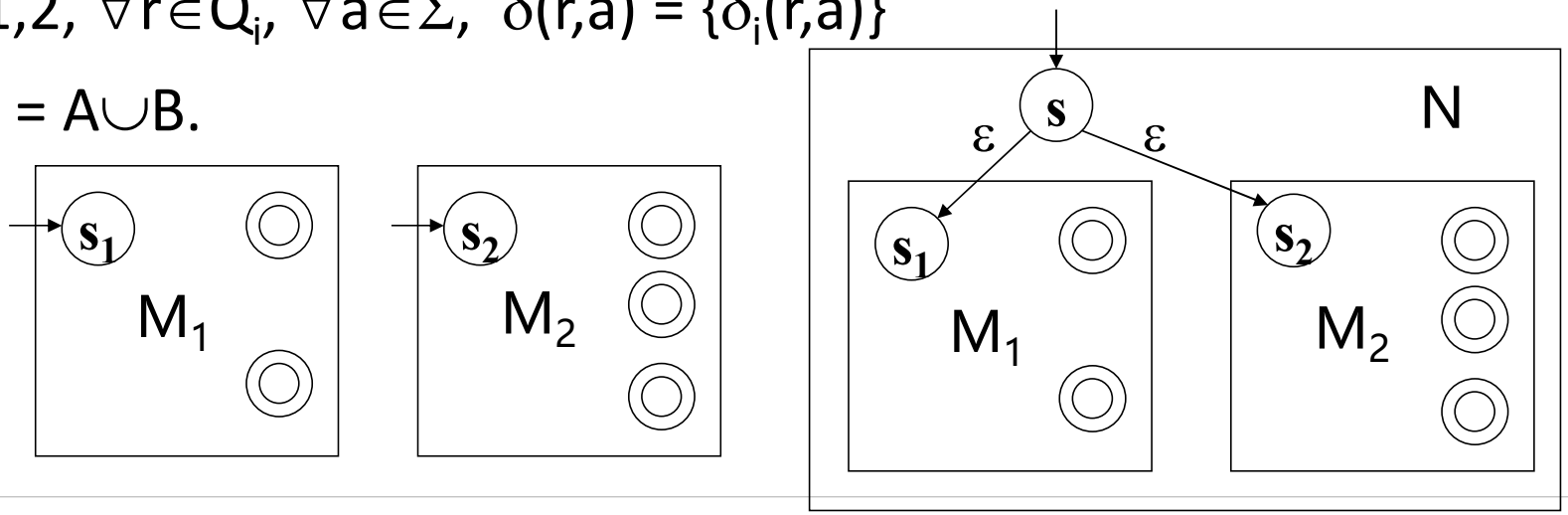
构造识别A∪B的NFA，$N=(Q,\Sigma,\delta,s,F)$。

令 $Q = Q_1 \cup Q_2 \cup \{s\}$, $F = F_1 \cup F_2$, s是N的初始状态;

$\delta(s,\varepsilon) = \{s_1,s_2\}$

$\forall i=1,2, \forall r \in Q_i, \forall a \in \Sigma, \delta(r,a) = \{\delta_i(r,a)\}$

则 $L(N) = A \cup B$.



证明：若A, B正则, 则A°B正则

DFA: $M_1=(Q_1,\Sigma,\delta_1,s_1,F_1)$, $L(M_1)=A$,

$M_2=(Q_2,\Sigma,\delta_2,s_2,F_2)$, $L(M_2)=B$。

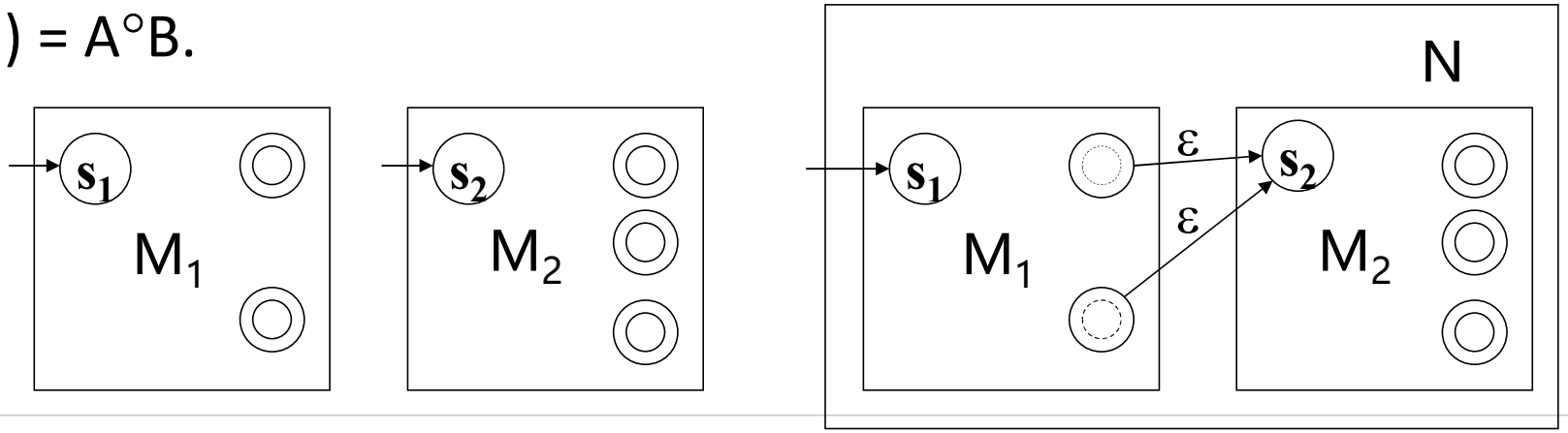
构造识别A°B的NFA， $N=(Q,\Sigma,\delta,s_1,F)$。

令 $Q = Q_1 \cup Q_2$ 不交并, $F = F_2$, s_1是N的初始状态；

$\forall r \in F_1, \delta(r,\varepsilon) = \{s_2\}$

$\forall i=1,2, \forall r \in Q_i, \forall a \in \Sigma, \delta(r,a) = \{\delta_i(r,a)\}$

则 L(N) = A°B.



