Chapter 1

拜伦：第一个给计算机写程序的人；图灵：人工智能之父、计算机科学之父、**计算机之父**；拜伦：现代计算机之父、博弈论之父；巴贝奇：差分机、分析机；丘奇：可计算理论

关系：$R \subset A \times B$；等价关系：自反、传递、对称

半群：(S,*)，其中*是满足**结合律**的二元运算. 若*还满足交换律，则为**交换半群**

同构：(S,*)和(T,○)是半群，若$f:S \to T$是双射且$f(a*b)=f(a)\circ f(b)$，$\forall a,b \in S$

- 同构四步骤：选f使得Dom(f)=S，f单，f满，$f(a*b)=f(a)\circ f(b)$

通路（边顺次相连）、路径（无重复边）、简单路径（无重复点）、树高（从0层开始）

字符串前缀后缀包括空串

$$w^n = \underbrace{ww\cdots w}_{n}$$

例: $(abba)^2 = abbaabba$

规定: $w^0 = \varepsilon$

$|\{\epsilon\}|=1$，$|\epsilon|=0$

语言L – “+” 闭包：

$$L^+ = L^1 \cup L^2 \cup \cdots$$

$$= L^* - \{\varepsilon\}$$

例：

$$\{a,bb\}^+ = \begin{Bmatrix} a,bb, \\ aa,abb,bba,bbbb, \\ aaa,aabb,abba,abbbb,\ldots \end{Bmatrix}$$

Chapter 2

确定有限自动机的定义

确定有限自动机 **DFA (deterministic finite automata)** 是五元组 $A = (Q, \Sigma, \delta, q_0, F)$.

- 有限状态集
- 输入符号集
- 转移函数
- 开始状态
- 终态集合

$q_0 \in Q,\ F \subseteq Q$

$\delta:\ Q \times \Sigma \to Q$

扩展转移函数 δ^*

$$\delta^*: Q \times \Sigma^* \to Q$$

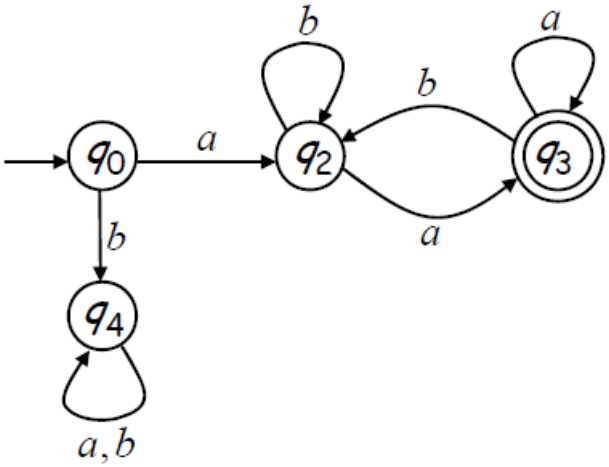
递归定义

基础： $\delta^*(q, \varepsilon) = q$

递归： $\delta^*(q, w\sigma) = \delta(\delta^*(q, w), \sigma)$

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

$$L(M) = \{\, awa \mid w \in \{a, b\}^* \}$$



Chapter 3

w被NFA接受：存在一条路径接受w

w被NFA拒绝：NFA中不存在接受w的路径（对于每个路径，要么字符串输完不在终态，要么无法输完）

NFA 的定义

非确定有限自动机是五元组 $A = (Q, \Sigma \cup \{\varepsilon\}, \delta, q_0, F)$

- 有限状态集
- 输入符号集
- 转移函数
- 开始状态
- 终态集合

$q_0 \in Q$
$F \subseteq Q$

与 DFA 有三处不同 δ: $Q \times \Sigma \cup \{\varepsilon\} \to 2^Q$

注意与DFA不同点：ϵ也可以作为输入、$\delta : Q \times \Sigma \cup \{\epsilon\} \to 2^Q$映射到状态的**幂集**

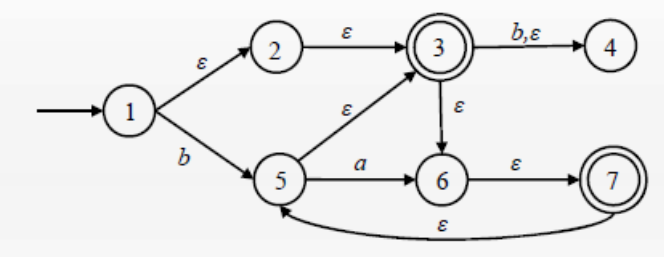
	0	1
$\to p$	$\{q\}$	$\varnothing$
q	$\{q\}$	$\{q, r\}$
$*r$	$\varnothing$	$\varnothing$

状态转移是集合↑

ε - 闭包

- 状态 q 的ε−闭包是 q(包括q自身) 的ε 路径到达的所有状态，记为$EC(q)$。
- 归纳定义：设 $NFA\ A = (Q, \Sigma, \delta, q_0, F)$，$q \in Q$，$EC(q)$ 满足如下条件：
  （1）$q \in EC(q)$
  （2）若 $p \in EC(q)$ 且 $r \in \delta(p, \varepsilon)$, 则 $r \in EC(q)$
- 集合S的ε-闭包$EC(S)$定义为：$EC(S) = \bigcup_{q \in S} EC(q)$

扩展转移函数（老麻烦了）：**与DFA相比，函数映射到一个集合**



$\delta^*(1, b) = \{3, 4, 5, 6, 7\}$

$\delta^*(1, bb) = \{4\}$

$\delta^*(1, ba) = \{3, 4, 5, 6, 7\}$

NFA M

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

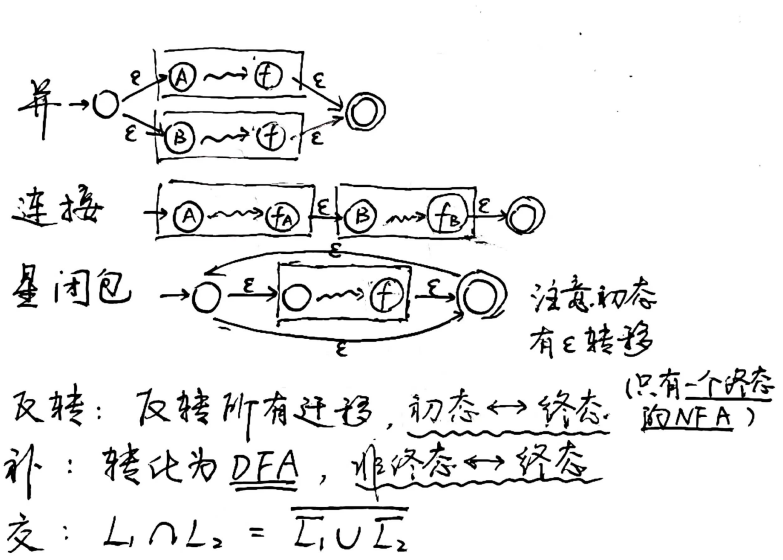
接受的语言是：

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

Chapter 4

单一终态的NFA：**任何NFA都等价于一个只有一个终态的NFA**（将所有终态用ϵ转移到一个状态）

正则语言的封闭性：



补要用DFA，原因是NFA一些因为没有合适转移的串在补的NFA里也没有合适转移；因此要用有明确转移的DFA

正则表示的定义：**三个基础、四个归纳形式**（基础不要忘了∅）

定义（归纳）：

基础：$\varnothing,\varepsilon,a$

给定两个正则表示 r_1, r_2

r_1+r_2

$r_1\cdot r_2$

$r_1{}^*$

(r_1)

