形式语言与自动机

课程复习

计算机科学与技术学院
哈尔滨工业大学（深圳）

郑宜峰

二、有穷状态自动机

(1) DFA 的定义与构造

(2) NFA 的定义与构造

(3) ε-NFA 的定义与构造

(4) DFA 与 NFA 的等价性

(5) DFA 最小化问题

- **定义：确定的有穷自动机（DFA, Deterministic Finite Automaton）** A**为五元组**$(Q, \Sigma, \delta, q_0, F)$

(1) Q:有穷状态集；

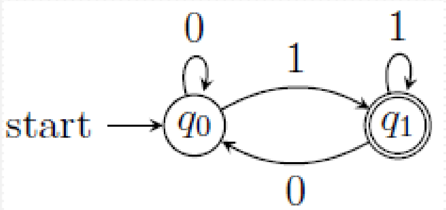
(2) Σ:有穷输入符号集或字母表；(a finite set of input symbols/alphabet)

(3) $\delta: Q \times \Sigma \longrightarrow Q$:状态转移函数(a transition function)；

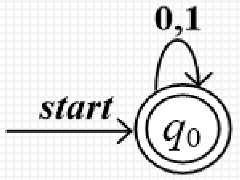
(4) $q_0 \in Q$: 初始状态(a start state)；

(5) $F \subseteq Q$: 终结状态集或接受状态集(a set of accepting states)。

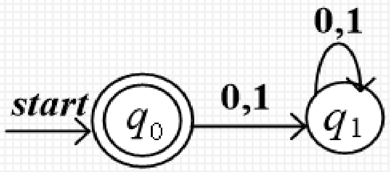
- **设计一个DFA，使其接受且仅接受给定的语言L(符号串的集合)。**
  - Σ={0,1}, 给出接受全部以1 结尾的串的DFA



  - Σ={0,1}, 给出接受Σ*的DFA



  - Σ={0,1}, 给出接受{ε}的DFA



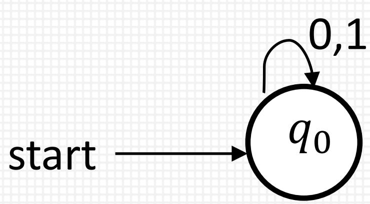
DFA的语言与正则语言

- **定义：若$D=(Q,\Sigma,\delta,q_0,F)$是一个DFA，则$D$识别的语言为**

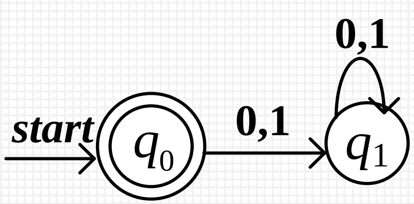
$$L(D)=\{w\in\Sigma^* | \hat{\delta}(q_0,w)\in F\}$$

即所有能使DFA从开始状态到达接受状态的符号串集合。

- **定义：如果语言L是某个DFA D识别的语言，则称其为正则语言。**
  - $\emptyset$，$\{\varepsilon\}$都是正则语言



$\emptyset$的DFA: 无接受状态，即该DFA不接受任何符号串（包括空串)。



识别{ ε }的DFA。

非确定有穷自动机的形式定义

- **定义：非确定有穷自动机（NFA, Non-deterministic Finite Automaton）** A **为五元组** $(Q, \Sigma, \delta, q_0, F)$

（1） Q: **有穷状态集；**

（2） Σ: **有穷输入符号集或字母表；**

（3） $\delta: Q \times \Sigma \longrightarrow 2^Q$: **状态转移函数；**

（4） $q_0 \in Q$: **初始状态**

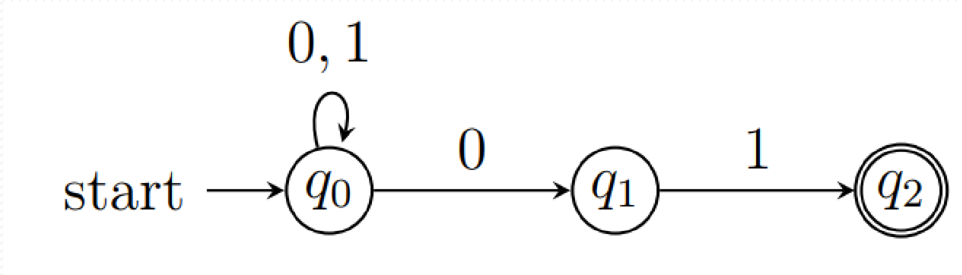
（5） $F \subseteq Q$: **终结状态集或接受状态集。**

注: 2^Q 是幂集，即 $2^Q = \{S|S \subset Q\}$ (S是Q的子集)

$\delta(q, a) = \{p_1, p_2, \ldots, p_n\}$

NFA的形式定义示例

- 例：接受全部以 01 结尾的串的NFA。



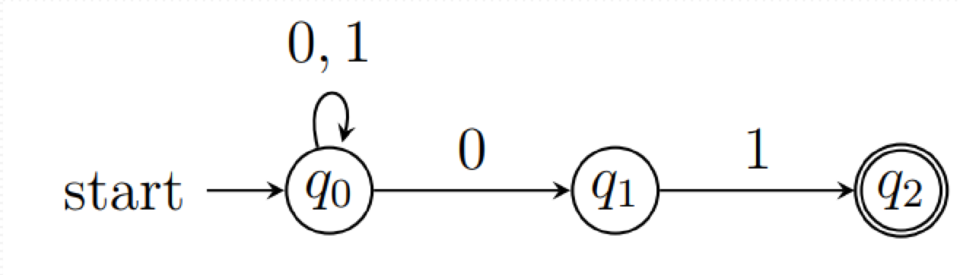
五元组为 $A=(\{q_0,q_1,q_2\},\{0,1\},\delta,q_0,\{q_2\})$ ，转移函数δ：

$$\delta(q_0,0)=\{q_0,q_1\} \quad \delta(q_1,0)=\varnothing \quad \delta(q_2,0)=\varnothing$$

$$\delta(q_0,1)=\{q_0\} \quad \delta(q_1,1)=\{q_2\} \quad \delta(q_2,1)=\varnothing$$

NFA的形式定义示例

- **续例：接受全部以 01 结尾的串的NFA。**



- **状态转移表：**

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$

NFA的语言

- **回顾:若**$D = (Q, \Sigma, \delta, q_0, F)$**是一个DFA, 则**$D$**接受的语言为**
$$L(D) = \{w \in \Sigma^* | \hat{\delta}(q_0, w) \in F\}$$

- **定义:若**$N = (Q, \Sigma, \delta, q_0, F)$**是一个NFA, 则**$N$**接受的语言为**
$$L(N) = \{w \in \Sigma^* | \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

DFA和NFA对比

DFA	NFA

• 状态的转移是确定的
  • 一次状态转移只会转移到一个确定的状态
	• 状态的转移是确定的
	  • 一次状态转移可能转移到多个状态

• 对于每一个状态，需要明确定义对应于各个可能的输入符号的状态转移	• 不需要明确地定义所有的状态转移（如果未定义的话那么会进入一个dead state）
• 有时设计较复杂	• 通常比DFA更为容易设计
• 最后所处的状态在接受状态集F中时，才接受符号串。	• 最后所处的状态集中有某个状态在接受状态集F中时，才接受符号串。

但是，DFA和NFA表示语言的能力是等价的。

DFA与NFA的等价性

- 定理:如果语言L被NFA接受，当且仅当L被DFA接受。
- 证明:给定一个NFA的五元组，构造一个DFA，然后证明它们是等价的。通过子集构造法（subset construction）进行。

从NFA到DFA：子集构造法

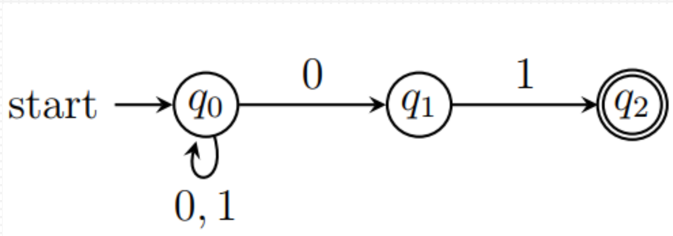
- **目标：NFA** $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$**构造** $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$，且$\mathbf{L}(D) = \mathbf{L}(N)$。
- **构造:**
  - 状态集： $Q_D = 2^{Q_N}$,即每个状态是Q_N的子集
  - 接受状态集：$F_D = \{S | S \subset Q_N, S \cap F_N \neq \emptyset\}$ (set of subsets S of Q_N)
  - 状态转移函数: $\forall S \subset Q_N, \forall a \in \Sigma, \delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$。

例如，假设DFA的某个状态为$\{q_1, q_2, q_3\}$，输入符号a,那么$\delta_D(\{q_1, q_2, q_3\}, \text{a})$是$\delta_N(q_i, a)$的并集($i = 1,2,3$)。

（注意，这里$\{q_1, q_2, q_3\}$表示DFA的一个状态，是状态名，而不是看作一个集合。）

示例

- **例:构造接受全部以 01 结尾的串的NFA。**



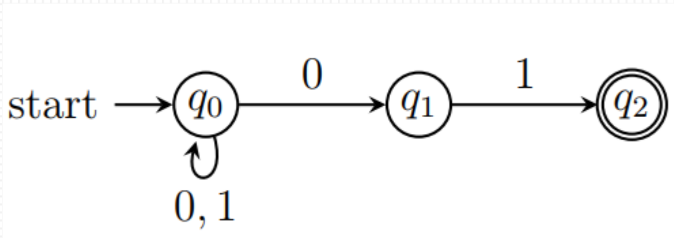
	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$

1. 列出DFA的所有可能状态集(为NFA状态集的所有子集)。

	0	1
$\emptyset$		
$\rightarrow \{q_0\}$		
$\{q_1\}$		
$\{q_2\}$		

	0	1
$\{q_0, q_1\}$		
$\{q_0, q_2\}$		
$\{q_1, q_2\}$		
$\{q_0, q_1, q_2\}$		

- **例:构造接受全部以 01 结尾的串的NFA。**



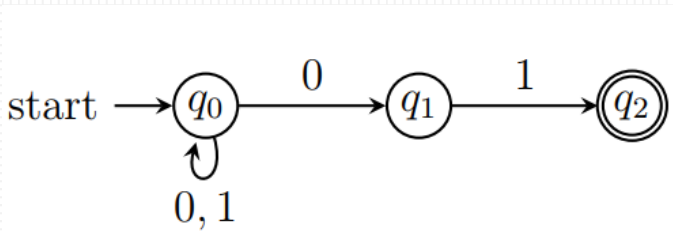
	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$

2. 转移状态为NFA中相应状态的并集。

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$\{q_2\}$	$\emptyset$	$\emptyset$