证明若A正则, 则A*正则

DFA: $M_1=(Q_1,\Sigma,\delta_1,s_1,F_1)$, $L(M_1)=A$,

构造识别A*的NFA， $N=(Q,\Sigma,\delta,s,F)$。

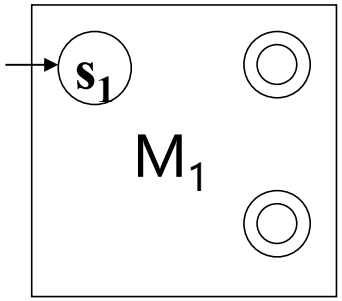
令 $Q = Q_1 \cup \{s\}$, $F = F_1 \cup \{s\}$

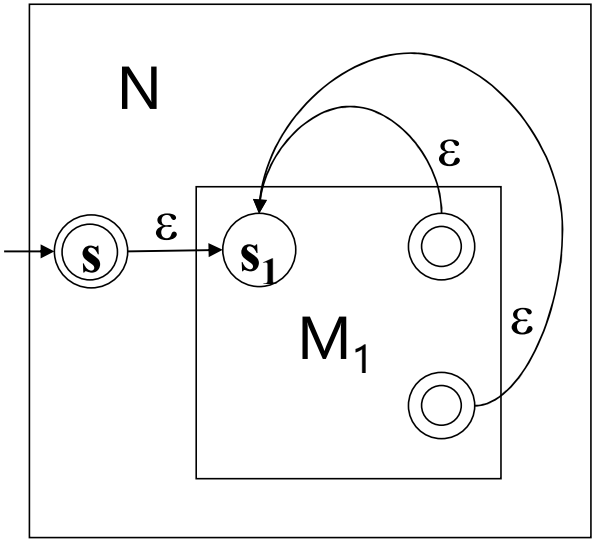
$\forall r \in Q_1$, $\forall a \in \Sigma$, $\delta(r,a) = \{\delta_1(r,a)\}$

$\forall r \in F_1$, $\delta(r,\varepsilon) = \{s_1\}$,

$\delta(s,\varepsilon) = \{s_1\}$,

则 $L(N) = A^*$.





第1章 有限自动机

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

0. 引论--语言--什么是问题
1. 确定有限自动机
2. 非确定有限自动机
3. 正则表达式
   正则表达式
   正则表达式与DFA等价
4. 正则语言的泵引理

正则表达式

定义: 称R是一个正则表达式, 若R是

1) a, $a \in \Sigma$;
2) ε ;
3) $\varnothing$;
4) $(R_1 \cup R_2)$, R_1和R_2是正则表达式;
5) $(R_1 \circ R_2)$, R_1和R_2是正则表达式;
6) (R_1^*), R_1是正则表达式;

每个正则表达式R表示一个语言,记为L(R).

1) $L(a)=\{a\}$. 2) $L(\varepsilon)=\{\varepsilon\}$.
3) $L(\varnothing)=\varnothing$
4) $L((R_1 \cup R_2))=L(R_1) \cup L(R_2)$.
5) $L((R_1 \circ R_2))=L(R_1) \circ L(R_2)$.
6) $L((R_1^*))=(L(R_1))^*$

每个正则表达式R表示一个语言,记为L(R).例:

$0^*10^*=\{w|\ w$恰好含有一个$1\}$

$01\cup10=\{01,\ 10\}$

$(\Sigma\Sigma)^*=\{w|\ w$是含有偶数个字符的字符串$\}$

$1^*\varnothing\ =\varnothing$

$\varnothing^*=\ \{\varepsilon\}$

正则表达式与DFA等价

定理2.3.1: 语言A是正则的⇔A可用正则表达式描述.

(⇐) 若 语言A可用正则表达式描述,

则 A是正则的.

证明方法：数学归纳法

即，用DFA(NFA)识别正则表达式的基本形式，然后归纳

(⇒) 若 语言A是正则的,

则A可用正则表达式描述.

证明方法：将DFA识别的语言变换为正则表达式

A可用正则表达式描述⇒A正则

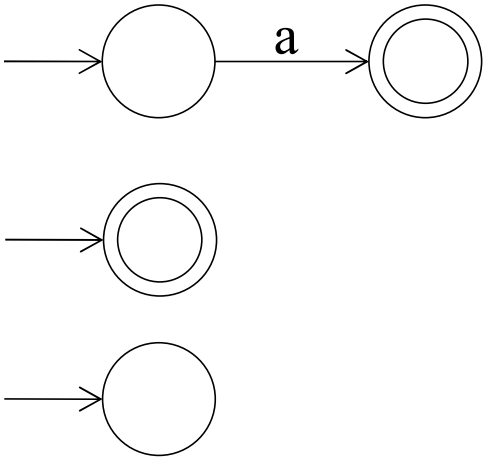
数学归纳法

R是一个正则表达式, 若R是

1) a, $a \in \Sigma$
2) ε
3) $\varnothing$
4) $(R_1 \cup R_2)$
5) $(R_1 \circ R_2)$
6) (R_1^*)