都是正则表示

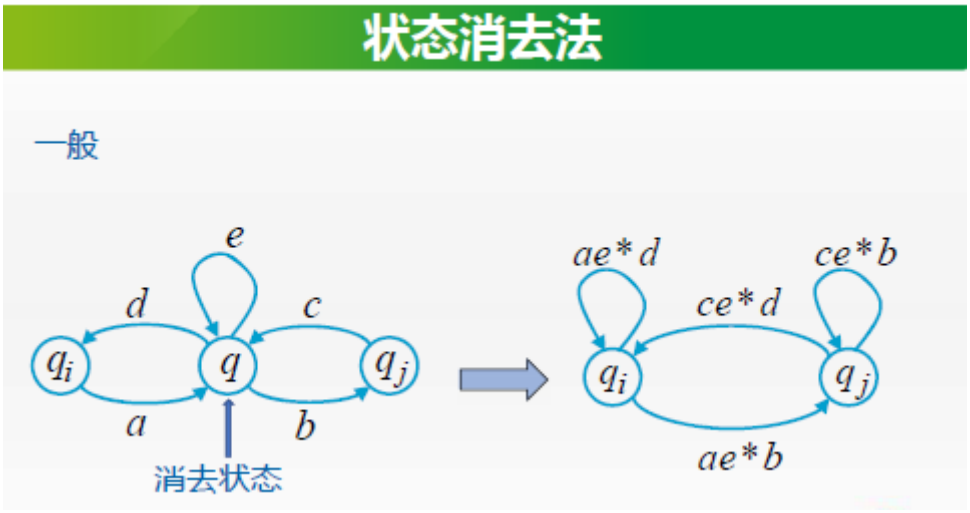
运算符优先级：* 、· 、+

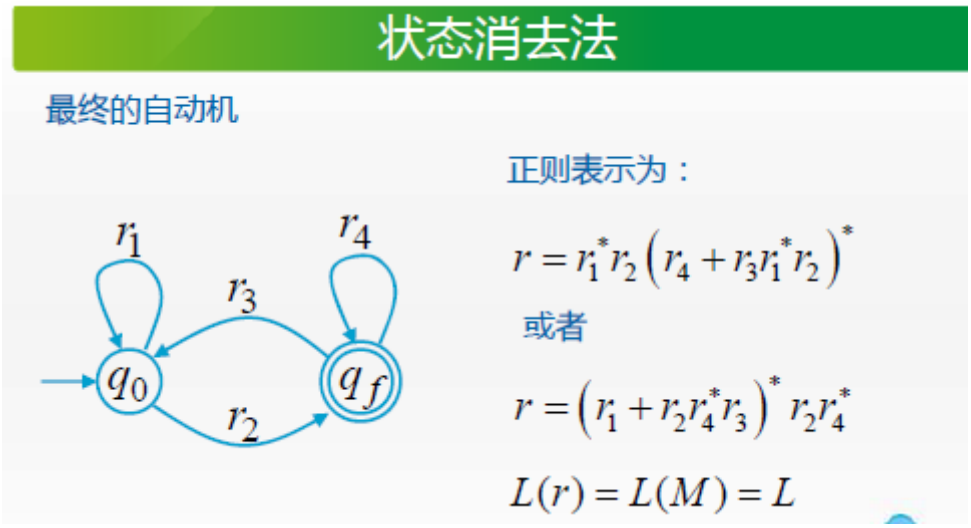
正则表示的语言定义：对语言从上面三个基础、四个归纳形式来定义

没有两个连续0的字符串：(1+01)*(0+ϵ)

01交替的字符串：(0+ϵ)(10)*(1+ϵ)

证明正则表示语言=正则语言：前推后从正则表示定义和正则语言封闭性，反推用状态消去构造





$r = r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*$

正则语言的同态：$h : \Sigma \to T^*$拓展为$h(w) = h(w_1) \cdots h(w_n)$（$w = w_1 \cdots w_n$）

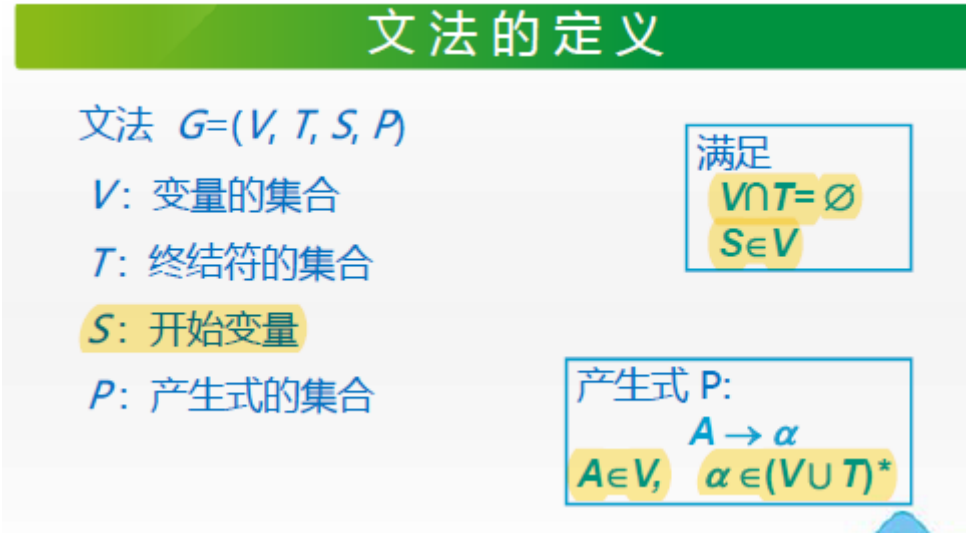
逆同态（映射到字符串集合）:$h^{-1}(L) = \{w | w \in \Sigma^* \wedge h(w) \in L\}$

定理1：若L是正则语言，则h(L)也是正则语言

定理2：若L是正则语言，则h^{-1}(L)也是正则语言

正则表示：连接运算的单位元是ϵ，**零元是**∅；并运算单位元是∅

Chapter 5



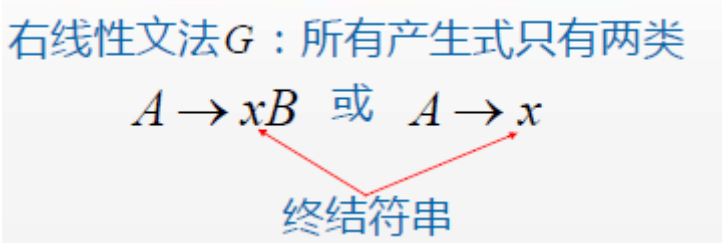
句子/句型

推导闭包$\overset{*}{\Rightarrow}$

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

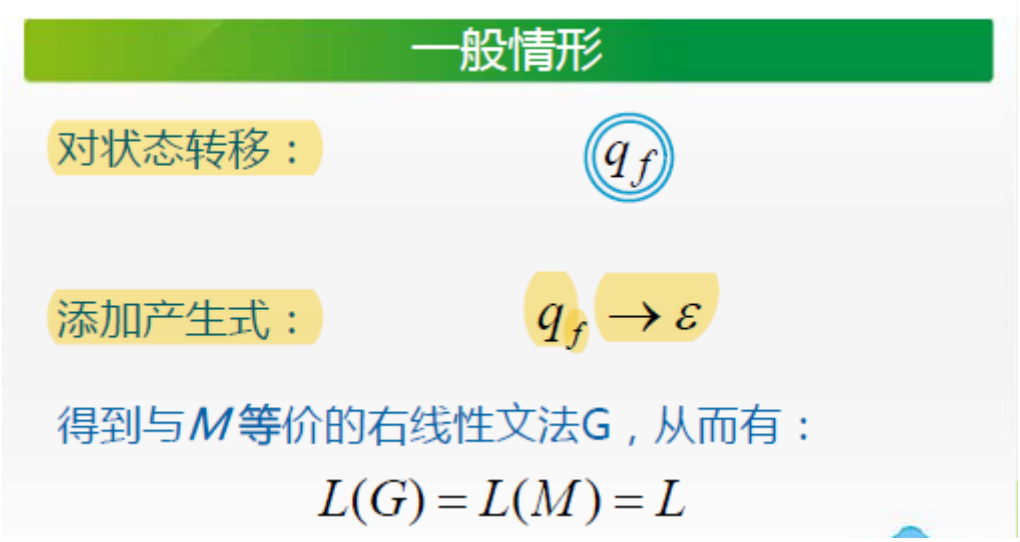
线性文法：产生式右侧最多一个变量，要么没变量，剩下的均是终结符串

右线性文法：产生式右侧要么没变量，有变量则至多一个且在最右侧

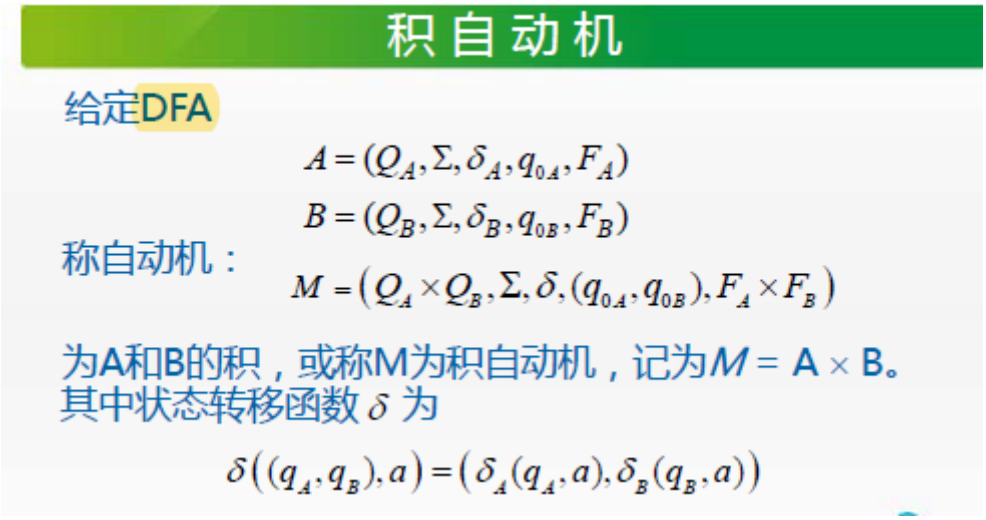


正则文法（3型文法）：左线性文法或右线性文法（二者只能出现一个）

NFA转化为右线性文法易错点：



积自动机，注意是DFA，语言是两个语言的并



注意F、S、δ

Chapter 6

判定算法：

- w是否能被接受：DFA O(n) / NFA O(ns^2) / 正则表达式转化为NFA
- 正则语言是否为空：DFA O(S) / 正则表达式 O(n)(符号数)
- 判断两个正则语言是否相等：转化为DFA，两个DFA并起来，用填表法看两个初态是否可区分O(n^2)
- 正则语言是否无限：DFA初态到终态是否存在环