	0	1
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

- **例：构造接受全部以 01 结尾的串的NFA。**



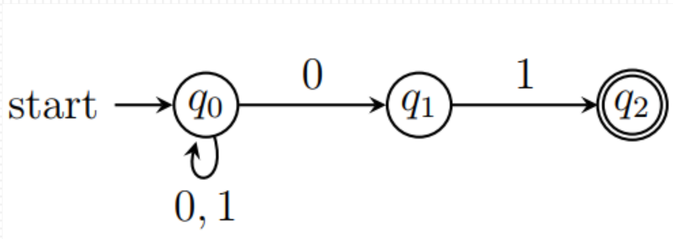
	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$^*q_2$	$\emptyset$	$\emptyset$

3. 删除含有∅的状态，删除不能从开始状态达到的状态。

	0	1
$\cancel{\emptyset}$	$\cancel{\emptyset}$	$\cancel{\emptyset}$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\cancel{\{q_1\}}$	$\cancel{\emptyset}$	$\cancel{\{q_2\}}$
$\cancel{\{q_2\}}$	$\cancel{\emptyset}$	$\cancel{\emptyset}$

	0	1
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\cancel{\{q_1, q_2\}}$	$\cancel{\emptyset}$	$\cancel{\{q_2\}}$
$\cancel{\{q_0, q_1, q_2\}}$	$\cancel{\{q_0, q_1\}}$	$\cancel{\{q_0, q_2\}}$

- **例:构造接受全部以 01 结尾的串的NFA。**



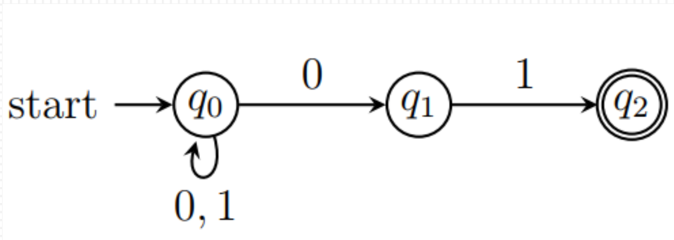
	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$

4. 含有NFA接受状态的状态集就是DFA的接受状态。

	0	1
~~$\emptyset$~~	~~$\emptyset$~~	~~$\emptyset$~~
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
~~$\{q_1\}$~~	~~$\emptyset$~~	~~$\{q_2\}$~~
~~$\{q_2\}$~~	~~$\emptyset$~~	~~$\emptyset$~~

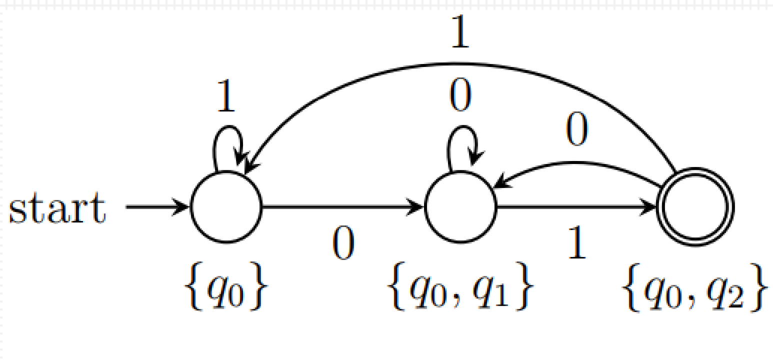
	0	1
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
~~$\{q_1, q_2\}$~~	~~$\emptyset$~~	~~$\{q_2\}$~~
~~$\{q_0, q_1, q_2\}$~~	~~$\{q_0, q_1\}$~~	~~$\{q_0, q_2\}$~~

- **例:构造接受全部以 01 结尾的串的NFA。**



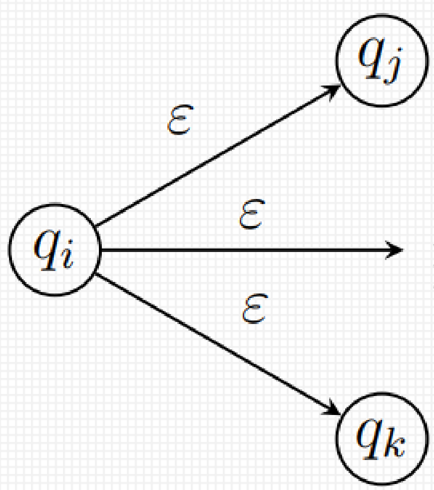
	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$*q_2$	$\emptyset$	$\emptyset$

4. 含有NFA接受状态的状态集就是DFA的接受状态。



带有空转移的NFA（ε-NFA）

- **ε-NFA：对NFA进行扩展，在不输入任何字符的情况下（空串ε），可以自发地发生状态转移。**
  - 可以简化NFA的构造
  - ε-NFA是明确地定义了至少一个空转移的NFA
  - 在状态表里面多了对应于ε的一列



ε-NFA的形式化定义

- **定义：带空转移非确定有穷自动机（ε-NFA）A 为五元组$A = (Q, \Sigma, \delta, q_0, F)$。**

（1） Q**:有穷状态集；**

（2） Σ: **有穷输入符号集或字母表；**

（3） $\delta: Q \times (\Sigma \cup \varepsilon) \longrightarrow 2^Q$**:状态转移函数；**

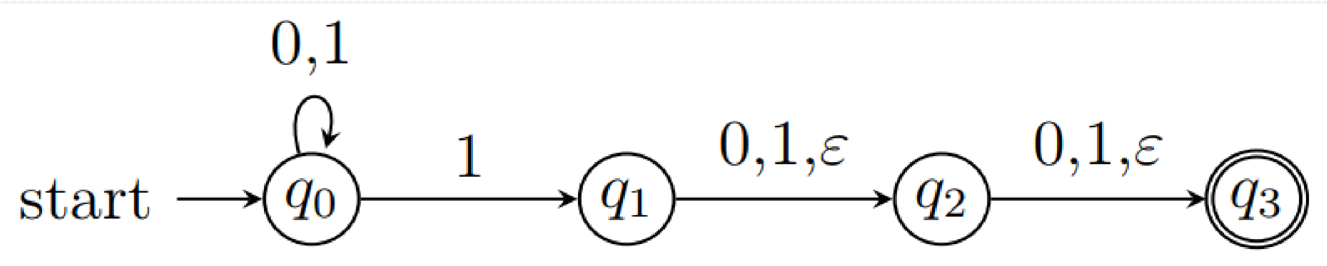
（4） $q_0 \in Q$: **初始状态**

（5） $F \subseteq Q$: **终结状态集或接受状态集。**

ε-NFA示例

- **例**: $L = \{w \in \{0,1\}^* \mid w$ 倒数3个字符至少有一个是1$\}$**的NFA。**

ε-NFA:



- 先在q_0输入消耗前面所有的字符，
  — 只要倒数3个字符里面至少有一个1，总能先跳到q_1，这样即使字符已经消耗完，也可以通过空转移跳到q_3，进入接受状态。

- **回顾：DFA** $D=(Q,\Sigma,\delta,q_0,F)$**和NFA** $N=(Q,\Sigma,\delta,q_0,F)$**的语言分别为**

$$\boldsymbol{L}(D)=\{w\in\Sigma^*|\hat{\delta}(q_0,w)\in F\}$$

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

- **定义:若**$E=(Q,\Sigma,\delta,q_0,F)$**是一个**$\varepsilon$**-NFA，则**$E$**接受的语言为**

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

- **定理:如果语言L被ε-NFA接受,当且仅当L被DFA接受。**

构造与ε-NFA等价的DFA

- **子集构造法（消除空转移）：如果ε-NFA** $E = (Q_E, \Sigma, \delta_E, q_E, F_E)$，**构造DFA**

$$D = (Q_D, \Sigma, \delta_D, q_D, F_D)$$

(1) $Q_D = 2^{Q_E}$

(2) $q_D = \mathrm{ECLOSE}(q_E)$

(3) $F_D = \{S | S \subseteq Q_E, S \cap F_E \neq \emptyset\}$

(4) $\forall S \subseteq Q_E, \forall a \in \Sigma,$

$$\delta_D(S, a) = \mathrm{ECLOSE}(\cup_{p \in S} \delta_E(p, a))$$

那么$\boldsymbol{L}(D) = \boldsymbol{L}(E)$。

小结

- 有穷自动机基本概念
- 三种识别正则语言的有穷自动机
  - 确定有穷自动机（DFA）
  - 非确定有穷自动机（NFA）
  - 带有空转移的有穷自动机（ε-NFA）
- 构造与NFA和ε-NFA等价的DFA的方法

三、正则表达式

(1)正则表达式定义与构造

(2)正则表达式与FA的等价性

(3)正则表达式的应用

正则表达式的递归定义

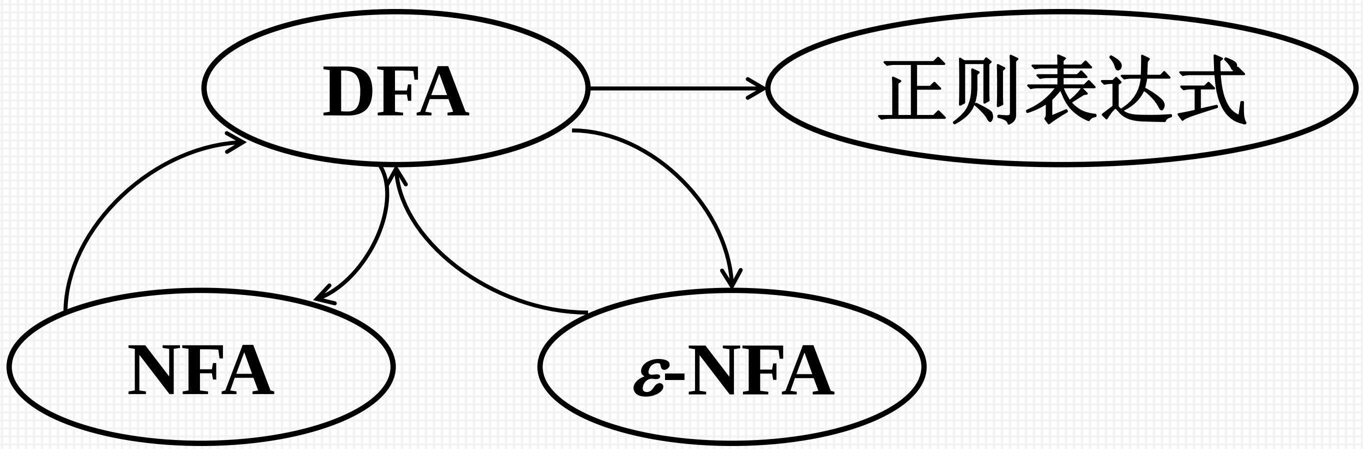
- 基础1：$\emptyset$是一个正则表达式，表示空语言$\emptyset$ 。
- 基础2：ε是一个正则表达式，表示语言$\{\varepsilon\}$。
- 基础3：对于任意一个符号a，a是一个正则表达式，表示语言$\{a\}$，其有一个长度为1的字符串。