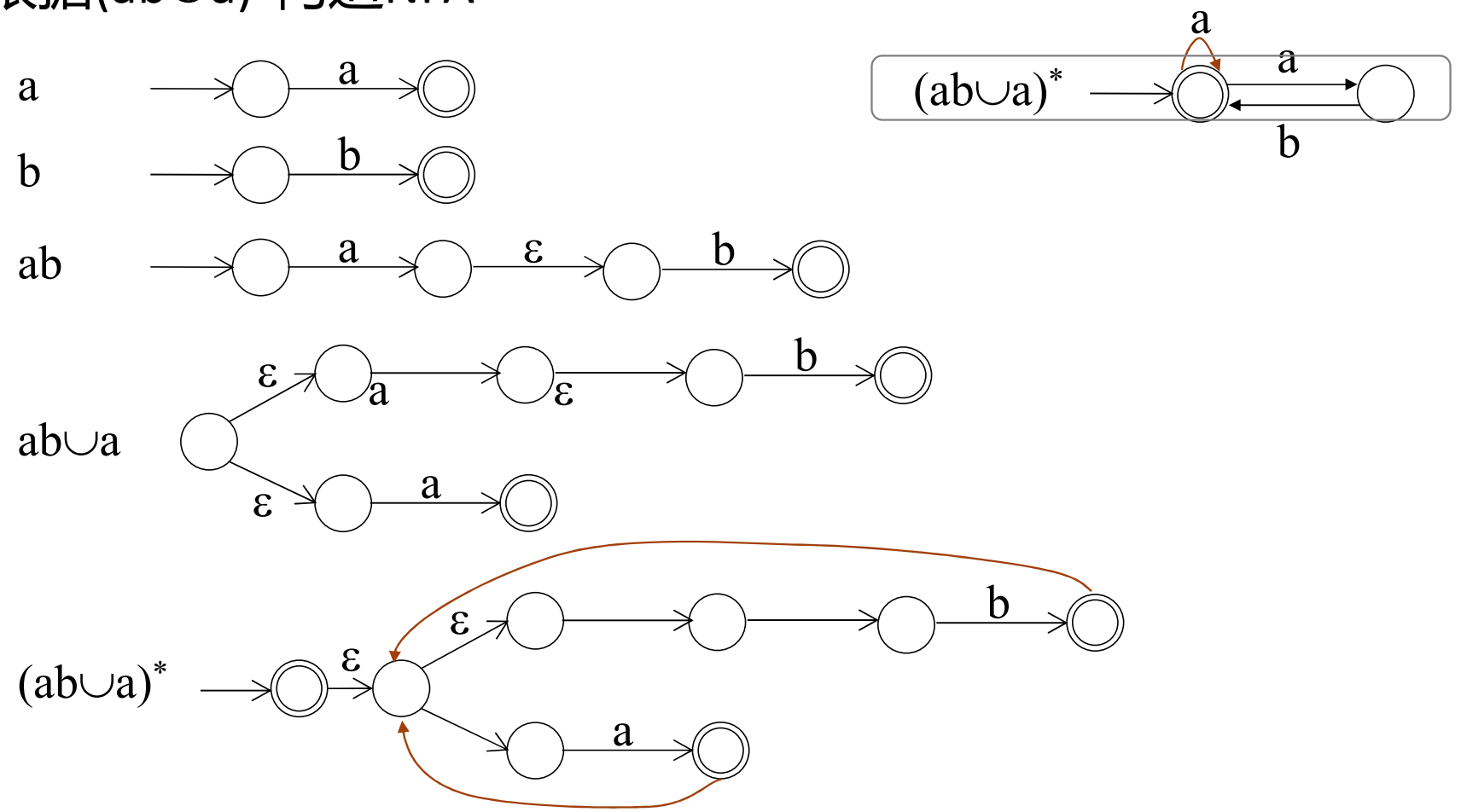
针对正则表达式进行归纳

例如：$(ab \cup a)^*$



A可用正则表达式描述⇒A正则

例：根据$(ab\cup a)^*$构造NFA



A正则⇒A可用正则表达式描述

证明方法：将DFA转换为等价的正则表达式

构造性证明：构造广义非确定有限自动机(GNFA)

转移箭头可以用任何正则表达式作标号，如ab^*，$ab \cup a$等。

GNFA读入符号段，不必一次读入一个符号

证明中的特殊要求:

起始状态无射入箭头.

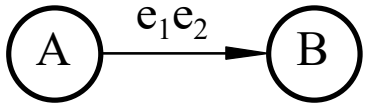
唯一接受状态(无射出箭头).

其它状态到自身和其它每一个状态都有一个箭头.

手段: 一个一个地去掉中间状态.

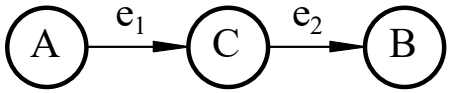
正则表达式到NFA的转换

(1)



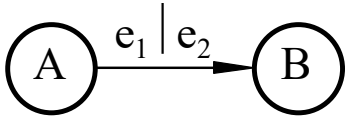
(a)

替换成



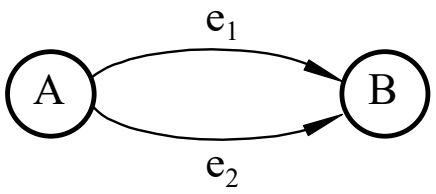
(b)

(2)



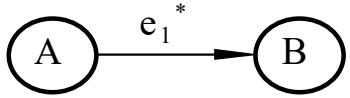
(c)

替换成



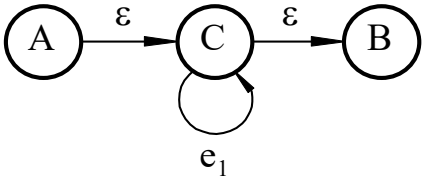
(d)

(3)



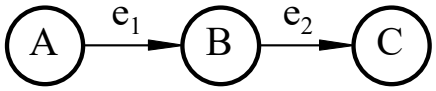
(e)

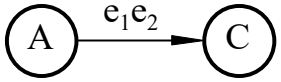
替换成

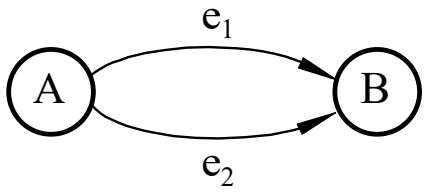


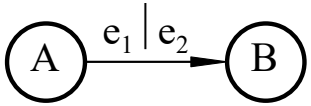
(f)

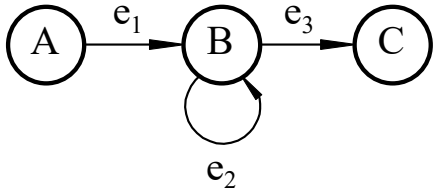
NFA到正则表达式的转换

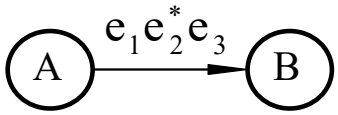
(1) 

替换成 

(2) 

替换成 

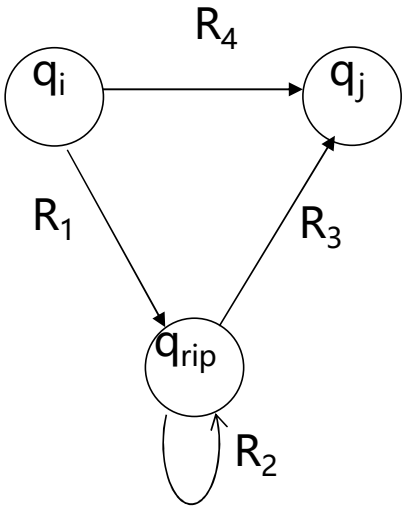
(3) 