泵引理:
$(\forall m)(\exists w)(w\in L\wedge|w|\geq m\wedge(\forall x,y,z)(\exists k)(xyz=w\wedge|xy|\leq m\wedge y\neq\epsilon\rightarrow xy^kz\notin L))$

注意k的选取可以与m,w,x,y,z有关，一般常用m、|y|

泵引理不是充分条件，而是必要条件

Chapter 7

DFA状态集等价关系：$p,q\in Q,pRq\quad iff\quad \delta^*(p,w)\in F\Leftrightarrow\delta^*(q,w),\forall w$

填表算法：基础是p终态、q非终态标记可区别；归纳定义状态r、s通过某个符号a转移到可区别的两个状态，则r、s可区别

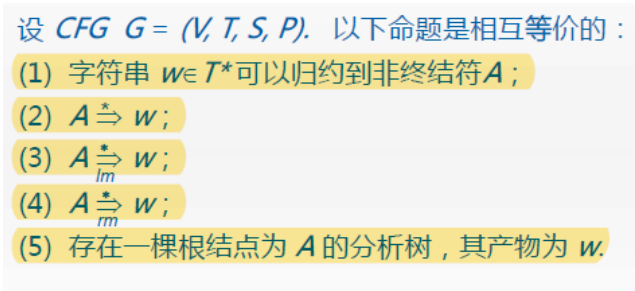
最优的DFA：用填表法计算出等价类后进行划分；可以反证状态数比M少的自动机来证明最优

归约：$v+d\Rightarrow E+d\Rightarrow EOd\Rightarrow EOE\Rightarrow E$，推导$E\Rightarrow v+d$

最左推导，最右推导：每次都替换最左/右侧的非终结符

左句型：$S\underset{lm}{\overset{*}{\Longrightarrow}}\alpha$

语法分析树：内部节点非终结符、**叶节点为终结符或非终结符**，如果叶节点ϵ则为父节点唯一子节点



证明某语言是某文法的语言：要正反两方面说包含关系

Chapter 8

文法二义性：**存在某字符串**有至少两种最左（或最右）推导⇔存在两棵不同的分析树

一个CFG是否为二义的问题是不可判定的

二义性消除

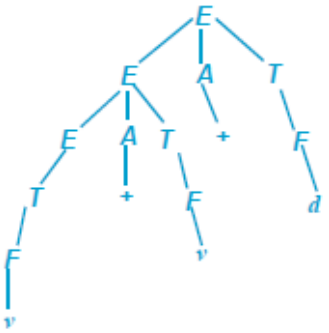
采用左结合方法将右上图的文法变换为左下图，串:

$v + v + d$

存在唯一的分析树（右下图）

$E \to EAE \mid T$
$T \to TMT \mid (E) \mid v \mid d$
$A \to +$
$M \to *$

$E \to EAT \mid T$
$T \to TMF \mid F$
$F \to (E) \mid v \mid d$
$A \to +$
$M \to *$



二义性消除

悬挂else二义性

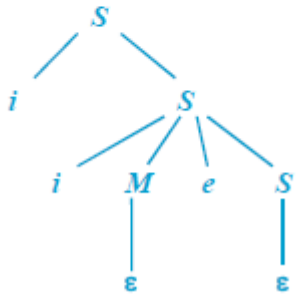
$S \to \varepsilon \mid iS \mid iSeS$

采用最近嵌套方法消除悬挂 else 二义性

将文法变换为下面的文法

$S \to \varepsilon \mid iS \mid iMeS$
$M \to \varepsilon \mid iMeM$

串 iie 存在唯一的分析树（右图）



固有二义：CFL的所有文法均是二义的（不存在无二义文法）

题目：$n_0 = n_1$

CFG构造实例

方法3:

另一种解决例3的方法. CFG 如下所示：

$$S \to 0B \mid 1A \mid \varepsilon,\ B \to 1S \mid 0BB,\ A \to 0S \mid 1AA$$

对于任意字符串 w, 有三种情况：

(a) w 中 0 的数量和 1 的数量相等，通过 S 生成；

(b) w 中 0 的数量比 1 的数量多一个，通过 A 生成；

(c) w 中 0 的数量比 1 的数量少一个，通过 B 生成。

CFG构造实例

方法4:

文法 G ：

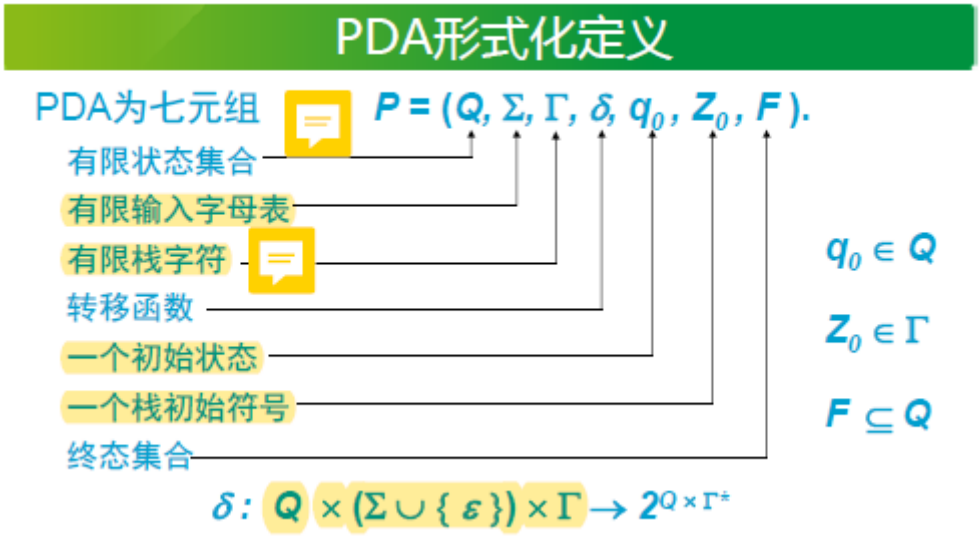
$$S \to SS \mid 0S1 \mid 1S0 \mid \varepsilon$$

Chapter 9

栈可以为空（可以把栈底符号弹出来），但空栈以后就不可以转移了

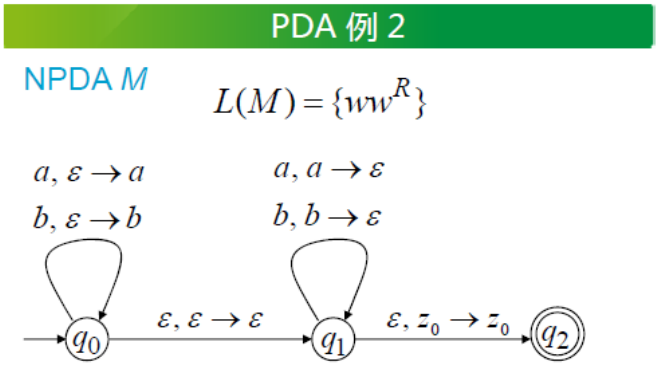
转移时可以有$\epsilon, \epsilon \to \epsilon$

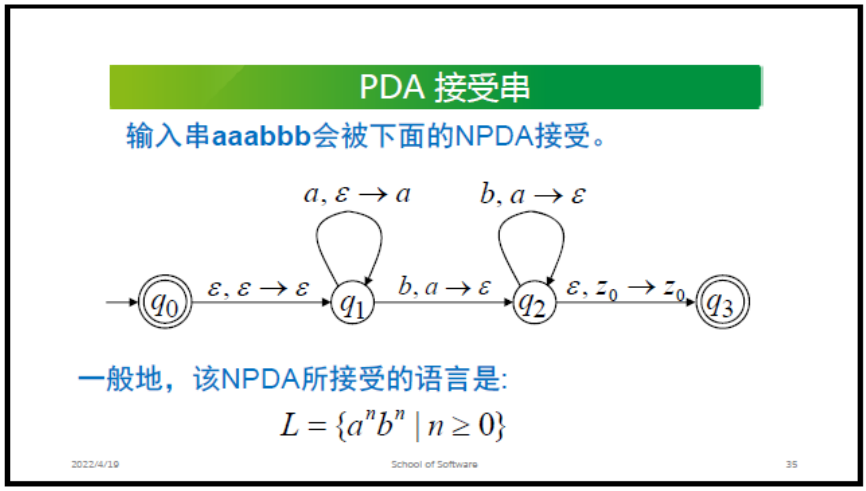
PDA是非确定的，转移的是一个集合



Σ和Γ不一定无交

终态型PDA接受字符串：所有字符串输入完毕且落在一个终态上

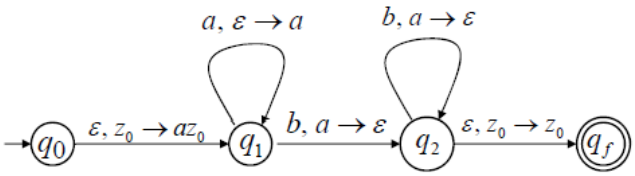




PDA 例 4

设计接受如下语言的PDA：

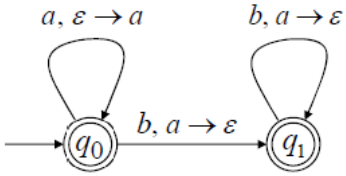
$$L(M)=\{a^{m-1}b^m \mid m\geq 1\}$$



PDA 例 3

设计接受如下语言的PDA：

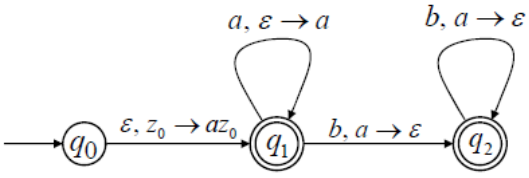
$$L(M)=\{a^n b^m \mid n\geq m\}$$



PDA 例 5

设计接受如下语言的PDA：

$$L(M)=\{a^n b^m \mid n\geq m-1\}$$
$$=\{a^{m-1}b^m \mid m\geq 1\}\cup\{a^{m+k}b^m \mid m,k\geq 0\}$$