正则表达式的递归定义

- **归纳1**：如果E_1和E_2是正则表达式，那么$E_1 + E_2$也是正则表达式，且$\mathrm{L}(E_1 + E_2) = \mathrm{L}(E_1) \cup \mathrm{L}(E_2)$。
- **归纳2**：如果E_1和E_2是正则表达式，那么E_1E_2也是正则表达式，且$\mathrm{L}(E_1E_2) = \mathrm{L}(E_1)\mathrm{L}(E_2)$。
- **归纳3**：如果E是正则表达式，则E^*也是正则表达式。
- **归纳4**:如果E是正则表达式，则(E)也是正则表达式，表示语言$\mathrm{L}(E)$。

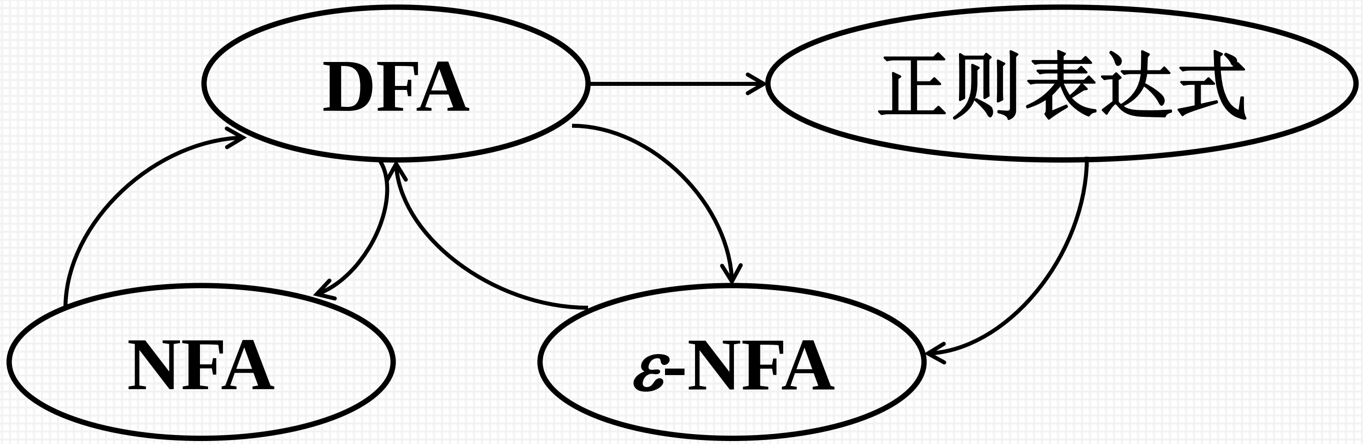
有穷自动机和正则表达式

- **DFA, NFA, ε-NFA和正则表达式在表示语言的能力上是等价的**
  - 相互之间的转换关系



对于一个DFA所能识别的语言，都存在一个正则表达式表示它。

- **DFA, NFA, ε-NFA和正则表达式在表示语言的能力上是等价的**
  - 相互之间的转换关系



任何一个正则表达式表示的语言，都可以由一个ε-NFA识别。

DFA到正则表达式

- **定理：若 $L = \mathbf{L}(A)$ 是某 DFA A 的语言，那么存在正则表达式R满足$L = \boldsymbol{L}(R)$。**
- **证明方法：**
  - 递归法
  - 状态消除法

DFA->正则表达式

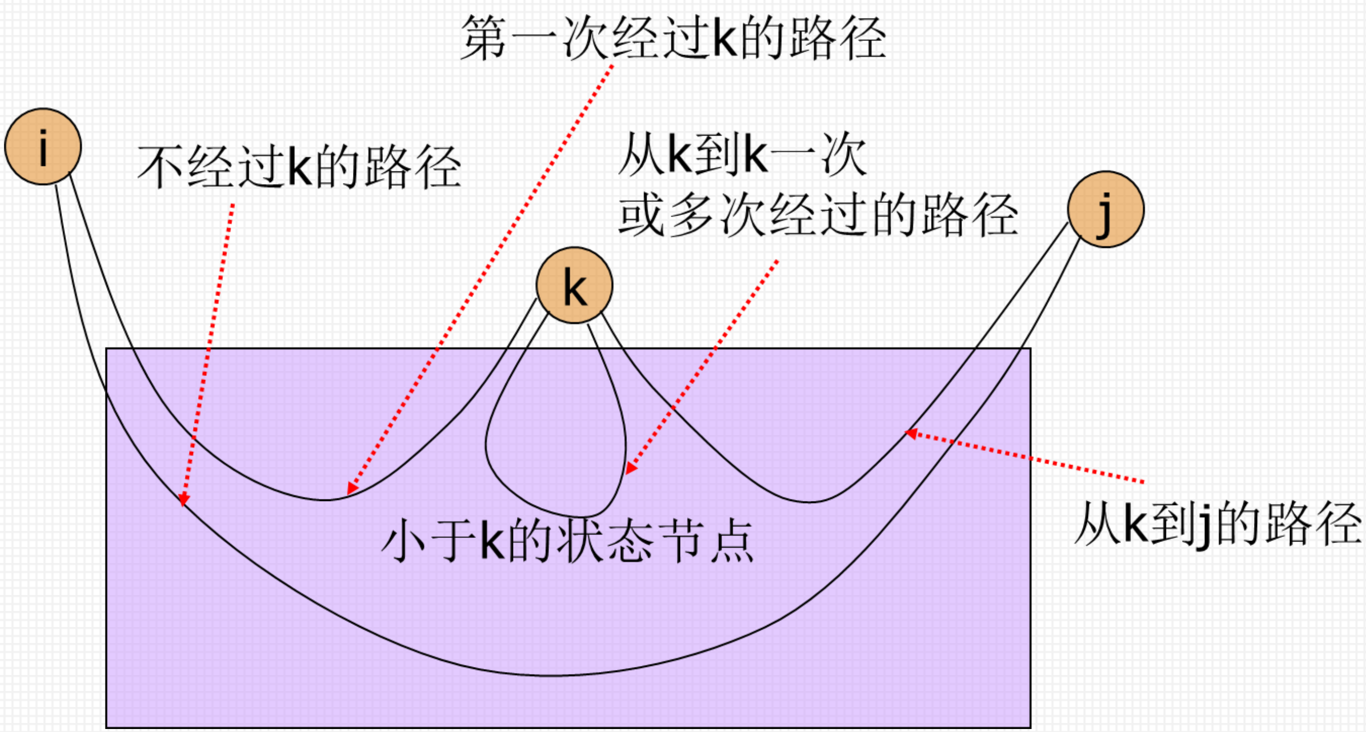
- **如果1是开始结点，则和DFA等价的正则表达式就是**

$$\bigcup_{j \in F} R_{1,j}^{(n)}$$

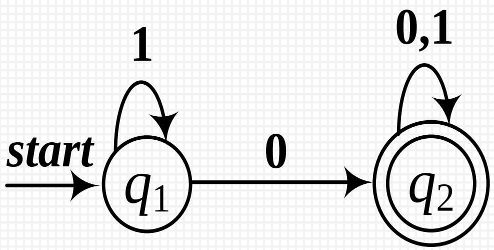
即，从状态1到终结状态的所有路径的集合。

递归关系示意图

- **递归关系：** $R_{i,j}^{(k)} = R_{i,j}^{(k-1)} + R_{i,k}^{(k-1)}\ (R_{k,k}^{(k-1)})^{*} R_{k,j}^{(k-1)}$

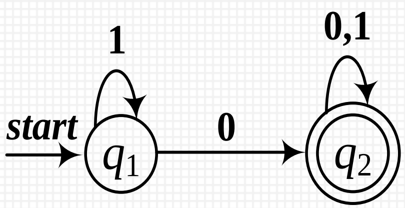


- 例:把下图所示DFA转换为正则表达式。这个DFA接受至少含有一个0的串。



- 状态总数$n = 2$，状态1为开始状态，状态2为结束状态，所以需要求解$R_{1,2}^{(2)}$

- 把下图所示DFA转换为正则表达式。这个DFA接受至少含有一个0 的串。



$k=0$的情况(状态i直接到状态j):

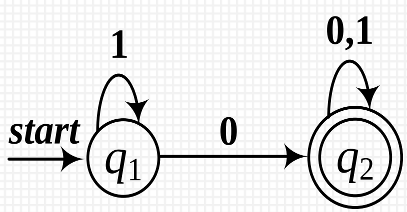
$R_{1,1}^{(0)}$	$\varepsilon+1$
$R_{1,2}^{(0)}$	0
$R_{2,1}^{(0)}$	$\emptyset$
$R_{2,2}^{(0)}$	$\varepsilon+0+1$

$R_{1,1}^{(1)}$	$\varepsilon+1+(\varepsilon+1)(\varepsilon+1)^*(\varepsilon+1)$
$R_{1,2}^{(1)}$	$0+(\varepsilon+1)(\varepsilon+1)^*0$
$R_{2,1}^{(1)}$	$\emptyset+\emptyset(\varepsilon+1)^*(\varepsilon+1)$
$R_{2,2}^{(1)}$	$\varepsilon+0+1+\emptyset(\varepsilon+1)^*0$

$R_{1,1}^{(2)}$	$1^*+1^*0(\varepsilon+0+1)^*\emptyset$
$\boldsymbol{R_{1,2}^{(2)}}$	$1^*0+1^*0(\varepsilon+0+1)^*(\varepsilon+0+1)$
$R_{2,1}^{(2)}$	$\emptyset+(\varepsilon+0+1)\ (\varepsilon+0+1)^*\emptyset$
$R_{2,2}^{(2)}$	$\varepsilon+0+1+(\varepsilon+0+1)(\varepsilon+0+1)^*(\varepsilon+0+1)$

DFA->正则表达式示例

- 把下图所示DFA转换为正则表达式。这个DFA接受至少含有一个0 的串。



简化规则：

$(\varepsilon+E)^*=E^*$

$E_1+E_1E_2{}^*=E_1E_2{}^*$

$\emptyset E_1 = E_1\emptyset = \emptyset$（零元）

$\emptyset E_1 = E_1 + \emptyset = E_1$ （单位元）

$R_{1,1}^{(0)}$	$\varepsilon + 1$
$R_{1,2}^{(0)}$	0
$R_{2,1}^{(0)}$	$\emptyset$
$R_{2,2}^{(0)}$	$\varepsilon + 0 + 1$