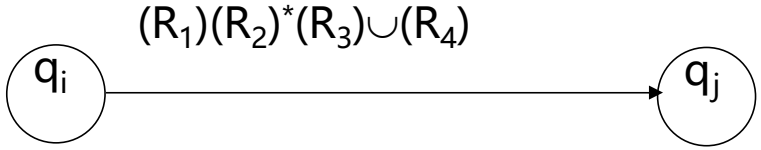
替换成 

删除中间状态

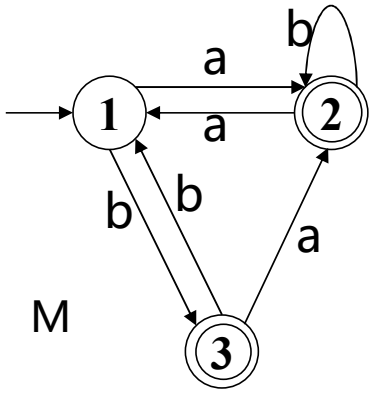
设q_{rip}为待删中间状态,

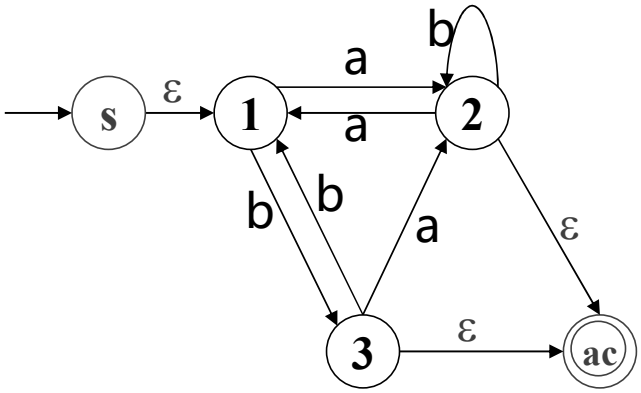
对任意两个状态q_i, q_j都需要修改箭头标号



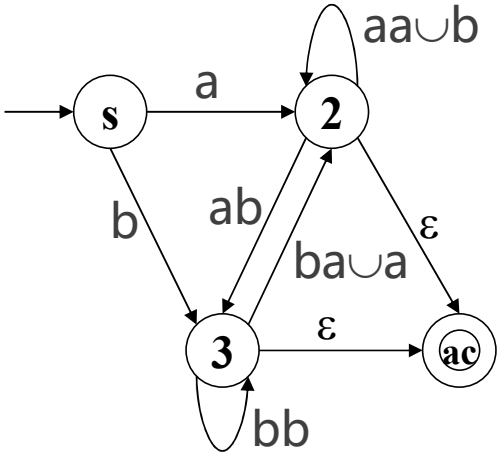


举例: A正则⇒A有正则表达式



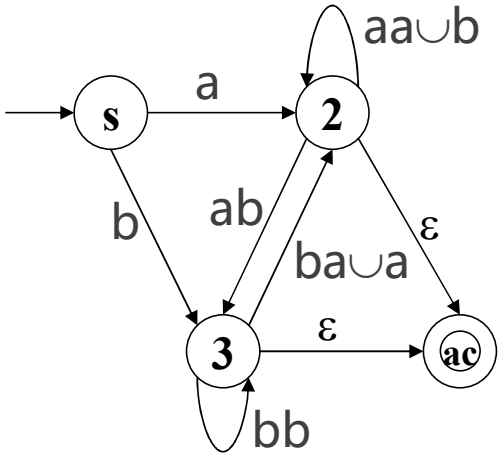


1. 添起始状态s，接受状态ac，
改掉其它接受状态

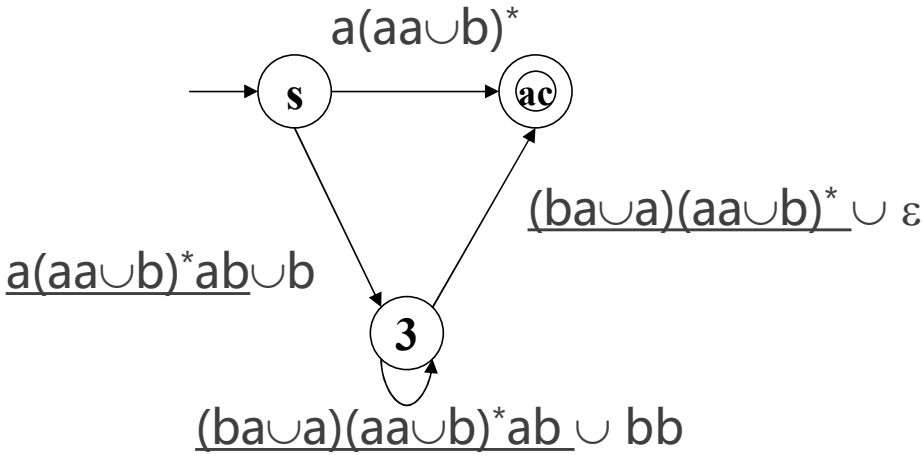


2. 删状态1

举例: A正则⇒A有正则表达式

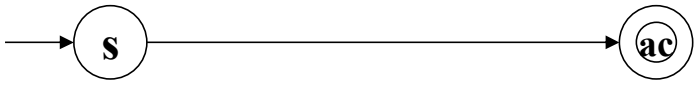


M



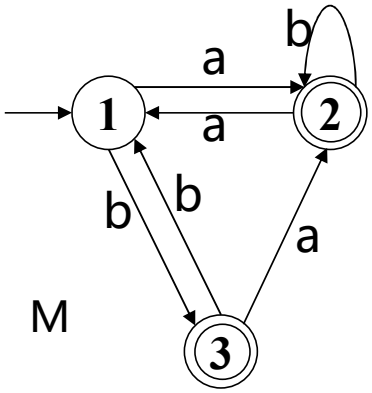
3. 删除状态2

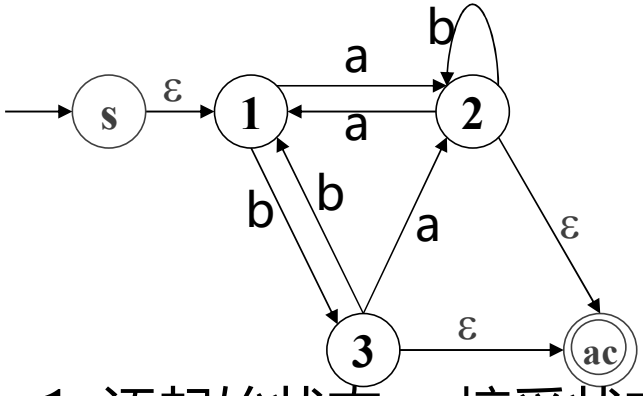
((a(aa∪b)*ab∪b)((ba∪a)(aa∪b)*ab ∪ bb)*
((ba∪a)(aa∪b)*∪ε)) ∪ (a(aa∪b)*)



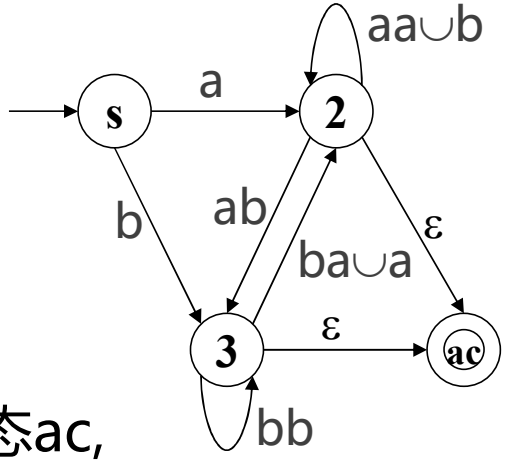
4. 删除状态3

举例: A正则⇒A有正则表达式

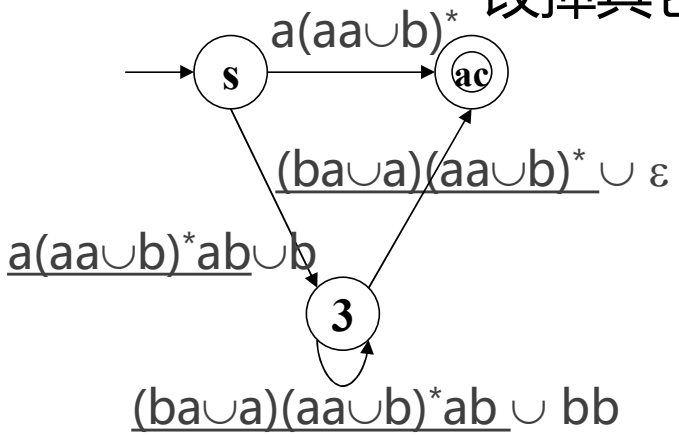




1. 添起始状态s，接受状态ac，改掉其它接受状态

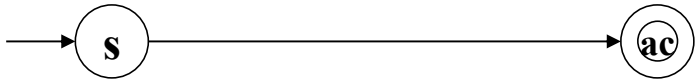


2. 删状态1



3. 删除状态2

((a(aa∪b)*ab∪b)((ba∪a)(aa∪b)*ab ∪ bb)*
((ba∪a)(aa∪b)*∪ε)) ∪ (a(aa∪b)*)



4. 删除状态3

第1章 有限自动机

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

0. 引论--语言--什么是问题
1. 确定有限自动机
2. 非确定有限自动机
3. 正则表达式
4. 正则语言的泵引理

   非正则语言

   泵引理

非正则语言

哪些是正则语言?