即时描述(q,w,α)，其中w是还未输完的串，α是栈底符号串；传递和传递闭包$\vdash$ 、 $\vdash^*$

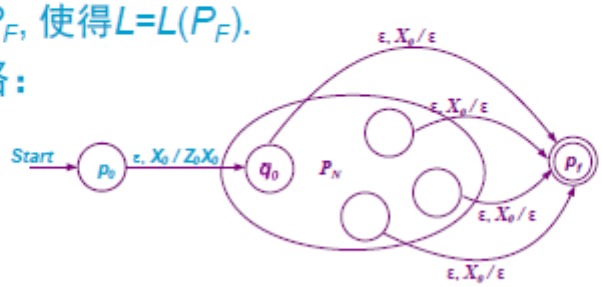
终态型PDA语言：$\{w|(q_0,w,Z_0)\vdash^*(q_f,\epsilon,\alpha),q_f\in F,\alpha\in\Gamma^*\}$

空栈型PDA语言：$\{w|(q_0,w,Z_0)\vdash^*(q,\epsilon,\epsilon),q\in Q\}$

两种 PDA 语言的关系

- 空栈型PDA到终态型PDA
  对于 PDA $P_N=(Q,\Sigma,\Gamma,\delta_N,q_0,Z_0)$, $L=L(P_N)$,存在一个 PDA P_F, 使得$L=L(P_F)$.

证明思路：

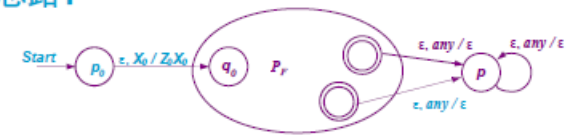


$$P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$$

两种 PDA 语言的关系

- 终态型PDA到空栈型PDA
  对于PDA $P_F=(Q,\Sigma,\Gamma,\delta_F,q_0,Z_0,F)$, $L=L(P_F)$, 存在一个 PDA P_N, 使得 $L=N(P_N)$.

证明思路：



$$P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$$

2022/4/19 School of Software 87

总结，都要添加一个状态并且**改栈底符号为**X_0，添加转移$\epsilon, X_0/Z_0X_0$

CFG 转换 PDA

例:

$E \to EOE \mid (E) \mid v \mid d$
$O \to + \mid *$

其中δ 定义为:

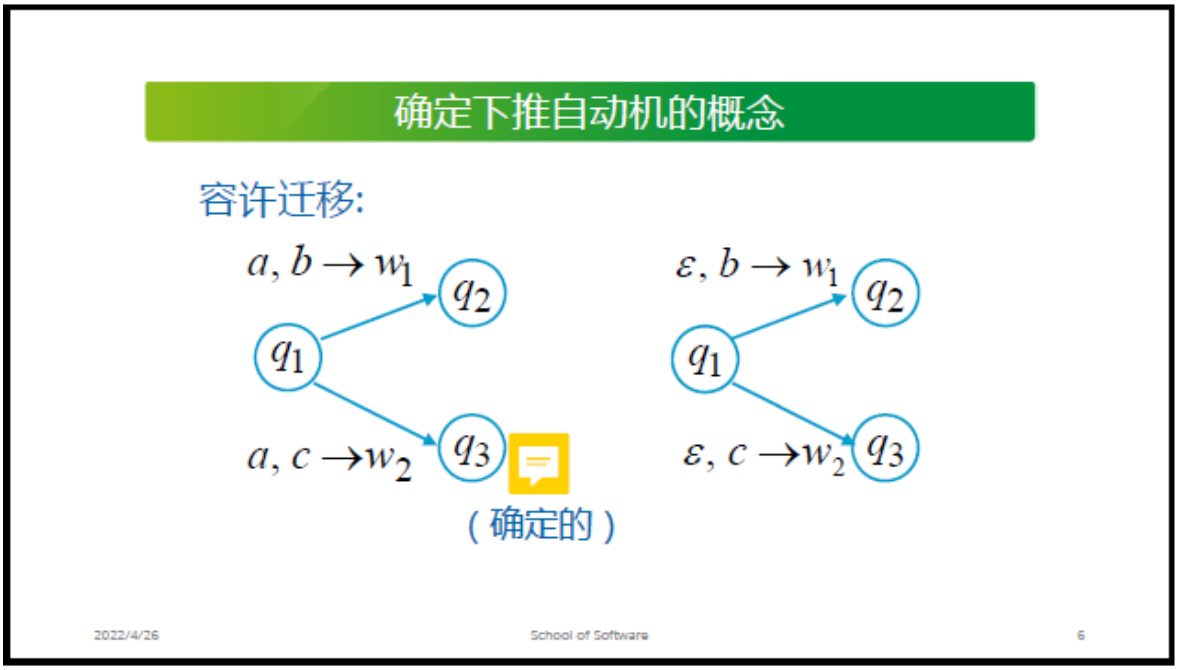
$$\delta(q, \varepsilon, E) = \{(q, EOE), (q, (E)), (q, v), (q, d)\},$$
$$\delta(q, \varepsilon, O) = \{(q, +), (q, *)\},$$
$$\delta(q, v, v) = \{(q, \varepsilon)\}, \quad \delta(q, d, d) = \{(q, \varepsilon)\}$$
$$\delta(q, +, +) = \{(q, \varepsilon)\}, \delta(q, *, *) = \{(q, \varepsilon)\}$$
$$\delta(q, (, () = \{(q, \varepsilon)\}, \quad \delta(q,),)) = \{(q, \varepsilon)\}$$

添加$\delta(q, \epsilon, S) = \{(q, \Sigma^*)\}$，注意添加$\delta(q, v, v) = \{(q, \epsilon)\}$

Chapter 10

确定下推自动机：与PDA只有转移函数上的不同

- **约束**：**栈顶符号不能为空、输入符号为空和不为空互斥**（只能有一个）、转移的状态唯一



注意输入符号为空的互斥条件仅限于相同栈顶元素的条件下：

- $\delta(q, \epsilon, X)$与$\delta(q, a, X)$互斥
- $\delta(q, \epsilon, X)$与$\delta(q, a, Y)$不互斥

确定性上下文无关语言：终态型DPDA语言

$\{0^n 1^n | n \geq 1\}$既是终态型DPDA语言，也是空栈型DPDA语言

$\{ww^R\}$不是DPDA语言，证明了DPDA<PDA

$\{wcw^R\}$是DPDA语言，但不是正则语言，证明了正则<DPDA（正则显然是DPDA语言）

前缀性质：语言中没有两个元素使得其中一个是另一个前缀

空栈型DPDA定理（充要）：

- 一个语言是空栈型DPDA语言，当且仅当　1）有前缀性质　2）是某终态型DPDA语言
- 空栈型DPDA与正则语言无关，因为$\{wcw^R\}$是KDPDA但不正则，$\{0^*\}$正则但不是KDPDA

判断KDPDA：前缀性质

判断DFA/CFG：泵引理

定理：

- L是空栈型DPDA语言，则L有一个无二义性文法G使L(G)=L：由空栈型PDA构造CFG过程看出
- L是终态型DPDA语言，则L有一个无二义性文法G使L(G)=L：在L每个元素后面添加$变成空栈型DPDA语言，CFG中添加$S \to \epsilon$

另一方面，$\{ww^R\}$为非固有二义的，因此包含是真包含

综上：

KDPDA<DPDA<非固有二义CFG<CFG=PDA

DFA<DPDA<非固有二义CFG<PDA=CFG

CFG的化简

消除ϵ产生式

可空符号：$A \overset{*}{\Rightarrow} \epsilon$，A是变量；归纳，若产生式B的右边均是可控符号，则B也是

消除可空符号：对于每个产生式$A \to A_1 \cdots A_n$，若m<n个为可空符号则扩写成2^m项，m=n则2^{m-1}项

定理：消除后的语言是$L(G) - \{\epsilon\}$

消除单一产生式

单一产生式：$A \to B$，其中AB均为非终结符

单一偶对(A,B)：$A \overset{*}{\Rightarrow} B$路径上全部都为单一产生式

设 CFG $G = (V, T, S, P)$, 通过下列步骤消去 G 中的单一产生式：

(1) 计算 G 的单一偶对集合；
(2) 对每个单一偶对 (A, B), 在 G_1 中加入产生式 $A \to \alpha$, 其中 $B \to \alpha$ 为一非单一产生式；
(3) G_1 中包含 G 的所有非单一产生式。