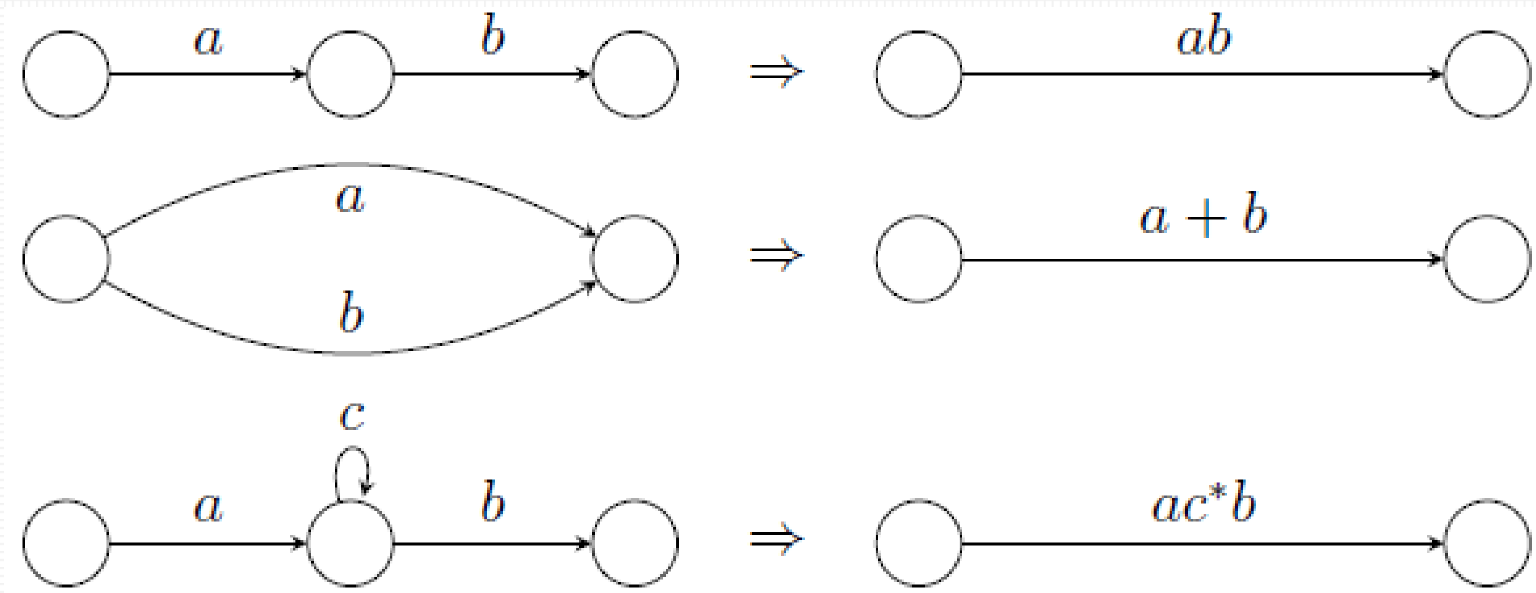
$R_{1,1}^{(1)}$	$\varepsilon + 1 + (\varepsilon + 1)(\varepsilon + 1)^*(\varepsilon + 1)$
$R_{1,2}^{(1)}$	$0 + (\varepsilon + 1)(\varepsilon + 1)^*0$
$R_{2,1}^{(1)}$	$\emptyset + \emptyset(\varepsilon + 1)^*(\varepsilon + 1)$
$R_{2,2}^{(1)}$	$\varepsilon + 0 + 1 + \emptyset(\varepsilon + 1)^*0$

$R_{1,1}^{(2)}$	$1^* + 1^*0(\varepsilon + 0 + 1)^*\emptyset$
$\boldsymbol{R_{1,2}^{(2)}}$	$\mathbf{1^*0(0 + 1)^*}$
$R_{2,1}^{(2)}$	$\emptyset + (\varepsilon + 0 + 1)\ (\varepsilon + 0 + 1)^*\emptyset$
$R_{2,2}^{(2)}$	$\varepsilon + 0 + 1 + (\varepsilon + 0 + 1)(\varepsilon + 0 + 1)^*(\varepsilon + 0 + 1)$

DFA->正则表达式:状态消除法

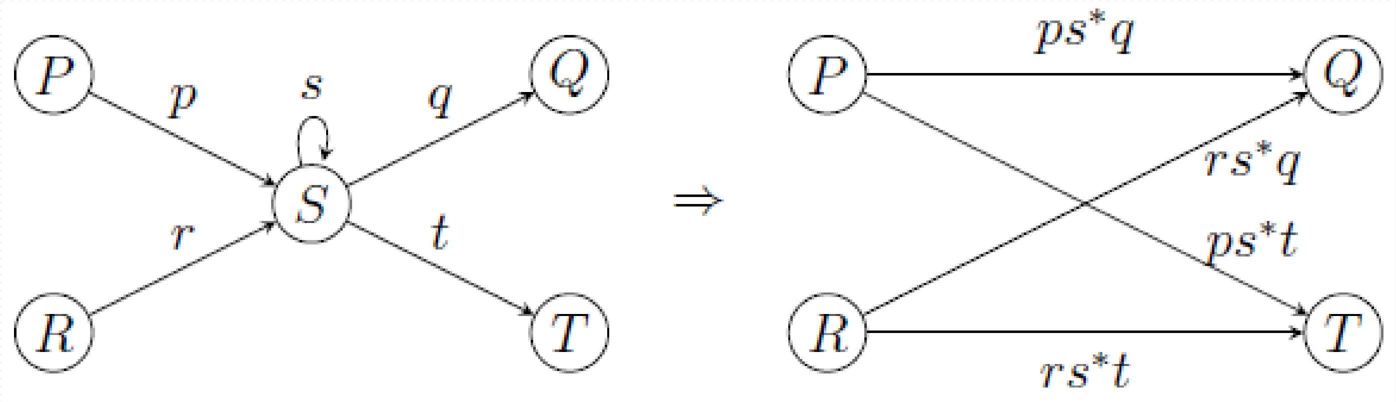
- **从DFA中逐个删除状态**
  - 会消除掉状态的原有路径，因此要补上等价的路径
- **用标记了正则表达式的新路径替换被删掉的路径**
- **保持“自动机”等价**

- 正则表达式运算（连接，加，闭包）和DFA中路径的关系



状态消除法

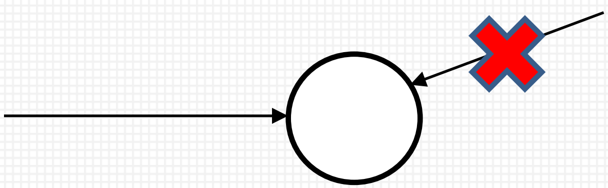
- **更一般的情况，要为被删除的状态S的每个“入”和“出”路径的组合，补一条等价的新路径，并用新的正则表达式表示**

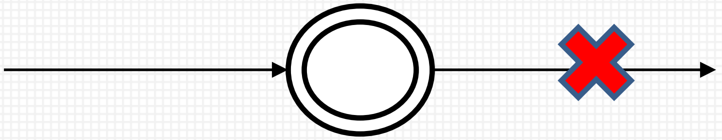


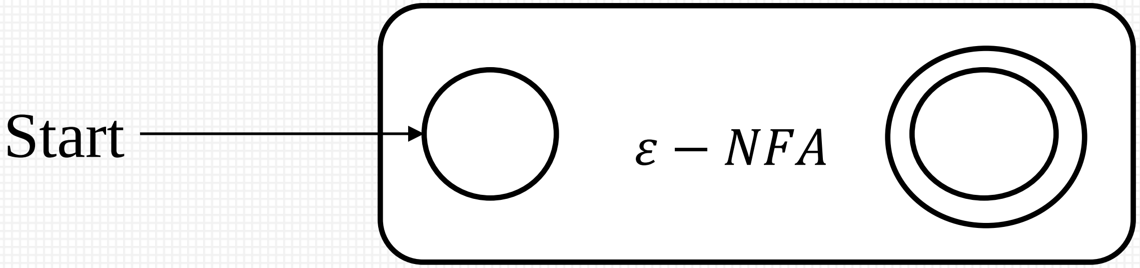
- **2条入，2条出，要补4条（P到Q，P到T，R到Q，R到T）**

正则表达式到自动机

- **定理:每个正则表达式定义的语言，都可被有穷自动机识别。**
- **⇔任何正则表达式R，都存在与其等价的ε−NFA A, 即$\mathbf{L}(A) = \mathbf{L}(R)$，并且$A$满足:**
  - 仅有一个接受状态;
  - 没有进入开始状态的边;
  - 没有离开接受状态的边。

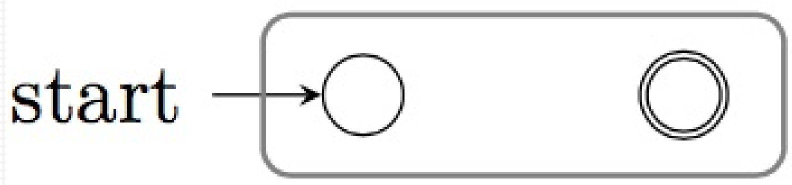




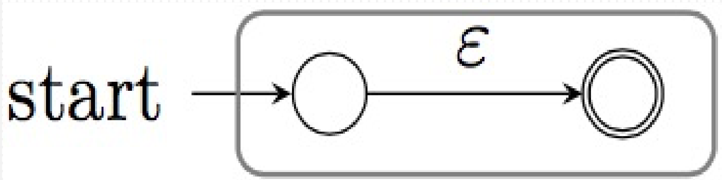


- **归纳基础：**

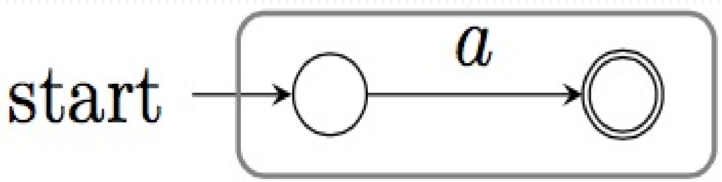
1. 对正则表达式∅，有ε−NFA



2. 对正则表达式ε，有ε−NFA



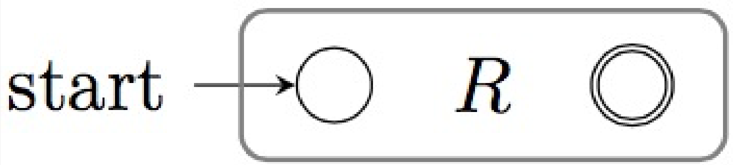
3. $\forall a \in \Sigma$，对正则表达式**a**，有ε−NFA