$B=\{ 0^n1^n \mid n\geq 0 \}$

$C = \{ ww \mid w\in\{0,1\}^* \}$

$D = \{ 1^k \mid k=2^n, n\geq 0 \}$

E={ w | w中0和1的个数相等 }

F={ w | w中01和10的个数相等 }

非正则语言

F={ w | w中01和10的个数相等 }是正则的：

101、010、111001001、000110110 $\in$ F

1010、0101 $\notin$ F

若干个1组成的子串(D_1)和若干个0组成的子串(D_2)交替出现

若w以(D_1)开始则以(D_1)结束，若w以(D_2)开始则以(D_2)结束

$F = ((1^+0^+)^*1^+) \cup ((0^+1^+)^*0^+)$

试着写出DFA

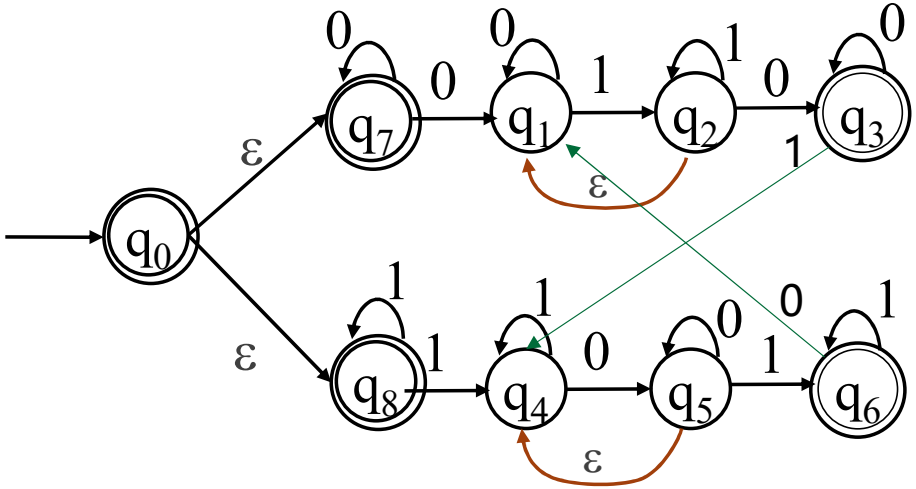
F={ w | w中01和10的个数相等 }：

设计NFA状态：

起始状态

空，0，01，010；

空，1，10，101



泵引理

定理(泵引理): 设A是正则语言,则存在p>0(泵长度)使得

对任意$w \in A$, $|w| \geq p$, 存在分割$w=xyz$满足:

1) 对任意 $i \geq 0$, $xy^iz \in A$;

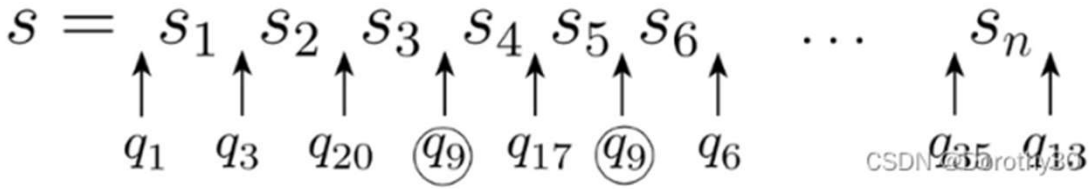
2) $|y| > 0$;

3) $|xy| \leq p$.

泵引理是一个充分非必要条件，它可以证明某个语言不是正则的，但不能证明某个语言是正则的。换句话说，如果一个语言不满足泵引理，那么它一定不是正则的；但如果一个语言满足泵引理，我们不能由此断定它是正则的，因为还有非正则语言也可能偶然满足泵引理的条件

泵引理

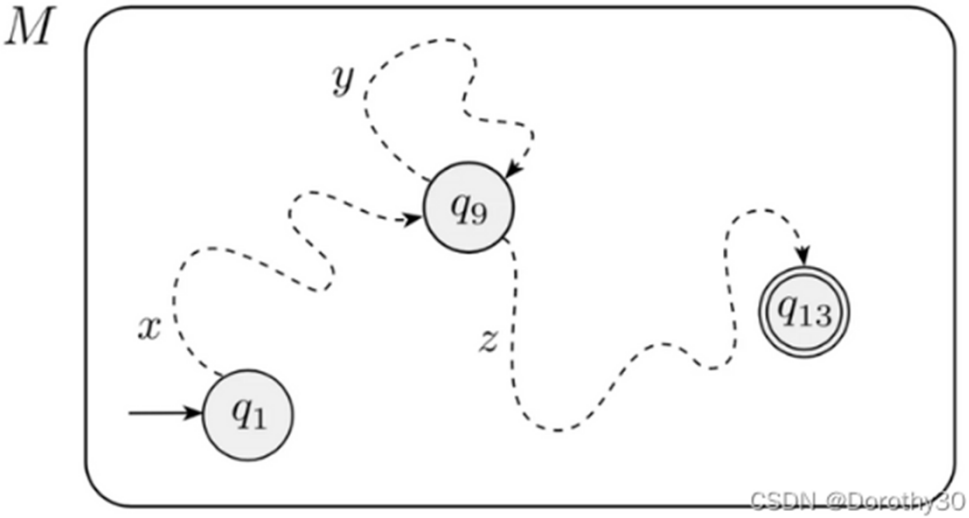
假设正则语言可以被DFA表示，设这个状态机为M，M有p个状态。我们从该正则语言中选取一个长度p(pumping length, 可理解为等价DFA的状态个数)的字符串s，且s可以被分成x，y，z三部分(s=xyz)，从状态q_1一直到状态q_{13}，如下图。



在DFA 的单状态性下，每读取一个字符，状态就会进行对应的转换，因此s的p个字符在状态机中应该要移动p次，需要p+1个状态。但是因为状态机M只有p个状态（状态之间的连线只有p-1条），根据鸽巢原理，肯定至少有两次移动在同一个状态上进行（成环）。我们可以假定这个状态为 q_9 。

泵引理

从q_1到q_8可以设为x，q_8到q_{13}为z，而第一个q_9回到q_9为y。这样子，pumping lemma的三个条件就会被满足。



条件1： $xy^iz \in A$ 。因为y无论怎么重复，都会回到q_9（回到状态机中，通过z字符串到达接收状态），所以都会被这个DFA接受。

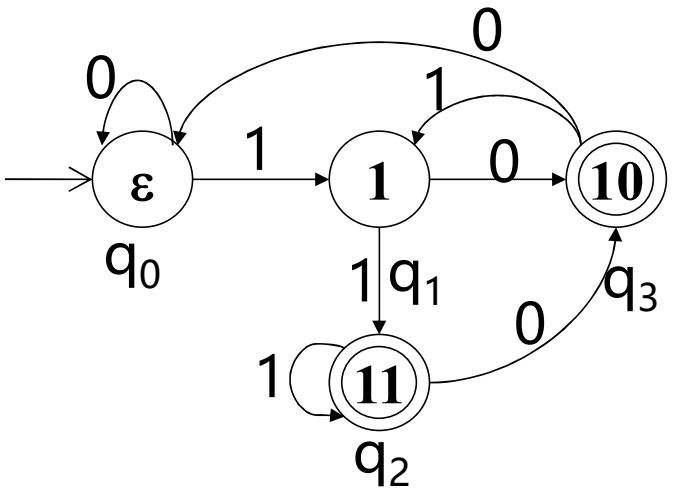
条件2：$|y| > 0$。因为y至少包含了两次，因此显然大于0

条件3： $|xy| \leq p$ 。由于$|s| = |xyz| = p$，且在需要p+1个状态进行转换的情况下DFA只有p个状态，那么作为s的子串xy（或者假设$|z|=0$），$|xy|$的长度也不会超过p。