消除无用产生式

有用符号：X在一条$S \overset{*}{\Rightarrow} w$的路径上；**有用符号既包括变量，也包括终结符**

无用产生式：含有无用符号的产生式

产生符号：$X \overset{*}{\Rightarrow} w$；可达符号：$S \overset{*}{\Rightarrow} \alpha X \beta$

有用符号⇒产生符号、可达符号（反之不成立）

先消去非产生、再消去非可达

- 画面：　S ———>α———>√
  β———>×

化为Chomsky范式

Chomsky范式（**CNF**）：1）G中不含无用符号；2）产生式P只有两种形式：$A \to BC \ / \ A \to a$

注意：消除无用符号可以在其中穿插反复进行

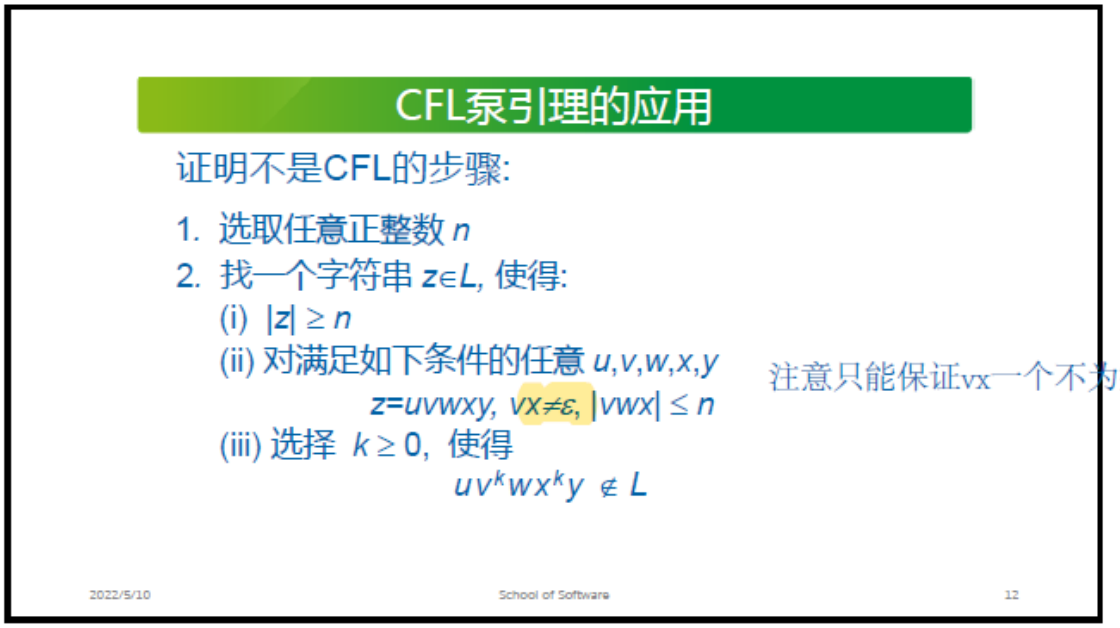
定理：设 CFG G 的语言包含非 ε 的字符串，通过上述步骤从 G 构造 G_1，则 G_1 符合 CNF 的要求，且满足：

$$L(G_1) = L(G) - \{\varepsilon\}$$

Chapter 11

CFL泵引理：

$\forall m \exists z(z \in L \wedge |z| \geq m \wedge (\forall u,v,w,x,y)(\exists k)(z = uvwxy \wedge vx \neq \epsilon \wedge |vwx| \leq m \rightarrow uv^k wx^k y \notin L))$



例子：$\{ww\}$、$\{0^n1^n2^n\}$不是CFL

CFL封闭运算：并、星闭包、连接、反转

非封闭运算：交、补、差

替换：$s:\Sigma \rightarrow \mathcal{L}$

定理：设$\forall a \in \Sigma, s(a)$是$CFL$，若L是Σ上的CFL，则s(L)是CFL

定理：设L是CFL，则h(L)（$h^{-1}(L)$）均是CFL

正则闭性质：CFL和正则语言的交为CFL（在此基础上**交运算**也封闭）

例题：$L = \{a^n b^n | n \neq 100\}$是CFL，取正则语言L={$a^{100}b^{100}$}，它的补为L={$w | w \neq a^{100}b^{100}$}，与CFL$\{a^n b^n\}$交得证

例题：$L = \{n_a n_b n_c\}$不是CFL，假设成立，取正则语言$\{a^*b^*c^*\}$归谬，二者交为$\{a^n b^n c^n\}$是CFL，显然不正确

Chapter 12

空语言问题：检查语言是否为空，则判断开始变量是否无用

无限语言问题：进行CFG的化简，之后构造变量依赖图，判断是否有环

语言元素问题：**CYK解析算法 O(n^3)**

i	1	2	3	4	5
a_i	b	a	a	b	a

j=5	{S,A,C}				
j=4	-	{S,A,C}			
j=3	-	{B}	{B}		
j=2	{S,A}	{B}	{S,C}	{S,A}	
j=1	{B}	{A,C}	{A,C}	{B}	{A,C}
	b	a	a	b	a

$S \to AB|BC$
$A \to BA|a$
$B \to CC|b$
$C \to AB|a$

字符串 bbabaa 是否属于该文法的语言？

CFL不可判定问题：

CFG是否歧义、CFL是否固有二义、CFL相交是否为空（交自动机）、CFL是否相等（并自动机看初态是否可区分）、CFL是否是Σ^*

上下文无关语言判定性质

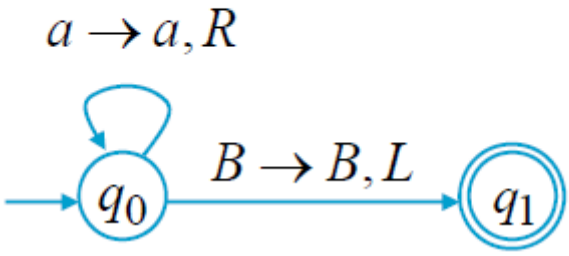
CFL的一些不可判定问题：

1. 上下文无关文法是否无歧义的?
2. 上下文无关语言是否固有歧义的?
3. 两个上下文无关语言相交是否为空?
4. 两个上下文无关语言是否相等?
5. 上下文无关语言是否等于 Σ^*?

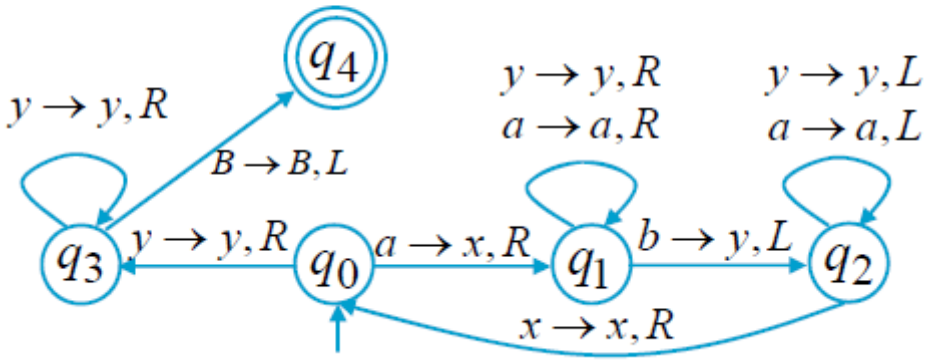
TM的注意事项：

- 初始时读头位于输入串最左
- 输入串不为空
- TM机默认是确定TM的
- 状态转移是一个元素（三元组）而非集合，不允许ϵ转移
- 接受输入：输入完字符串，TM停在某终态（反之停在非终态、进入无限循环（我们假定到终态即停机，可以证明任何一个图灵机都等价于一个到终态即停机的TM））
- 注意，图灵机可以不接收完输入串

图灵机接受语言： aa^*

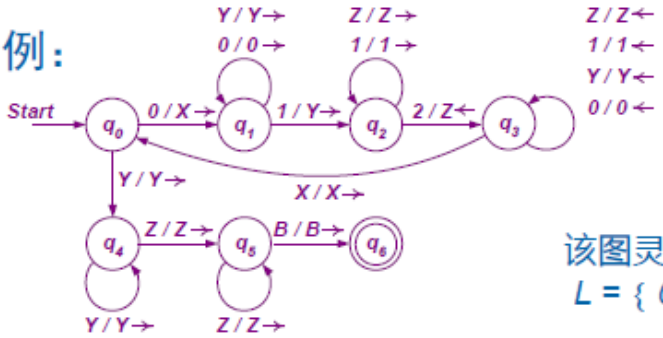


TM接受的语言 $\{a^n b^n | n \geq 1\}$



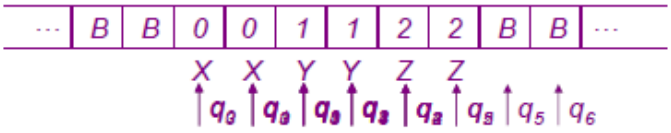
图灵机例

另例：



该图灵机接受的语言为

$L = \{ 0^n 1^n 2^n \mid n \geq 1 \}$



定义 ID's 之间的推导关系 $\vdash_M$（或 $\vdash$）如下：

1. 设 $\delta(q, X_i) = (p, Y, L)$, 则有

$X_1X_2\ldots X_{i-1}qX_iX_{i+1}\ldots X_n \vdash_M X_1X_2\ldots X_{i-2}pX_{i-1}Y\ldots X_n$