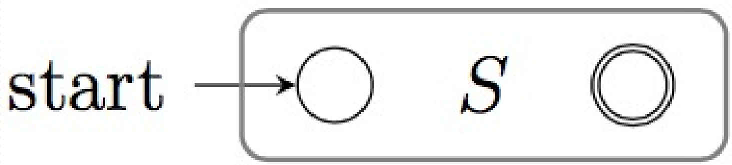


正则表达式到自动机

- **归纳递推：**

假设若r和s为正则表达式，则它们对应的ε−NFA分别为R和S。





构造$r+s, rs, r^*$的自动机。

小结

- **正则表达式基本概念**
- **正则表达式与有穷自动机的转换**
  - 正则表达式→DFA
  - DFA→正则表达式：构造法＋状态消除法

四、正则语言的性质

（1）正则语言的泵引理及证明

（2）用泵引理证明某些语言不是正则语言

（3）正则语言的封闭性

（4）正则语言的判定问题

- **泵引理(pumping lemma)：如果语言L是正则的，那么存在正整数 N，对$w \in L$, 只要$|w| \geq N$，就可以将w分为三部分$w = xyz$ 满足:**
  - $y \neq \varepsilon$（或$|y| > 0$）；
  - $|xy| \leq N$;
  - $\forall k \geq 0, xy^k z \in L$;
    - 中间的串不论循环几次，所得的串仍属于该正则语言。

泵引理的应用示例

- **例1：证明$L_{01} = \{0^n 1^n | n \geq 0\}$不是正则语言。**
- **证明：假设L_{01}是正则的，**
  - 那么一定存在正整数N，对$w \in L_{01}(|w| \geq N)$满足泵引理。
  - 从L_{01}中取$w = 0^N 1^N$，显然$w \in L_{01}$,且$|w| = 2N > N$。
  - 那么, w可被分为$w = xyz$，且$|xy| \leq N$和$y \neq \varepsilon$。
  - 因此，y只能是0^m且$m > 0$。
  - 那么$xy^2z = 0^{N+m}1^N \notin L_{01}$，而由泵引理，$xy^2z \in L_{01}$，矛盾。

所以假设不成立，L_{01}不是正则的。

正则语言的封闭性

- **正则语言经某些运算后得到的新语言仍保持正则，称正则语言在这些运算下封闭。**
- **正则语言L和M，在这些运算下封闭：**
  - 并: $L \cup M$，连接: $L \cdot M$，闭包: L^*
  - 交: $L \cap M$
  - 反转: $L^R = \{w^R | w \in L\}$
  - 同态: $h(L) = \{h(w) | w \in L, 同态: h: \Sigma \to \Gamma^*\}$
  - 逆同态: $h^{-1}(L) = \{w \in \Sigma^* | h(w) \in L \subset \Gamma^*, 同态: h: \Sigma \to \Gamma^*\}$
  - 补: $\overline{L}$
  - 差: $L - M$

状态的等价性

- **定义：DFA** $A = (Q, \Sigma, \delta, q_0, F)$**中两个状态** p **和** q**, 对** $\forall w \in \Sigma^*$**:**

$$\hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$$

则称这两个状态是等价的，否则称为可区分的。

即，从p**开始而被接受的串w, 从**q**开始也会被接受。**

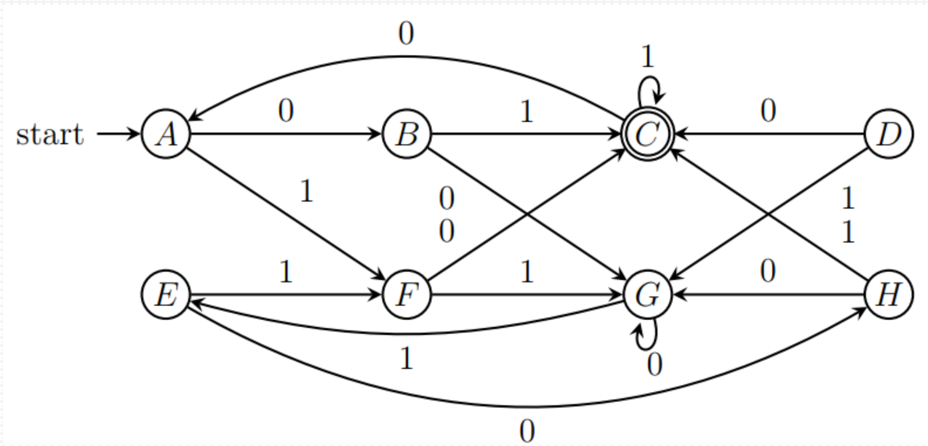
- **等价性只要求**$\hat{\delta}(p, w)$ **和** $\hat{\delta}(q, w)$**同时在或不在**F**中，而不必是相同状态。**

填表算法

- **填表算法递归寻找DFA中全部的可区分状态对:**
  - 基础：如果$p \in F$而$q \notin F$，则$[p, q]$是可区分的；
  - 归纳：$\forall a \in \Sigma$，如果$[r = \delta(p, a), s = \delta(q, a)]$是可区分的，则$[p, q]$是可区分的。

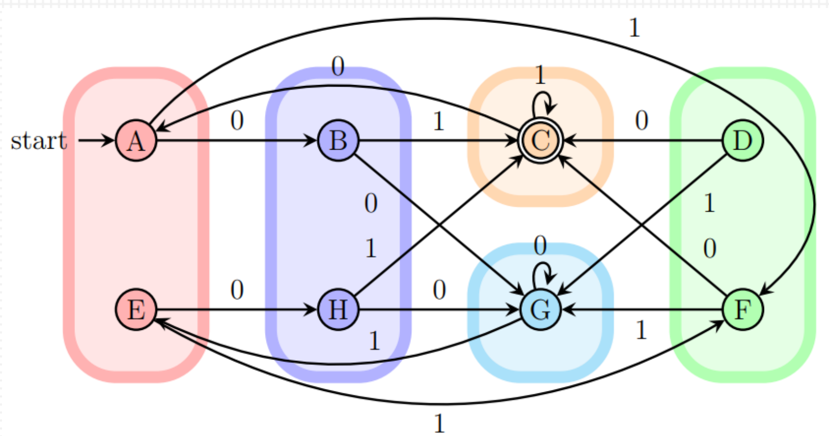
如果填表算法不能区分两个状态，则这两个状态是等价的.

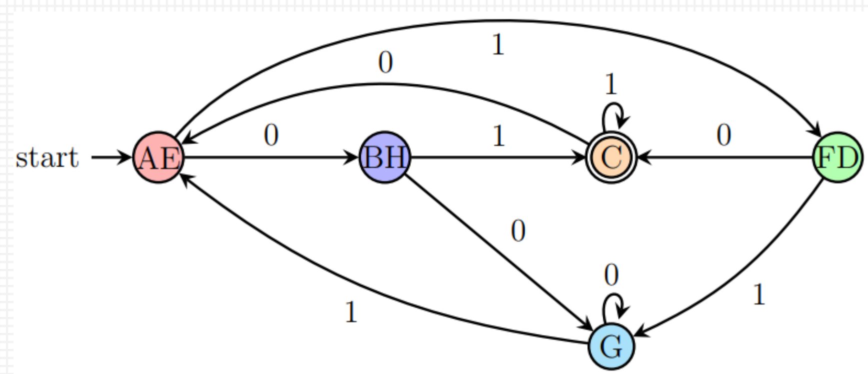
- 例:用填表算法找到如图DFA中全部可区分状态对。



	A	B	C	D	E	F	G
B	×						
C	×	×					
D	×	×	×				
E		×	×	×			
F	×	×	×		×		
G	×	×	×	×	×	×	
H	×		×	×	×	×	×

- 根据填表算法取得的DFA A状态间等价性，将状态集进行划分，得到不同的块；利用块构造新的DFA B，B的开始状态的为包含A初始状态的块，B的接受状态为包含A的接受状态的块，转移函数为块之间的转移；则B是A的最小化DFA。





小结

- **正则语言的泵引理。**
- **正则语言对有关运算的封闭性。**
  - **RL**在并、乘、闭包、补、交、同态映射运算下是有效封闭的。**RL**的同态原像是 **RL**。
- **状态等价性和最小化DFA。**

五、上下文无关文法

(1) 上下文无关文法的定义与构造

(2) 文法的二义性

(3) 上下文无关文法的Chomsky范式

(4) 上下文无关文法的应用

上下文无关法的形式定义

- **定义:上下文无关文法(CFG, Context-Free Grammar，简称文法）G 是一个四元组**

$$G = (V, T, P, S)$$

- **其中,**
  - V:变元的有穷集, 变元也称为非终结符或语法范畴
  - T:终结符的有穷集，且$V \cap T = \emptyset$（即字母表）
  - P:产生式的有穷集，每个产生式包括:
    - 一个变元，称为产生式的头(head)或左部;
    - 一个产生式符号→，读作定义为;
    - 一个$(V \cup T)^*$中的符号串, 称为体(body)或右部（由变元和终结符组成的字符串）
  - S:初始符号$S \in V$，表示文法开始的地方

上下文无关法的形式定义

- 产生式 $A \rightarrow \alpha$，读作 A 定义为α
- 如果有多个 A 的产生式:

$$A \rightarrow \alpha_1, A \rightarrow \alpha_2, \cdots, A \rightarrow \alpha_n$$

- 可简写为:

$$A \rightarrow \alpha_1|\alpha_2|\cdots|\alpha_n|$$

- 文法中变元A的全体产生式，称为A产生式
- 续例：回文语言语法可以表示为

$$G = (\{A\}, \{0,1\}, \{A \rightarrow \varepsilon|0|1|0A0|1A1\}, A)$$

- **从文法变元到字符串的分析过程，称为派生（derivation）。**
- **定义:若上下文**无关文法 $G = (V, T, P, S)$，**设** $\alpha, \beta, \gamma \in (V \cup T)^*$，$A \in V, A \to \gamma \in P$，**那么称在**$G$**中由** $\alpha A \beta$ **可派生出** $\alpha\gamma\beta$，**记为**

$$\alpha A \beta \underset{G}{\Rightarrow} \alpha\gamma\beta.$$

- **若**α**派生出**β**刚好经过了**i**步，可记为**$\alpha \underset{G}{\Rightarrow} \beta$**。**
- **最左派生和最右派生**

文法的语言

- **定义：上下文无关文法$G = (V, T, P, S)$的语言定义为**

$$\mathbf{L}(G) = \{w | w \in T^*, S \underset{G}{\overset{*}{\Rightarrow}} w\}$$

即符号串w在$\mathbf{L}(G)$中要满足：

1. **w仅由终结符组成；**
2. **初始符号S能派生出w.**

上下文无关语言

- 如果语言L是某个 CFG G 定义的语言，即 $L = L(G)$，则称L为上下文无关语言（CFL，Context-Free Language）
- 上下文无关是指在文法派生的每一步

$$\alpha A\beta \Longrightarrow \alpha\gamma\beta$$

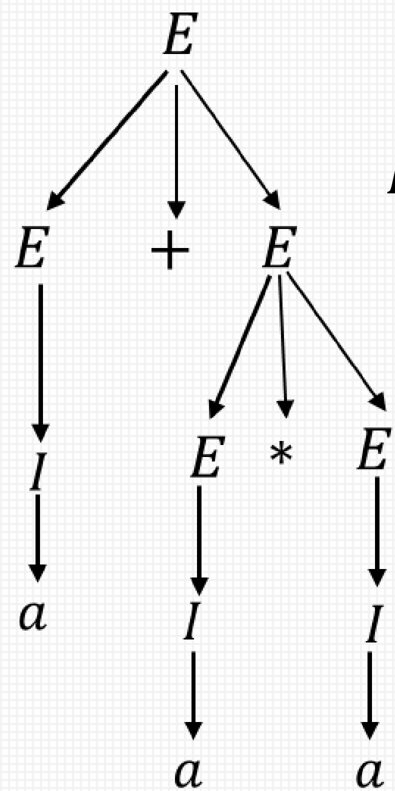
符号串γ仅根据A的产生式派生，而无需依赖 A 的上下文 α 和 β.