泵引理

{ w∈{0,1}* | w倒数第2个符号是1 }

11011
1010

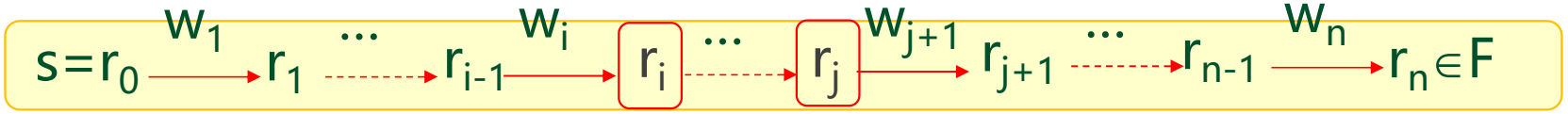


11011 : q_0-1- q_1- 101 - q_1 - 1 - q_3 -接受
1$(101)^i$1: q_0-1- q_1- $(101)^i$ - q_1 - 1 - q_3-接受
11011=xyz
x=1, y=101, z=1. xy^iz 被接受的原因?
取p为DFA状态个数.
由鸽巢原理, 读前p个符号必有状态重复

泵引理的证明

- 定理(泵引理): 设A是正则语言,则存在p>0(泵长度)使得
  对任意$w \in A$, $|w| \geq p$, 存在分割$w=xyz$满足
  1) 对任意 $i \geq 0$, $xy^iz \in A$;
  2) $|y| > 0$;
  3) $|xy| \leq p$.

证明: 令$M=(Q,\Sigma,\delta,s,F)$ 且 $L(M)=A$, 令$p=|Q|$,
设 $w = w_1w_2...w_n \in A$, $w_i \in \Sigma$, 且$n \geq p$, 则有



由鸽巢原理, 存在$i<j \leq p$使得$r_i=r_j$, 令$x=w_1...w_i$, $y=w_{i+1}...w_j$,
$z=w_{j+1}...w_n$. 那么对$\forall k \geq 0$, $xy^kz \in A$.

泵引理的等价描述

- 定理(泵引理): 设A是正则语言,则存在p>0(泵长度)使得
  对任意w∈A, |w|≥p, 存在分割w=xyz满足:
  1) 对任意 $i \geq 0$, $xy^iz \in A$;
  2) $|y| > 0$;
  3) $|xy| \leq p$.

若A是正则语言,
则∃p>0
∀w∈A(|w|≥p)
∃x,y,z(|y|>0, |xy|≤p, w=xyz)
∀i ≥ 0, $xy^iz \in A$.

若∀p>0
∃w∈A(|w|≥p)
∀x,y,z(|y|>0, |xy|≤p, w=xyz)
∃i ≥ 0, $xy^iz \notin A$.
则A非正则语言

泵引理的应用实例4

$L1 = \{ 0^m1^n \mid m,n\geq 0 \}$ 是正则语言吗?

$L2 = \{ 0^m1^n \mid m>=2,n\geq 4 \}$ 是正则语言吗?

$L3 = \{ 0^n1^n \mid n\geq 0 \}$ 是正则语言吗?

泵引理的应用实例1

$L3 = \{ 0^n1^n \mid n \geq 0 \}$ 非正则

∵ 假定L3是正则语言，那么一定

存在一个P，对任意的∀ $w \in L3(|w| \leq p)$满足泵引理。

令$w=0^p1^p = xyz$,

$\forall x,y,z(|y|>0, |xy| \leq p, w=xyz)$

因为$|xy| \leq p$，所以y只能取k个0$(1 \leq k \leq p)$

即：$y=0^k$, $x=0^{p-k}$

令i=0,

$xz = 0^{p-|y|}1^p \notin B$

∴ L3非正则语言

若$\forall p>0$

$\exists w \in A(|w| \geq p)$

$\forall x,y,z(|y|>0, |xy| \leq p, w=xyz)$

$\exists i \geq 0,$

$xy^iz \notin A.$

则A非正则语言

泵引理的应用实例2

$L5=\{ ww \mid w\in\{0,1\}^* \}$ 非正则

∵ $\forall p>0$,

令$w=0^p10^p1$

$\forall x,y,z(|y|>0,\ |xy|\le p,\ w=xyz)$

则$y=0^k$, $x=0^{p-k}(1\le k\le p)$.

令i=0,

$xz = 0^{p-|y|}10^p1 \notin C$

∴ C非正则语言

若$\forall p>0$

$\exists w\in A(|w|\ge p)$

$\forall x,y,z(|y|>0,\ |xy|\le p,\ w=xyz)$

$\exists i\ge 0$,

$xy^iz \notin A$.

则A非正则语言

泵引理的应用实例3

$L6=\{1^k \mid k=2^n, n\geq 0\}$ 非正则

$\because \forall p>0$,

令$w=1^k$, $k=2^{p+1}$,

$\forall x,y,z(|y|>0, |xy|\leq p, w=xyz)$

则$y=1^t(1\leq t\leq p)$,$x=1^{k-t}$,$z=11$

$|xyz| = 1^2P+1$

令$i=2$,

$2^{p+1}<|xy^2z|=k+|y|$,

$k+|y| \leq 2^{p+1}+ p <2^{p+2}$

即 $xy^2z \notin D$

$\therefore$ D非正则语言

若$\forall p>0$

$\exists w\in A(|w|\geq p)$

$\forall x,y,z(|y|>0, |xy|\leq p, w=xyz)$

$\exists i\geq 0$,

$xy^iz \notin A$.

则A非正则语言

泵引理的应用实例3

若$\forall p>0$
$\exists w \in A(|w| \geq p)$
$\forall x,y,z(|y|>0, |xy| \leq p, w=xyz)$
$\exists i \geq 0,$
$xy^iz \notin A.$
则A非正则语言

$L6=\{ 1^k \mid k=2^n, n \geq 0 \}$ 非正则

$\because \forall p>0,$
令$w=1^k, k=2^{p+1},$
$\forall x,y,z(|y|>0, |xy| \leq p, w=xyz)$
$w=xyz = 1^{2\wedge P}11, z=11$
$|xyz| = 1^{2\wedge P}+2 \quad >= p$
$y=1^t(1 \leq t \leq p)$,$x=1^{2\wedge p-t}$,$z=11$
令$i=2,$
$2^{p+1} < |xy^2z| = k+|y| < 2^{p+1}+t$
$\leq 2^{p+1}+ p < 2^{p+2}$
即 $xy^2z \notin D$
$\therefore$ D非正则语言

泵引理的应用实例4

$L7 = \{ 0^m1^n \mid m \geq n \}$ 是正则语言吗?