两个例外：

(1) i=1 时，$qX_1X_2\ldots X_n \vdash_M pBYX_2\ldots X_n$

(2) i=n 且 Y=B 时，

$X_1X_2\ldots X_{n-1}qX_n \vdash_M X_1X_2\ldots X_{n-2}pX_{n-1}$

2. 设 $\delta(q, X_i) = (p, Y, R)$, 则有:

$X_1X_2\ldots X_{i-1}qX_iX_{i+1}\ldots X_n \vdash_M X_1X_2\ldots X_{i-1}YpX_{i+1}\ldots X_n$

两个例外：

(1) i=n 时，$X_1X_2\ldots X_{n-1}qX_n \vdash_M X_1X_2\ldots X_{n-1}YpB$

(2) i=1及 Y=B 时，

$qX_1X_2\ldots X_n \vdash_M pX_2\ldots X_{n-1}X_n$.

类似，可定义推导关系 $\vdash_M$ 的传递闭包记为 $\vdash^*_M$（或 $\vdash^*$）

总结一下：状态右边一定要有符号，与状态右边无关的空白符可以删去

递归可枚举语言：有定理称任何图灵机都可以转化为到达终态就停机的图灵机（根据原来的定义，有可能还会继续，以后假定到达终态一定停机

递归语言：对于任何不属于L(M)的字符串也可以使得M停机。递归语言对应的问题是可判定的。

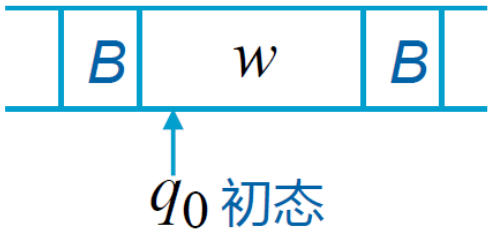
Chapter 13

单位制：00000（十进制的5）

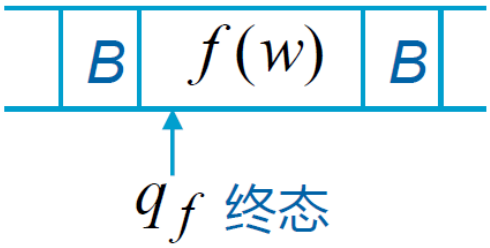
定义:

函数 f 是可计算的，如果存在图灵机M，对任意 $w \in D$，满足

开始结构

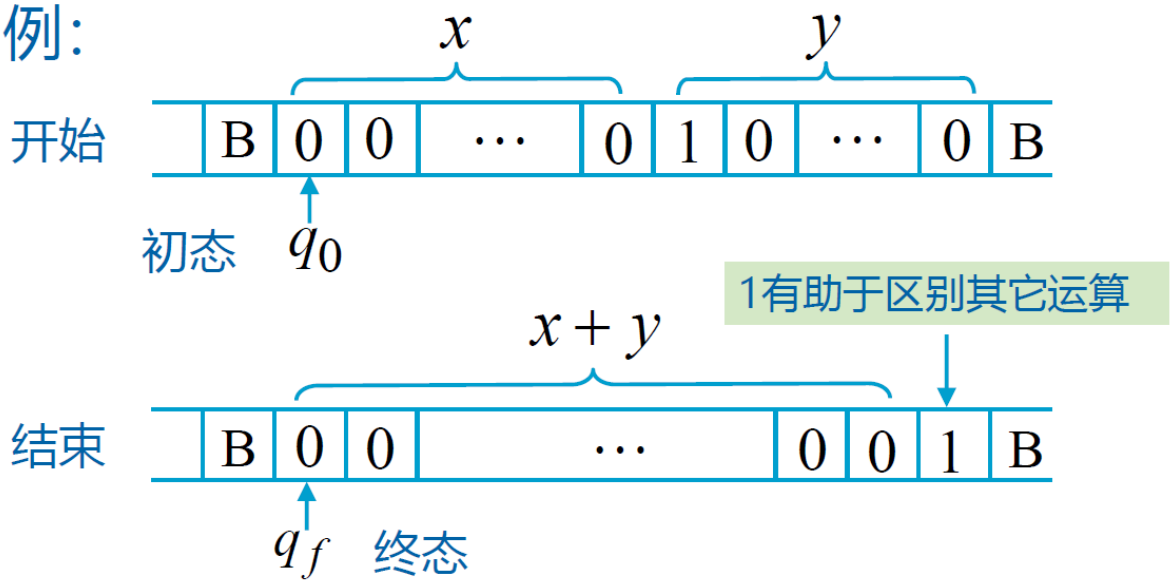


最终结构

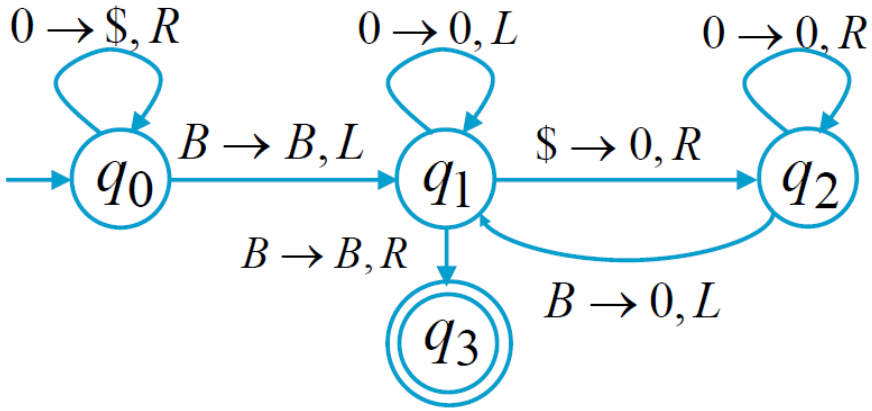


f可计算：$\forall w \in D$, 有 $f: q_0w \vdash^* q_f f(w)$，例如 $f(x,y) = x + y$，输入x1y则输出xy1（单位制下）

例：

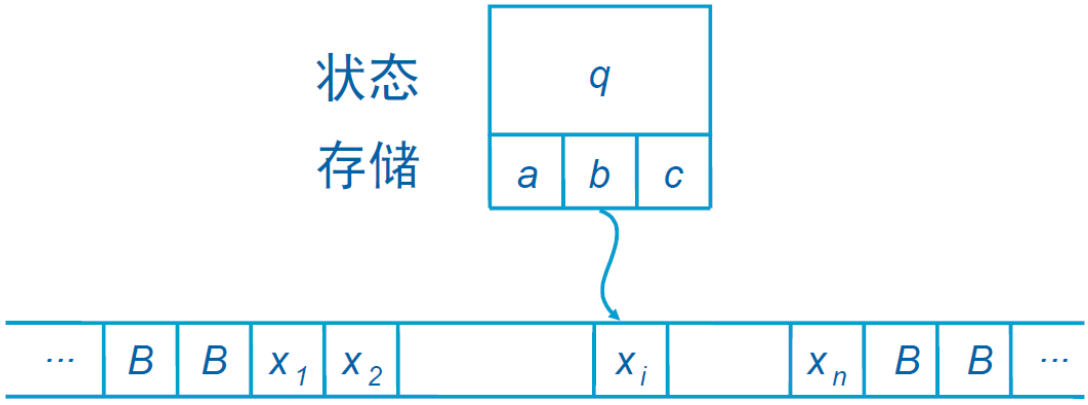


$f(x) = 2x$ 的图灵机为：



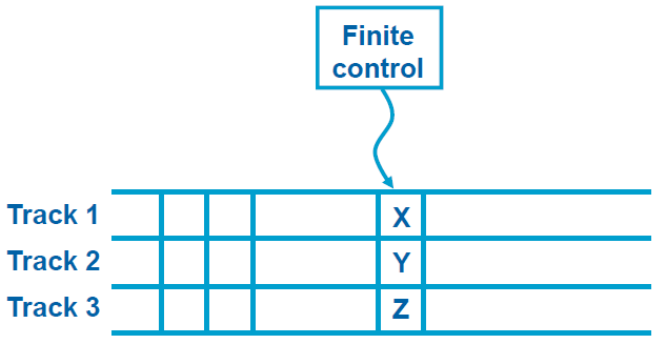
计算后要回到初始状态

带存储区的状态 (*storage in the state*)