文法的歧义性

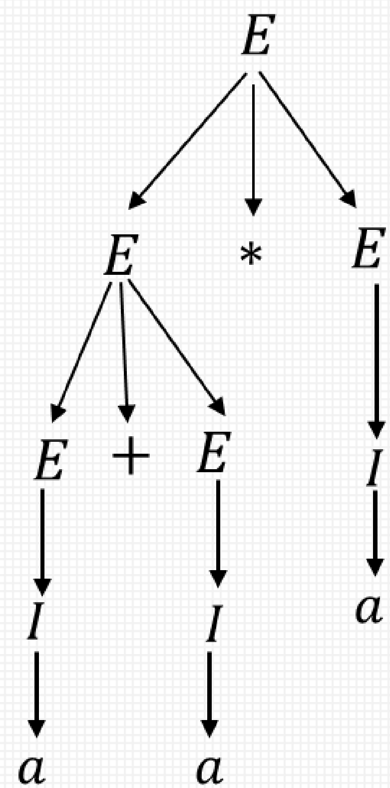
- **定义:如果CFG G使某些符号串有两棵不同的语法分析树，则称该文法是歧义的.**

文法的歧义性

- **例. 算数表达式的文法G_{exp}中，对句型$a+a*a$有下面两棵语法树.**



$$
\begin{aligned}
E &\Rightarrow E+E \\
&\Rightarrow E+E*E \\
&\Rightarrow a+a*a
\end{aligned}
$$



$$
\begin{aligned}
E &\Rightarrow E*E \\
&\Rightarrow E+E*E \\
&\Rightarrow a+a*a
\end{aligned}
$$

- 消除无用符号（useless symbols）：对文法定义语言没有贡献的符号
- 消除$\varepsilon-$产生式（ε−productions）：$A \rightarrow \varepsilon$（得到语言$L-\{\varepsilon\}$）
- 消除单元产生式（unit productions）：$A \rightarrow B$

文法化简的可靠顺序

- **文法简化步骤的顺序是重要的，一个可靠的顺序是**
  - 消除ε −产生式
  - 消除单元产生式
  - 消除非产生的无用符号
  - 消除非可达的无用符号

乔姆斯基范式 CNF

- **定理：每个不带ε的CFL都可以由这样的CFG G定义，G中每个产生式的形式都为**

$$A \rightarrow BC \text{ 或 } A \rightarrow a$$

这里的A, B和C是变元，a是终结符。

格雷巴赫范式 GNF

- **定理:每个不带ε的CFL都可以由这样的CFG G定义,G中每个产生式的形式都为**

$$A \ \rightarrow \ a\alpha$$

其中A是变元, a是终结符, α是零或多个变元的串。

- **GNF 每个产生式都会引入一个终结符。**

小结

- **CFG，派生，最左派生与最右派生，句型，句子**
- **语法分析树，派生与语法树的等价性，二义性文法与文法的固有二义性**
- **CFG的化简**
  - 无用符号、ε –产生式和单元产生式的消除
- **CFG的范式**
  - CNF和GNF

六、下推自动机

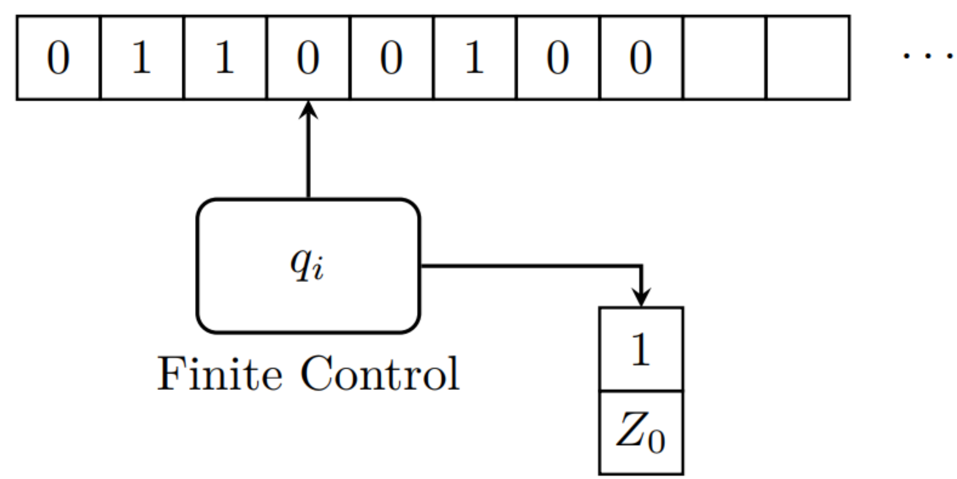
(1) 下推自动机的定义与构造

(2) 两种接收语言方式及等价性

(3) PDA与CFG的等价性

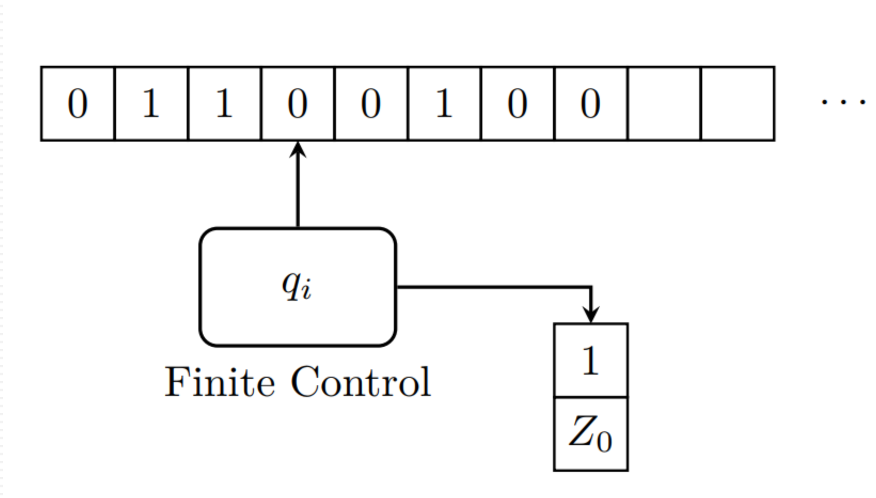
(4) 确定的下推自动机

下推自动机的抽象装置



- **ε−NFA：有限状态，非确定性，空转移**
- **栈:后进先出，只用栈顶，长度无限**
  – 弹栈(Pop): 仅弹出栈顶的一个符号
  – 压栈(Push): 可压入一串符号

下推自动机的抽象装置



- **运转：**
  - 控制器从输入带读入一个符号a，控制栈弹出一个栈顶符号Z;
  - 根据符号a,符号Z,当前所处的状态进行状态的转移并对栈压入0个符号或者压入一个符号串。
    - 0个符号意味着弹出(Pop)操作；
    - 一个符号串意味着压入（Push）操作。

下推自动机的形式定义

- **定义：下推自动机(Pushdown Automata, PDA) P为七元组**

$$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

(1) Q：**有穷状态集**；

(2) Σ：**有穷输入符号集（input alphabet)；**

(3) Γ：**有穷栈符号集(stack alphabet)；**

(4) $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow 2^{Q\times\Gamma^*}$ **:状态转移函数**；

(5) $q_0 \in Q$：**初始状态**；

(6) $Z_0: \Gamma - \Sigma$: **初始栈底符号(start stack symbol)；**

(7) $F \subseteq Q$：**接受状态集。**

状态转移函数和PDA的动作

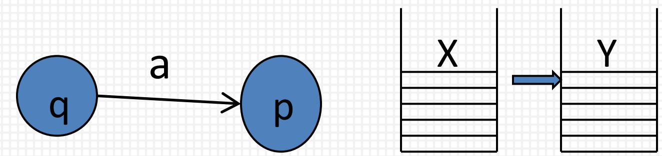
- **三个输入参数：**

(1) 某个状态$q \in Q$**；**

(2) 输入的符号a**（**$a \in \Sigma$**或**$a = \varepsilon$**）；**

(3) 栈顶符号$\mathrm{X} \in \Gamma$**。**

- $\delta(q,a,X) = \{(p,Y),\cdots\}$
  - 1.根据a和X,将当前状态由q转移到p
  - 2.用Y替代栈顶的X
    - 若$Y = \varepsilon$，则弹出X
    - 若$Y = X$，栈顶符号仍为X
    - 若$Y = Z_1Z_2\cdots Z_k$，则弹出X，依次压入Z_k，Z_{k-1}，$\cdots$，Z_1

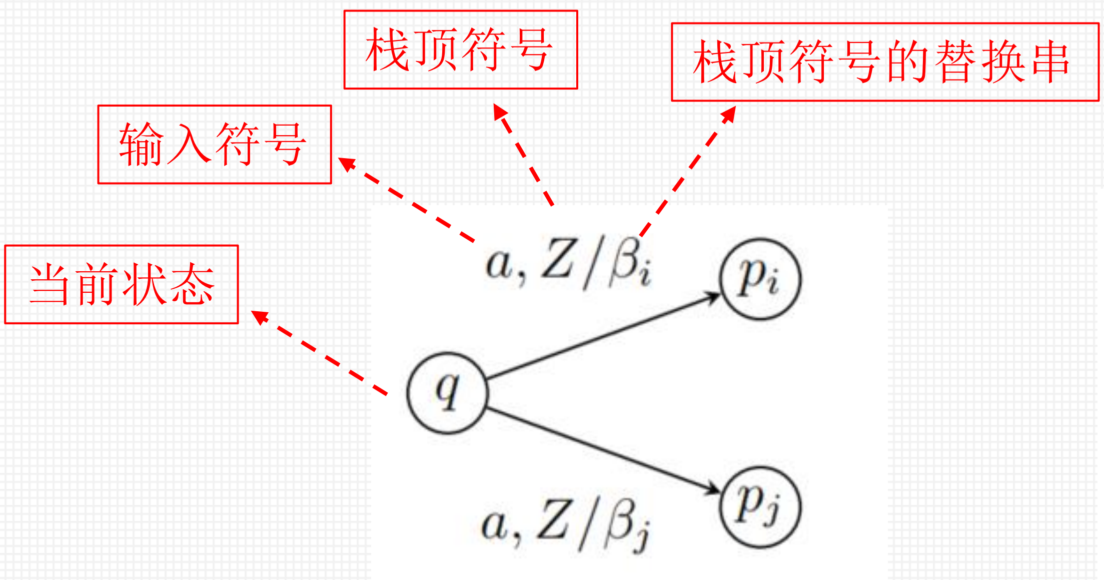


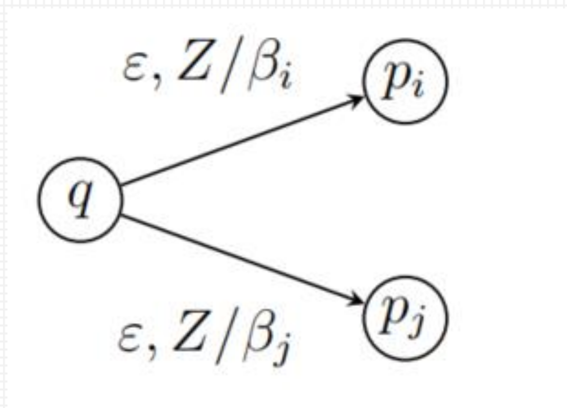
Y = ?	Action
Y=ε	Pop(X)
Y=X	Pop(X) Push(X)
Y=$Z_1Z_2..Z_k$	Pop(X) Push(Z_k) Push(Z_{k-1}) … Push(Z_2) Push(Z_1)