∵ 假定L7是正则语言，那么一定

存在一个P，对任意的$\forall$ w $\in$L7($|w| \leq p$)满足泵引理。

令$w=0^{p+1}1^p = xyz$, $|w|=2P+1>P$

$\forall x,y,z(|y|>0, |xy| \leq p, w=xyz)$

因为$|xy| \leq p$，所以y只能取k个0($1 \leq k \leq p$)

即：$y=0^k$, $x=0^{p-k}$

令i=0,

$xz = 0^{p+1-|y|}1^p \notin L7$

∴ L7非正则语言

泵引理的应用实例4

$L9 = \{ 0^i1^j \mid i \geq j \}$ 是正则语言吗?

泵引理

应用泵引理来证明非正则语言，一般采用反证法，需要一些创造性思维，证明的过程需要巧妙设计

$L8 = \{ a^3b^nc^{n-3} \mid n>=3 \}$ 是正则语言吗?

∵ 假定L8是正则语言，那么一定存在一个P，

对任意的$\forall$ w $\in$L7($|w| \leq p$)满足泵引理。

令$w= a^3b^Pc^{P-3} = xyz$, $|w|=2P>=P$

$\forall x,y,z(|y|>0, |xy| \leq p, w=xyz)$

y只能是下面三种情况：

1：$y=a^m$：令i=2，$xy^2w = a^{3+m}b^Nc^{N-3}$，$\notin$ L8

2：$y=b^m$: 令i=2，$xy^2w = a^3b^{N+m}c^{N-3}$，$\notin$ L8

3：$y=a^rb^s$:令i=2，$xy^2w = a^3a^rb^sb^Nc^{N-3}$，$\notin$ L8

∴ L8非正则语言

L3 = { 0^n1^n | 0<=n<=100 } 是正则的?

有限的语言都是正则语言，不需要泵引理证明

泵引理是必要条件，只能证明某个语言不是正则语言
不满足 =>不是正则
无限的语言，正则 => 满足泵引理

满足泵引理，不一定是正则的

与正则语言等价的定理：
Myhill-Nerode Theorem

下面的语言L不是正则的，但每个串都可以应用泵引理

$L = \{ca^nb^n \mid n>=1\} \cup \{c^kw \mid k\neq 1, w\in\{a,b\}^*\}$

$A = \{ca^nb^n \mid n>=1\}$ 不是正则的；

$B = \{ckw \mid k\neq 1, w \in\{a,b\}^*\}$ 是正则的；

A中的任何串，都可以应用泵引理，因为$w=(\varepsilon)(c)(a^ib^i)$
重复字符c生成的新串，最终都落入B中。

本章作业

1.1 下图给出了两台DFA M_1和M_2的状态图。

回答下述关于这两台机器的问题。

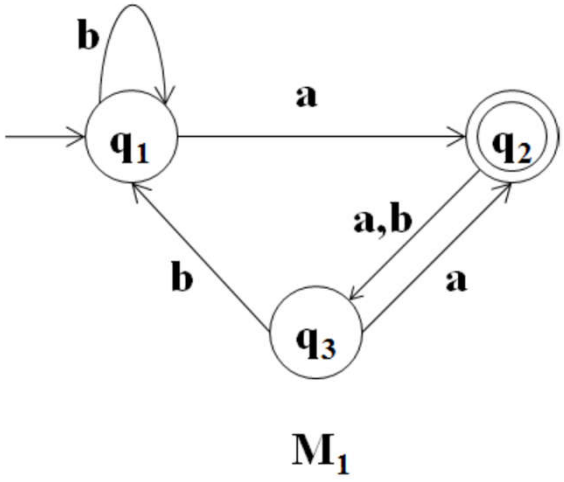
a. 它们的起始状态是什么?

b. 它们的接受状态集是什么?

c. 对输入aabb，它们经过的状态序列是什么?

d. 它们接受字符串aabb吗?

e.它们接受字符串ε吗?



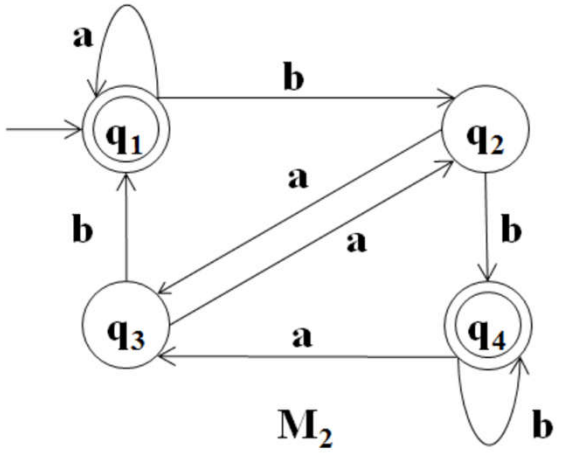
M_1

1.6 画出识别下述语言的DFA状态图。字母表为{0,1}

d. { w | w的长度不小于3, 并且第3个符号为0};

1.7. 给出下述语言的NFA，并且符合规定的状态数。

字母表为{0,1}，语言：0*1*0*0, 3个状态。

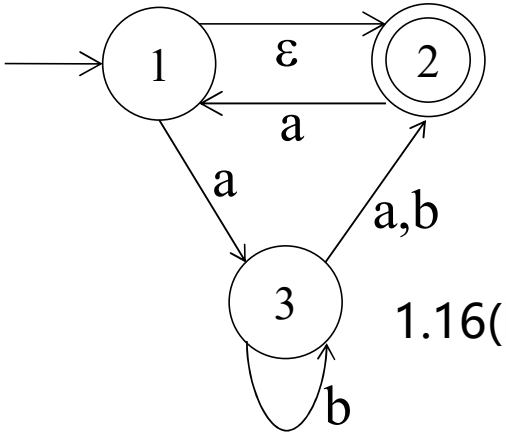


M_2

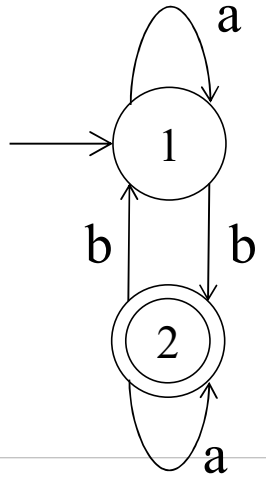
本章作业

1.16(b) 将如右图的非确定有限自动机转换成等价的确定有限自动机.

1.21(a) 将如右图的有限自动机转换成等价的正则表达式.



1.16(b)题图



1.21(a)题图

本章作业

1.22 在某些程序设计语言中, 注释出现在两个分隔符之间, 如/#和#/. 设C是所有有效注释串形成的语言. C中的成员必须以/#开始, #/结束, 并且在开始和结束之间没有#/. 为简便起见, 所有注释都由符号a和b写成; 因此C的字母表 $\Sigma=\{a, b, /, \#\}$.

a. 给出识别C的DFA.

b. 给出产生C的正则表达式.