相当于对**状态**进行扩展，扩展为q'=[q,a,b,c]，但存储区只能有有限个

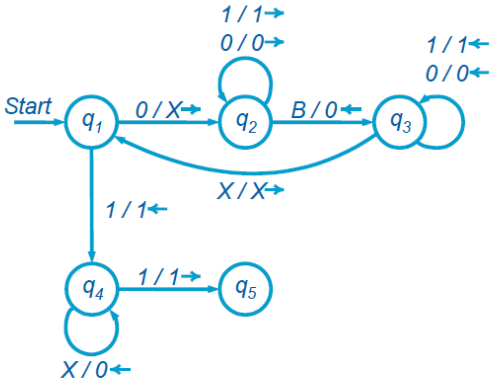
多道 (*multiple tracks*) 图灵机



此类图灵机 $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ 中，带符号是向量的形式。如上图中的图灵机，带符号的形式为三元组。

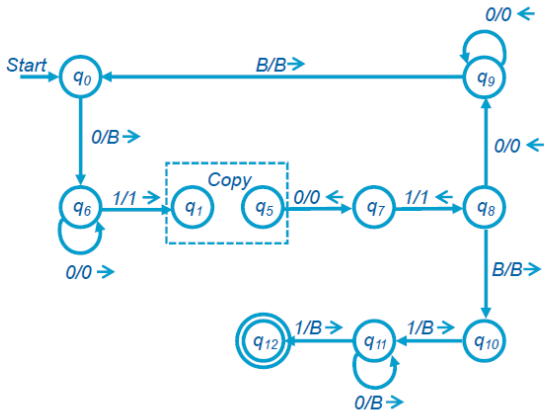
相当于对**带符号**进行扩展，带符号变为[X,Y,Z]

子例程的设计



完成两个正整数的乘法. 初始时，带上的符号串形如 0^m10^n1，而结束时，带上的符号串变为0^{mn}。

左图的图灵机表示子例程 *copy* ，下图的图灵机表示可以调用*copy* 的主程序。



第一类TM结构对应模拟TM的结构

原TM　　最终结构

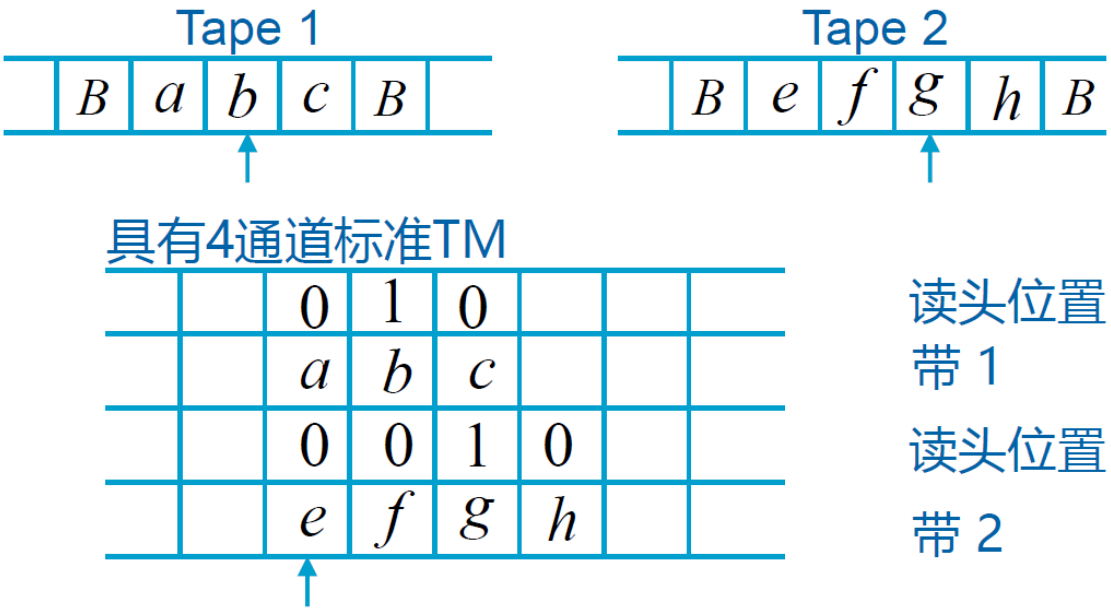
$d_0 \vdash d_1 \vdash \dots \vdash d_n$　　d_f

$d_0' \vdash^* d_1' \vdash^* \dots \vdash^* d_n'$　　d_f'

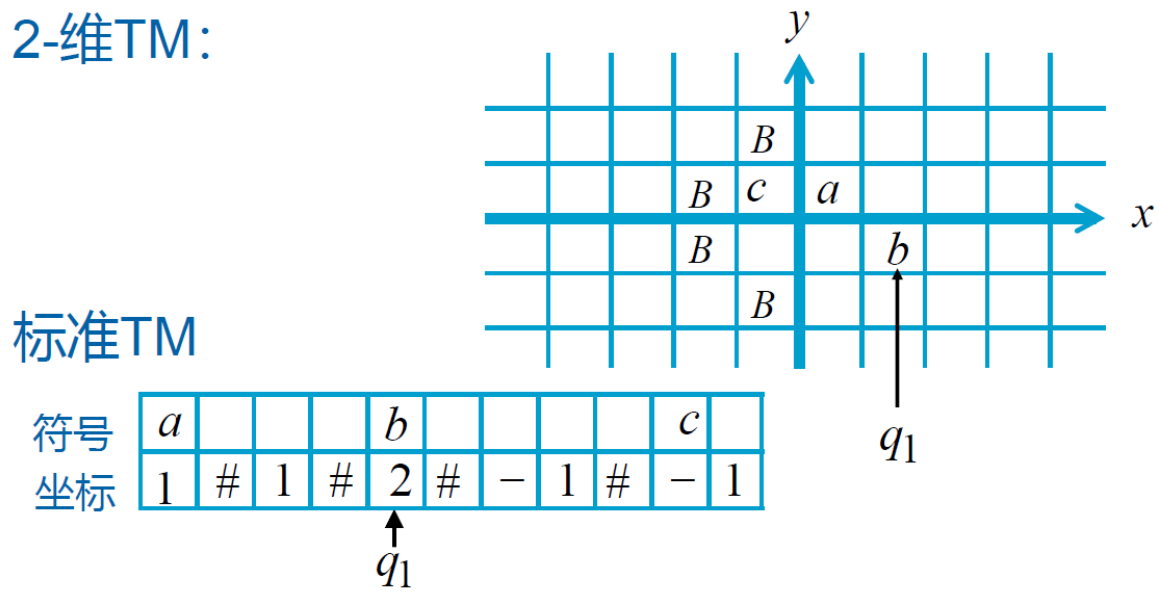
模拟TM

模拟TM与原TM接受相同的语言

多带图灵机：第一条带处于输入符号的最左端，其余读头任意放置；**k条带可用2k通道的TM模拟**

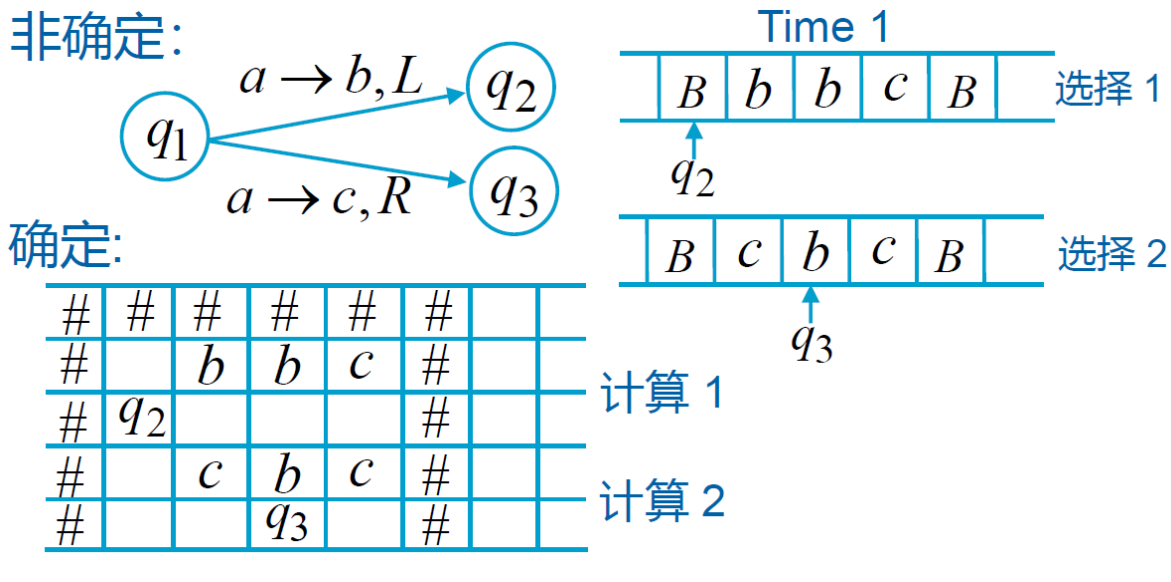


语言L$\{a^n b^n\}$；标准图灵机O(n^2)，2带O(n)



2维TM用二通道模拟

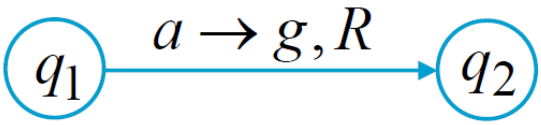
非确定图灵机：转移映射到一个三元组集合



可以用一个二维图灵机模拟，每一步复制当前结构并改变副本中的状态

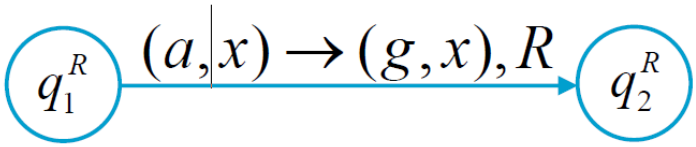
两道半无限带机可以模拟TM

标准图灵机:

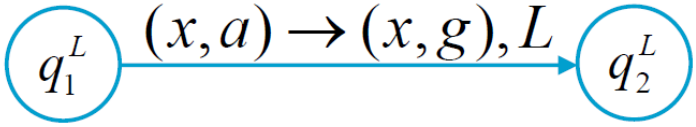


半无限带机:

右部



左部



x 为任意符号

Time 1

右部

左部

#	d	e	B	B	B	
#	c	b	g	B	B	

q_1^L

.....