PDA的动作和状态转移图

- 如果$q, p_i \in Q\ (1 \le i \le m)$，$a \in \Sigma, Z \in \Gamma, \beta_i \in \Gamma^*$，$\delta(q, a, Z)$表示如下的动作：

$$\delta(q,a,Z) = \{(p_1,\beta_1),(p_2,\beta_2),\cdots,(p_m,\beta_m)\}, \text{或}$$
$$\delta(q,\varepsilon,Z) = \{(p_1,\beta_1),(p_2,\beta_2),\cdots,(p_m,\beta_m)\}.$$





转移符号

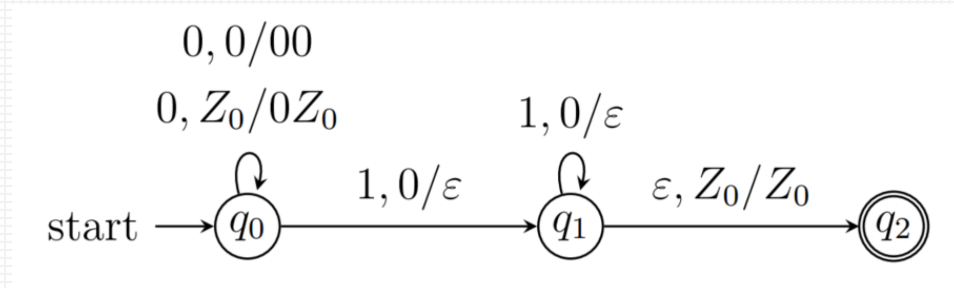
- **定义：在PDA的一个动作下，由ID I到ID J的变化，称为ID的转移。具体的，如果** $(p, \beta) \in \delta(q, a, Z)$，**由** $(q, aw, Z\alpha)$ **到** $(p, w, \beta\alpha)$ **的变化，称为ID的转移$\vdash_P$, 或者$\vdash$（当所指PDA明确时） 记为**

$$(q, aw, Z\alpha) \vdash (p, w, \beta\alpha)$$

该动作表示：当前状态为q, 从当前符号串aw中消耗符号a（可为ε）, 转移到状态p, 同时用β替换栈顶的Z，余下符号串w。

- **例:语言** $L_{01}=\{0^n1^n|n\geq 1\}$**的PDA，识别000111时的ID序列。**

$(q_0, 000111, Z_0) \vdash$
$(q_0, 00111, 0Z_0) \vdash$
$(q_0 0111, 00Z_0) \vdash$
$(q_0, 111, 000Z_0) \vdash$
$(q_1, 11, 00Z_0) \vdash$
$(q_1, 1, 0Z_0) \vdash$
$(q_1, \varepsilon, Z_0) \vdash$
(q_2, ε, Z_0)



因此，$(q_0, 000111, Z_0) \vdash^* (q_2, \varepsilon, Z_0)$。

下推自动机接受的语言

- **PDA** $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$**的语言一般通过终态方式接受来定义，记为L**$(P)$**，定义为**

  $$\mathbf{L}(P) = \{w \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \gamma), p \in F\}$$

  即能够使PDA到达终态的符号串的集合。

下推自动机接受的语言

- **另外一种PDA定义的语言是以空栈方式接受的语言，记为$\mathbf{N}(P)$，定义为**

$$\mathbf{N}(P) = \{w \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon)\}$$

即能够使得PDA的栈变空的符号串集合。

由CFG到PDA的形式化构造

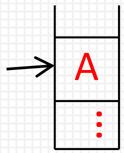
- **给定CFG** $G = (V, T, P, S)$，**构造PDA** 初始栈底符号

$$P_N = (\{q\}, T, V \cup T, \delta, q, S, \emptyset),$$

- **其中**δ**：**

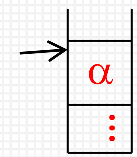
（1）对每个变元A**：（输入空串，压入产生式）**

变化前:



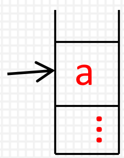
$$\delta(q, \varepsilon, A) = \{(q, \alpha) | A \to \alpha \in P\}$$

变化后:



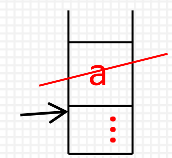
（2）对每个终结符a：(输入字符a，弹出字符a)

变化前:



$$\delta(q, a, a) = \{(q, \varepsilon)\}$$

变化后: a...



弹出

- **构造：设PDA** $P_N = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$，**那么构造CFG** $G = (V, \Sigma, P, S)$。**产生式集合**P**包括：**

（1）对$\forall p \in Q$，**构造产生式**$S \to [q_0 Z_0 p]$;

（2）对 $\forall(p, Y_1 Y_2 \ldots Y_n) \in \delta(q, a, X)$, **构造**$|Q|^n$**个产生式**

$[qXr_n] \to a[pY_1 r_1][r_1 Y_2 r_2] \cdots [r_{n-1} Y_n r_n]$,

其中 $a \in \Sigma \cup \{\varepsilon\}, X, Y_i \in \Gamma$，$r_1, r_2, \cdots, r_n$**是**$Q$**中各种可能组合的**$n$**个状态，即**$r_i$ **可以是**Q **中任意一个状态。**

为了弹出某个栈符号X，第一步是利用某个动作将X替换为其他栈符号串$Y_1 Y_2 \ldots Y_n$，然后再一步一步地弹出这些栈符号串。利用这一个动作便可以定义一组产生式。

(3) 若$Y_1 Y_2 \ldots Y_n = \varepsilon$，**则有产生式**$[qXp] \to a$。

小结

- **PDA定义，PDA两种接受方式及等价性**
- **PDA与CFG的等价性**
  - 从CFG到PDA： CFG →GNF → PDA
  - 从PDA到CFG

七、上下文无关语言的性质

（1）上下文无关语言的泵引理及其证明

（2）泵引理应用

（3）上下文无关语言的的封闭性

（4）上下文无关语言的判定问题

上下文无关语言的泵引理

- **定理：如果语言L是CFL，那么存在正整数N，对$\forall z \in L$，只要$|z| \geq N$，就可以将z分为五部分$z = uvwxy$满足：**

（1） $vx \neq \varepsilon$（或$|vx| > 0$）；

（2） $|vwx| \leq N$；

（3） $\forall i \geq 0, uv^iwx^iy \in L$。

- 例:证明 $L = \{0^n1^n2^n | n \geq 1\}$ 不是上下文无关语言。

(1) 假设L是CFL，那么存在整数N，对$\forall z \in L\ (|z| \geq N)$ 满足泵引理;

(2) 从L中取$z = 0^N1^N2^N$，则显然$z \in L$且$|z| = 3N \geq N$;

(3) 由泵引理，z可被分为$z = uvwxy$, 且有 $|vwx| \leq N$ 和 $vx \neq \varepsilon$;

(4) 那么vwx可能:

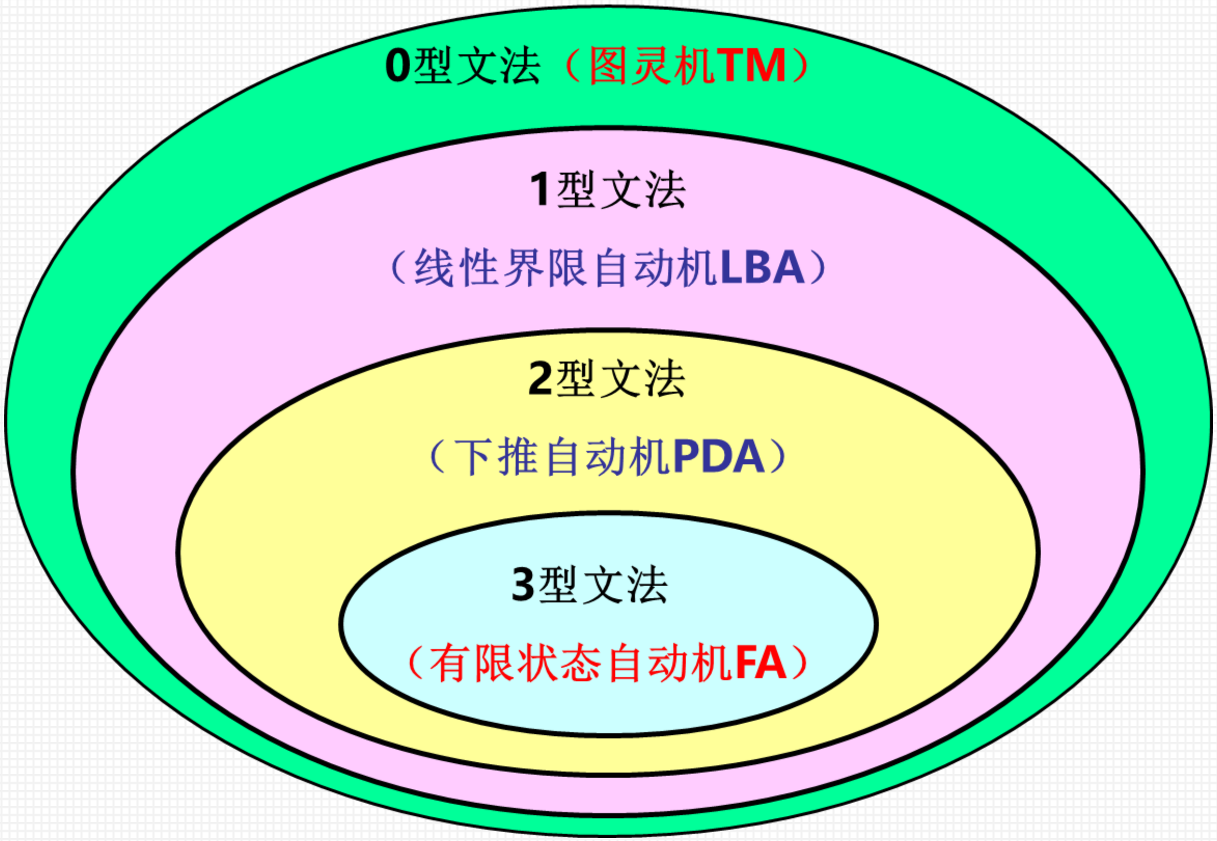
(i) 只包含0或1或2，那么$uwy \notin L$;