1.29 使用泵引理证明下述语言不是正则的。

b. $A = \{ www \mid w \in \{a,b\}^* \}$

附录

在计算理论中，自动机主要被用来模拟和分析计算问题的可解性和复杂性。例如，有限自动机（finite automata）被用来描述和理解正则语言，图灵机（Turing machine）则被用来模拟任意的算法过程。

在编译器设计中，自动机被用于实现词法分析和语法分析，识别和解析编程语言中的语法结构。例如，有限自动机被用于实现词法分析器，将源代码分割为一系列的词素；推导自动机（pushdown automata）则被用于实现语法分析器，检查词素序列是否满足语法规则。

在人工智能和解析学中，自动机被用于实现状态机，描述和模拟复杂系统的行为。例如，在游戏编程中，状态机被用于描述和管理游戏角色的各种状态和行为。

在正则表达式中，自动机被用于实现模式匹配，识别和提取文本中满足特定模式的部分。例如，有限自动机被用于实现正则表达式的匹配引擎，检查文本是否满足正则表达式描述的模式。

字符串匹配问题

输入: 两个字符串T(ext),P(attern), (|T|=n, |P|=m)

输出: 所有P在T中出现的起点位置

例: T=abaababababbababbabababaa, P=abababababaa

输出13

直接法: 以每个位置为起点对比一遍P. 时间?

O((n-m+1)m). 能否利用已经看到的信息?

动态规划: 子结构[1:i], 决策量?

决策量设为T[1:i]的能成为P前缀的最大后缀(长度)

这就是字符串匹配的自动机算法

字符串匹配的自动机算法

动态规划: 子结构[1:i], 决策量?

决策量设为T[1:i]的能成为P前缀的最大后缀(长度)

这就是字符串匹配的自动机算法

令 $P_j = p_1 p_2 \ldots p_j$, $1 \le j \le m$, $P_0 = \varepsilon$ //代表状态0~m

转移函数: $\delta(j, a) = \max\{ k \mid P_k \text{ is a suffix of } P_j a \}$

是递推关系

definition of prefix function

Σ, Σ^*, prefix, suffix

P = $p_1p_2...p_m \in \Sigma^*$ a pattern, $P_j = p_1p_2...p_j$ $1 \le j \le m$, $P_0 = \varepsilon$

The transition function for P, $0 \le j \le m$, $a \in \Sigma$,

$$\delta(j,a) = \max \{ k \mid P_k \text{ is a suffix of } P_j a \} \quad // \ge 0$$

The prefix function for P, $1 \le j \le m$,

$$\pi(j) = \max\{ k \mid k<j, P_k \text{ is a suffix of } P_j \} \quad // \ge 0$$

example of prefix function

Σ = {a,b,c}, P =ababaca

$\pi(j)$ = max{ $k \mid k<j$, P_k is a suffix of P_j },

$\pi(1)$ = ? P_1 = a,

real suffixes of P_1: $\varepsilon = P_0$,

$\pi(5)$ = ? P_5 = ababa,

real suffixes of P_5: $\varepsilon = P_0$, a=P_1, ba, aba=P_3, baba

i	1	2	3	4	5	6	7
P[i]	a	b	a	b	a	c	a
π[i]	0	0	1	2	3	0	1

i	…	4	5	6	7	8	9	10	…		
T[i]	…	a	b	a	b	a	a	a	…		
P_5		a	b	a	b	a	c	a			π(5)=3
P_3				a	b	a	b	a	c		π(3)=1
P_1						a	b	a	b	…	π(1)=0
P_0						ε	a	b	a	…	

computation of prefix function

Let P be a pattern with length m

$\pi(q) = \max\{k \mid k<q, P_k \text{ is a suffix of } P_q\}$,

ComputePF(P, Σ, m)

1. $\pi[1]=0$, k = 0
2. For q = 2 : m // O(m)
3. while k>0 and P[k+1] $\neq$ P[q], k=π[k] //totally O(m)
4. If P[k+1] == P[q], k=k+1 //totally O(m)
5. π[q]=k

time complexity O(m) ? //aggregate analysis

KMP matcher

Let π be the transition function for a pattern P,
T[1:n] = $t_1 t_2 \ldots t_n$ be a text
KMPMatch(T, P, n, m) //m is the length of P
1. q = 0
2. For i = 1 : n // O(n)
3. while q>0 and P[q+1] $\neq$ T[i], q=π[q] //totally O(n)
4. If P[q+1] == T[i], q=q+1 //totally O(n)
5. If q == m, Print(i-m); q = π[q]
// print all place that P occur
time complexity O(n) ? //aggregate analysis

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] $\neq$ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q	0														
pattern	a	b	c	d	a	b	d								
q	1														

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] ≠ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q	0	1													
pattern	a	b	c	d	a	b	d								
q		2													

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] ≠ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q	0	1	2												
pattern	a	b	c	d	a	b	d								
q			3												

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] $\neq$ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q	0	1	2	3											
pattern	a	b	c	d	a	b	d								
q				0											
pattern				a	b	c	d	a	b	d					
q															

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] ≠ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q															
pattern															
q				0											
pattern				a	b	c	d	a	b	d					
q				1											

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] $\neq$ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q				0	1										
pattern				a	b	c	d	a	b	d					
q					2										

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] ≠ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q				0	1	2									
pattern				a	b	c	d	a	b	d					
q						3									

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] $\neq$ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q				0	1	2	3								
pattern				a	b	c	d	a	b	d					
q							4								

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] ≠ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q				0	1	2	3	4							
pattern				a	b	c	d	a	b	d					
q								5							

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] $\neq$ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q				0	1	2	3	4	5	6					
pattern				a	b	c	d	a	b	d					
q										2					
								a	b	c	d	a	b	d	

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] ≠ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q															
pattern															
q										2					
pattern								a	b	c	d	a	b	d	
q										3					

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] ≠ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q										2	3				
pattern								a	b	c	d	a	b	d	
q											4				
pattern															
q															

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] $\neq$ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q										2	3	4			
pattern								a	b	c	d	a	b	d	
q												5			
pattern															
q															

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] ≠ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q										2	3	4	5		
pattern								a	b	c	d	a	b	d	
q													6		
pattern															
q															

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] ≠ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q										2	3	4	5	6	
pattern								a	b	c	d	a	b	d	
q														7	
q														0	
pattern															a

example of KMP

Σ = {a,b,c,d,e}, P =abcdabd, T=abcabcdabcdabde

i	1	2	3	4	5	6	7
P[i]	a	b	c	d	a	b	d
π[i]	0	0	0	0	1	2	0

1. q = 0
2. For i = 1 : n
3. while q>0 and P[q+1] $\neq$ T[i], q=π[q]
4. If P[q+1] == T[i], q=q+1
5. If q == m, Print(i-m); q = π[q]

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T[i]	a	b	c	a	b	c	d	a	b	c	d	a	b	d	e
q															0
pattern															a
q															0