Time 2

右部

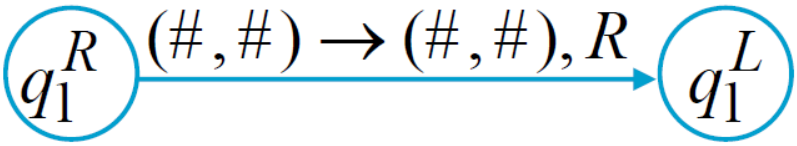
左部

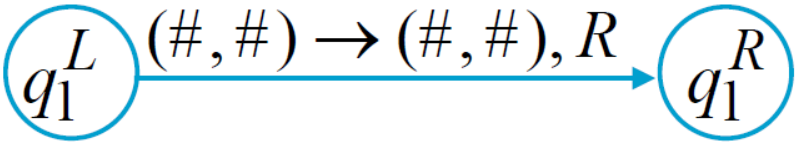
#	d	e	B	B	B	
#	c	b	g	B	B	

q_1^R

.....

半无限带机在分界端:

右部 

左部 

传递函数是非确定的

多栈机

k个栈的多栈机可以用k+1个带的多带图灵机模拟（其中一个用来扫描输入）

一个双栈机可以模拟标准图灵机：一个堆栈模拟读头左边单元格，一个模拟右边

计数器机

计数器机只有两个栈符号Z_0, X，X只能被替换为X^i，Z_0只能被替换为X^iZ_0

定理：一个计数器的计数器机语言接受能力相当于确定下推自动机KDPA；两个相当于图灵机

Chapter 14

图灵机编码:

对图灵机作一些假定:

(1) 输入字母表为{0,1}

(2) 设有限状态为q_1， q_2，…， q_k，并假定初态总是q_1，终态总是 q_2（假定一个终态即可）

(3) 设带符号为X_1， X_2，…， X_m，并假定 X_1 总代表 0，X_2 总代表 1， X_3 总代表 B

(4) 假定带头移动方向为D_1 和 D_2，分别代表 L 和 R

假定

- 字母表为0,1
- 初态1，终态2
- 带符号 0编码1 1编码2 B编码3
- 带头移动 L编码1 R编码2

图灵机的二进制编码：状态转移函数

$$\delta(q_i, X_j) = (q_k, X_l, D_m)$$

可以编码为：　　$0^i10^j10^k10^l10^m$

所有转移规则的编码排列在一起可以作为该图灵机的编码。例如：

$$11C_111C_211\ldots C_{n-1}11C_n$$

Xsu1023 6月1日

可以有多个编码，
对应的正整数不唯一

输入字符串w编码：1w，(Eg: 0000000对应128个输入字符串)

对角线语言：$L_d = \{w_i | w_i \notin L(M_i)\}$，不是递归可枚举语言

- 对角线语言 L_d

  定义：对任何 $i \geq 1$，设M_i为第 i 个图灵机。称语言

  $$L_d = \{ w_i \mid w_i \notin L(M_i) \}$$

  为对角线（对角化）语言。

通用语言：L_U M_i111w_i，递归可枚举语言

- 通用语言
  - 设 M 为接受二进制输入串的图灵机，M的二进制编码为 C，w 为$(0+1)^*$ 中的串C'
  - 图灵机与输入串偶对 (M, w) 的二进制编码记为：

    $$C\,111\,C'$$

- 定义：称语言

  $$L_u = \{C\,111\,C' \mid (M, w) \in \{0,1\}^*, w \in L(M)\}$$

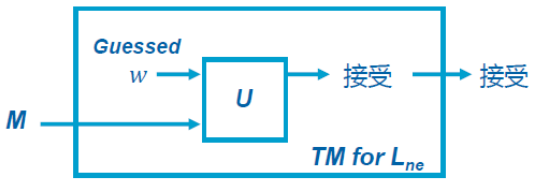
  为通用语言。

语言非空图灵机是不可判定的，但是是RE的

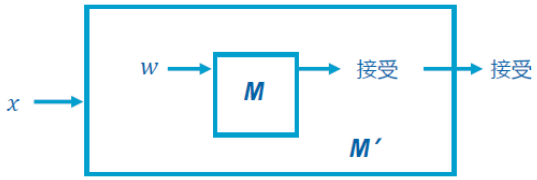
- **语言非空图灵机**

该问题可对应语言：

$$L_{ne}=\{\, M \mid L(M) \neq \varnothing \,\}$$



- L_{ne}可以简约到通用语言 L_u，则该问题是部分可判定的；
- 而 L_u 也可以简约到 L_{ne}，则该问题是不可判定的。



定理：L_{ne} 是递归可枚举语言，但不是递归语言。

则语言为空图灵机非RE

RICE引理描述的是图灵机编码的性质，即它是将一些接受特定语言的图灵机挑出来

- 一台图灵机接受的语言是否是正则语言/CFL均是不可判定的

定理：

- 递归语言的补也是递归语言
- 若语言L和L的补都是递归可枚举语言，则二者均是递归语言

定义：

- 若一个问题对应的语言是递归语言，则可判定，否则不可判定
- 若一个问题对应的语言是递归可枚举语言，则称为**部分可判定**

问题P1到P2的简约：$P_1 \leq_T P_2$，则P_2可判定/部分可判定 $\Rightarrow$ P_1也；P_1不可判定/非RE $\Rightarrow$ P_2也

图灵机停机问题：任给图灵机M和字符串w，问是否停机——不可判定

- **Post 对应问题（PCP）**

PCP的实例包含同一字母表上数量相同的两组字符串：

$$A = w_1, w_2, \ldots, w_k$$

$$B = x_1, x_2, \ldots, x_k$$

(w_i, x_i) $1 \leq i \leq k$ 称为对应对。

称PCP的该实例有解，当且仅当存在整数序列：

$$i_1, i_2, \ldots, i_m$$

使得 $w_{i_1} w_{i_2} \cdots w_{i_m} = x_{i_1} x_{i_2} \cdots x_{i_m}$

例：设 Σ={0,1}，两组字符串A，B由下图定义

	A	B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

图(1)

	A	B
i	w_i	x_i
1	10	101
2	011	11
3	101	011

图(2)

a) 图(1) PCP 的实例有解，其中一个解为: 2,1,1,3
即 $w_2w_1w_1w_3 = x_2x_1x_1x_3 = 101111110$

b) 图(2) PCP 的实例无解。

该问题不是可判定问题

- 图灵机的时间复杂度

  如果对任何长为n的输入串w，图灵机 M 可以在最多 $T(n)$ 移动步停机，则称图灵机 M 的时间复杂度为$T(n)$。

- 非确定图灵机的时间复杂度

  若对任何长为n 的输入串 w，非确定图灵机 M 的任何一个转移序列，可以最多 $T(n)$ 个移动步停机，则称非确定图灵机 M 的复杂度为$T(n)$。

非确定图灵机时间复杂度：树高

- $\mathcal{P}$ 问题

  如果问题（语言）L 满足：存在一确定图灵机 M，使得 $L=L(M)$，且 M 的时间复杂度 $T(n)$ 为多项式，则称该问题是 $\mathcal{P}$ 问题，即 L 属于$\mathcal{P}$.

- $\mathcal{NP}$ 问题

  如果问题（语言）L 满足：存在一个非确定图灵机 M，使得 $L=L(M)$，且 M 的时间复杂度 $T(n)$ 为多项式，则称该问题是 $\mathcal{NP}$ 问题，即 L 属于$\mathcal{NP}$.

复杂度归约$P_1 \leq_P P_2$

- 若P2是P问题，则P1也是；**若P2是NP问题，则P1也是**

- **若P1不是P问题，则P2也不是；若P1不是NP问题，则P2也不是**

NP-hard：任何一个NP问题都可以简约到该问题上

如果可以证明某个NPC是P的，则有P=NP