(i i) 只包含0和1, 或只包含1和2，那么也有$uwy \notin L$;

(5) 与泵引理$uwy = uv^0wx^0y \in L$矛盾，假设不成立;

(6) L不是上下文无关的。

乔姆斯基文法体系



小结

- **CFL的泵引理及其应用：证明不是CFL**
- **CFL的封闭性**
  - 封闭运算：并、乘、闭包、代换、同态映射、逆同态映射
  - 不封闭运算：交、补
- **CFL的判定性质**
  - 可判定性：空性，成员性
  - 不可判定性
- **乔姆斯基文法体系**

八、图灵机

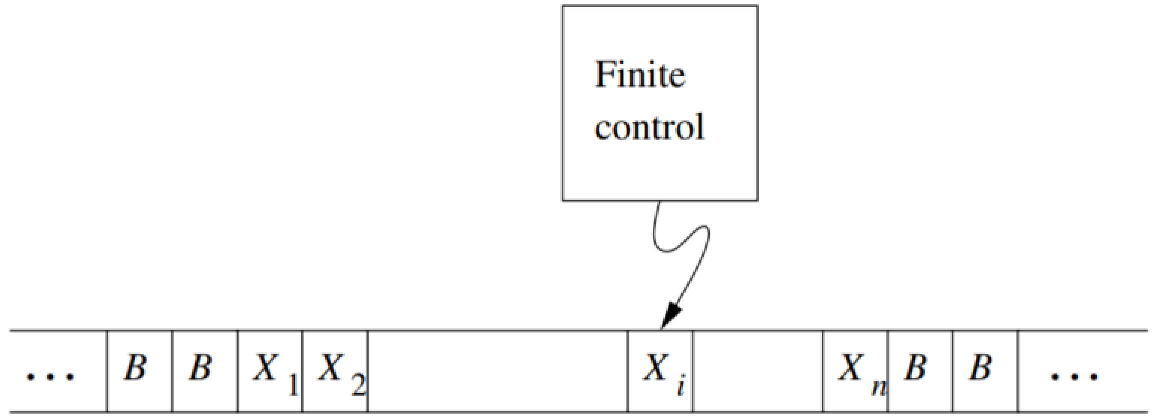
(1) 图灵机的定义和构造

(2) 图灵机作为识别器和转换器

(3) 通用图灵机

(4) 图灵论题

图灵机简介与定义



- **图灵机有一个有穷控制器，一条两端无穷的输入输出带和一个带头(tape head)。**
- **带划分为单元格，每个单元格可以放置一个符号。**
- **带头每次根据当前状态和带头处单元格的符号内容，根据转移规则选择下一个动作。**
  - 每个动作都包括下一个状态，修改带头处单元格的符号以及带头向左或向右移动一个单元格。

图灵机的形式化定义

- 图灵机(TM, Turing Machine) M为七元组

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

(1) Q:有穷状态集

(2) Σ:有穷输入符号集(input alphabet)

(3) Γ:有穷带符号集(tape alphabet)，且总有$\Sigma \subseteq \Gamma$

(4) δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$:转移函数

(5) $q_0 \in Q$：初始状态

(6) $B \in \Gamma - \Sigma$:空格/空白符号(Blank Symbol)

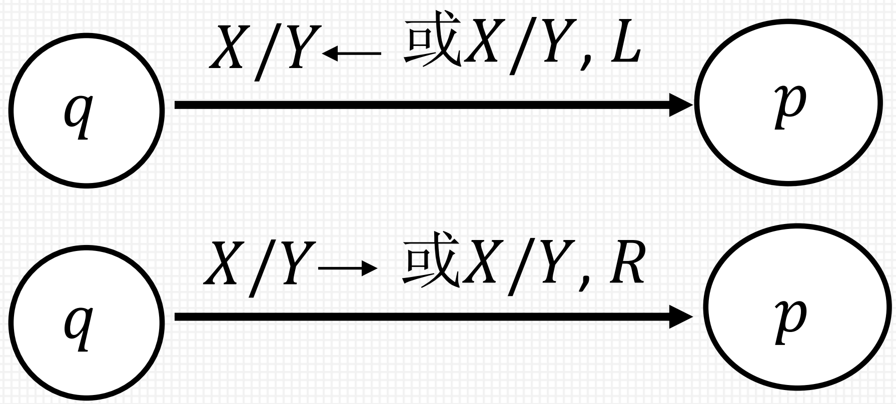
(7) $F \subseteq Q$：接受状态集

图灵机的动作及状态转移图

- **有穷控制器状态为q且带头读入符号X时，若有**

$$\delta(q, X) = (p, Y, L/R)$$

则状态移到p, 带头写符号Y并左或右移一格。图示为



因为每个动作都是确定的，因此是确定的图灵机。

瞬时描述

- **定义:图灵机虽有无穷长的带，但经过有限步,带上非空内容总是有限的. 因此用全部非空符号、当前状态及带头位置，定义瞬时描述(ID):**

$$X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n$$

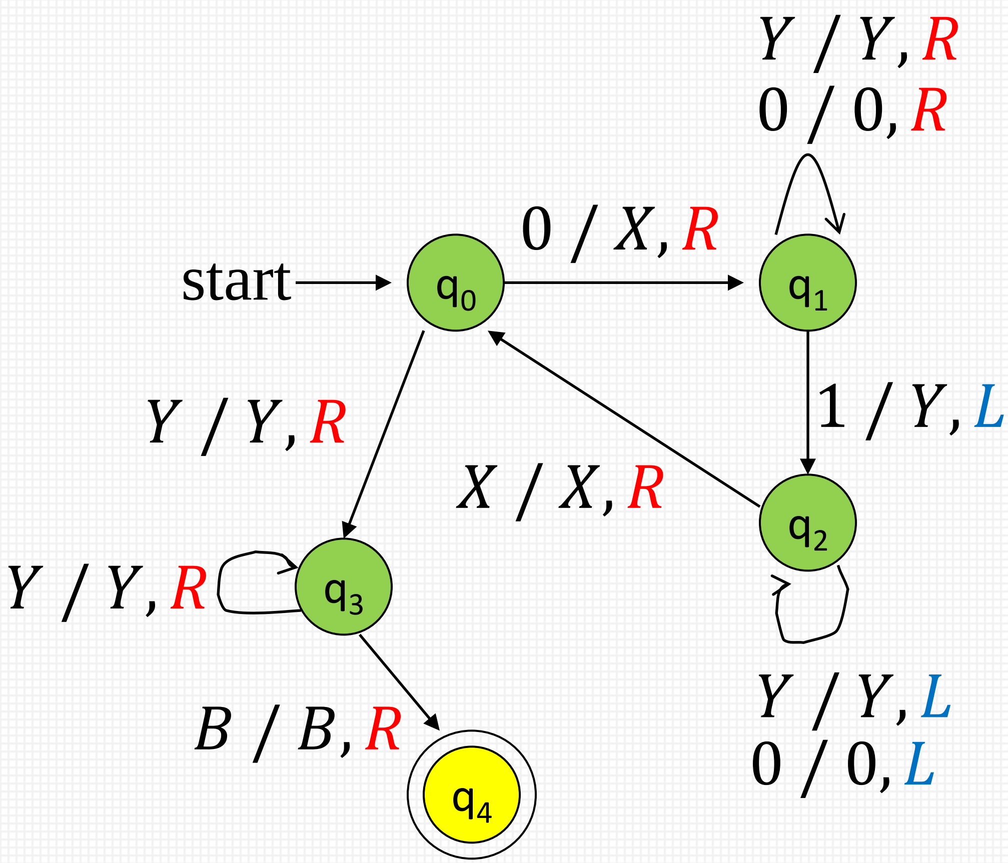
(1) q是图灵机的当前状态;

(2) 带头在左起第i个非空格符X_i上;

(3) $X_1X_2\cdots X_n$是从最左到最右非空格内容（$i=0$表示带头左端有空格符号，$i=n$表示带头右端有空格符号）。

- **ID转移符号$\vdash_M$。若某ID是从另一个经有限步转移而得到的，记为$\vdash_M^*$。若M已知，简记为$\vdash$和$\vdash^*$。**

识别$\{0^n1^n|n \geq 1\}$的图灵机



1. 处于q_0, 若待读入的为0，则将0标记为X, 向右移动

2. 一直向右移动(跳过0和Y)，直到遇到第一个1，标记为Y，改为向左移动

3. 一直向左移动(跳过Y和0)，直到遇到X, 改为向右移动

4. 如果下一个要读入的符号是0，则跳转到第一步；否则，向右移动检查是否还有剩余的1。如果后面已经全是Y，那么在空格符停下，进入接受状态。

图灵机的语言

- **定义:如果**$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$**是一个图灵机,则**$M$**接受的语言为**

$$\mathbf{L}(M) = \{w \mid w \in \Sigma^*, q_0 w \vdash^* \alpha p \beta, p \in F, \alpha, \beta \in \Gamma^*\}$$

  - 输入串放在输入带上,M处于q_0,带头位于输入串的第一个字符上,输入串最终会导致M进入某个终止(接受)状态。
- **一般假定当输入串被接受时**M**总会停机(halt),即没有下一个动作的定义。**
- **而对于不接受的输入,TM可能永远不停止。我们无法知道,到底是因为运行的时间不够长而没有接受呢,还是根本就不会停机。**

整数函数计算器

- **整数计算器：图灵机可以作为语言的识别器，也可以用作整数函数计算器。**
- **例如，将整数**$i \geq 0$**表示为字符串**0^i
- **若计算**k**个自变量**i_1**，**$i_2, \cdots, i_k$**的函数**f**，用**
$$0^{i_1}10^{i_2}\cdots 10^{i_k}$$

作为图灵机M**的输入**

- **当**M停机**且输入带上为**0^m**，表示计算**
$$f(i_1, i_2, \cdots, i_k) = m$$

扩展及受限图灵机

- **扩展图灵机**
  - 状态中存储
  - 多道(multi-track)技术
  - 多带(multi-tape)技术
  - 非确定图灵机
- **受限图灵机**
  - 半无穷带图灵机
  - 多栈机器

小结

- 图灵机简介、形式化定义
- 图灵机的表示能力：接受语言、整数运算
- 扩展及受限图灵机