形式语言与自动机理论

Formal Languages and Automata Theory

蒋宗礼

课程目的和基本要求

- **课程性质**
  - 技术基础
- **基础知识要求**
  - 数学分析（或者高等数学），离散数学
- **主要特点**
  - 抽象和形式化
  - 理论证明和构造性
  - 基本模型的建立与性质

课程目的和基本要求

- 本专业人员4种基本的专业能力
  - 计算思维能力
  - 算法的设计与分析能力
  - 程序设计和实现能力
  - 计算机软硬件系统的认知、分析、设计与应用能力
- 计算思维能力
  - 逻辑思维能力和抽象思维能力
  - 构造模型对问题进行形式化描述
  - 理解和处理形式模型

课程目的和基本要求

- 知识
  - 掌握正则语言、下文无关语言的文法、识别模型及其基本性质、图灵机的基本知识。
- 能力
  - 培养学生的形式化描述和抽象思维能力。
  - 使学生了解和初步掌握“问题、形式化描述、自动化（计算机化）”这一最典型的计算机问题求解思路。

主要内容

- 语言的文法描述。
- **RL**
  - **RG、 FA、RE、RL的性质 。**
- **CFL**
  - **CFG(CNF、GNF)、PDA、CFL的性质。**
- **TM**
  - 基本**TM**、构造技术、**TM**的修改。
- **CSL**
  - **CSG、LBA。**

教材及主要参考书目

1. 蒋宗礼，姜守旭. 形式语言与自动机理论. 北京：清华大学出版社，2003年
2. **John E Hopcroft, Rajeev Motwani, Jeffrey D Ullman. Introduction to Automata Theory, Languages, and Computation (2nd Edition). Addison-Wesley Publishing Company, 2001**
3. **John E Hopcroft, Jeffrey D Ullman. Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Publishing Company, 1979**

第1章 绪论

- **1.1 集合的基础知识**
- **1.1.1 集合及其表示**
  - 集合：一定范围内的、确定的、并且彼此可以区分的对象汇集在一起形成的整体叫做**集合(set)**，简称为集**(set)**。
  - 元素：集合的成员为该集合的**元素(element)**。
  - 集合描述形式。
  - 基数。
  - 集合的分类。

1.1.2 集合之间的关系

- 子集
  - 如果集合**A**中的每个元素都是集合**B**的元素，则称集合**A**是集合**B**的**子集(subset)**，集合**B**是集合**A**的**包集(container)**。记作$A \subseteq B$。也可记作$B \supseteq A$。$A \subseteq B$读作集合**A**包含在集合**B**中；$B \supseteq A$读作集合**B**包含集合**A**。
  - 如果$A \subseteq B$，且$\exists x \in B$，但$x \notin A$，则称**A**是**B**的**真子集(proper subset)**，记作$A \subset B$

1.1.2 集合之间的关系

- 集合相等
  - 如果集合**A**，**B**含有的元素完全相同，则称集合**A**与集合**B**相等**(equivalence)**，记作**A=B**。
- 对任意集合**A**、**B**、**C**：

⑴ **A=B iff A⊆B**且**B⊆A**。

⑵ 如果**A⊆B**，则**|A|≤|B|**。

⑶ 如果**A⊂B**，则**|A|≤|B|**。

⑷ 如果**A**是有穷集，且**A⊂B**，则**|B|>|A|**。

1.1.2 集合之间的关系

(5) 如果$A \subseteq B$，则对$\forall x \in A$，有$x \in B$。

(6) 如果$A \subset B$，则对$\forall x \in A$，有$x \in B$并且$\exists x \in B$，但$x \notin A$。

(7) 如果$A \subseteq B$且$B \subseteq C$，则$A \subseteq C$。

(8) 如果$A \subset B$且$B \subset C$，或者$A \subseteq B$且$B \subset C$，或者$A \subset B$且$B \subseteq C$，则$A \subset C$。

(9) 如果$A=B$，则$|A|=|B|$。

1.1.3 集合的运算

- **并(union)**
- **A与B的并(union)是一个集合，该集合中的元素要么是A的元素，要么是B的元素，记作A∪B。**

$A \cup B=\{a|a \in A$或者$a \in B\}$

$A_1 \cup A_2 \cup \ldots \cup A_n=\{a|\exists i，1 \leqslant i \leqslant n，$使得$a \in A_i\}$

$A_1 \cup A_2 \cup \ldots \cup A_n \cup \ldots=\{a|\exists i，i \in N,$使得$a \in A_i\}$

$$\bigcup_{i=1}^{\infty} A_i$$

$$\bigcup_{A \in S} A = \{a \mid \exists A \in S, a \in A\}$$

交(intersection)

- 集合**A**和**B**中都有的所有元素放在一起构成的集合为**A**与**B**的**交**，记作**A∩B**。

  A∩B={a|a∈A且a∈B}

- **“∩”**为交运算符，**A∩B**读作**A**交**B**。
- 如果**A∩B= Φ**，则称**A**与**B**不相交。
- ⑴ **A∩B= B∩A**。

  ⑵ **(A∩B)∩C=A∩(B∩C)**。

  ⑶ **A∩A=A**。

交(intersection)

(4) $A \cap B = A$ iff $A \supseteq B$。

(5) $\Phi \cap A = \Phi$。

(6) $|A \cap B| \leqslant \min\{|A|, |B|\}$。

(7) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$。

(8) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$。

(9) $A \cap (A \cup B) = A$。

(10) $A \cup (A \cap B) = A$。

差(difference)

- 属于A，但不属于B的所有元素组成的集合叫做A与B的差，记作A-B。

  $A-B=\{a|a\in A且a\notin B\}$

- “-”为减(差)运算符，A-B读作A减B。
- ⑴ $A-A=\Phi$。

  ⑵ $A-\Phi=A$。

  ⑶ $A-B \neq B-A$。

  ⑷ $A-B=A$ iff $A\cap B=\Phi$。

  ⑸ $A\cap(B-C)=(A\cap B)-(A\cap C)$。

  ⑹ $|A-B|\leqslant|A|$。

对称差(symmetric difference)

- 属于**A**但不属于**B**，属于**B**但不属于**A**的所有元素组成的集合叫**A**与**B**的**对称差，记作**$A \oplus B$。

$$A \oplus B=\{a \mid a \in A \text{且} a \notin B \text{或者} a \notin A \text{且} a \in B\}$$

- “$\oplus$”为对称差运算符。$A \oplus B$读作A对称减B。
- $A \oplus B=(A \cup B)-(A \cap B)=(A-B) \cup (B-A)$。

笛卡儿积(Cartesian product)

- **A与B的笛卡儿积(Cartesian product)是一个集合，该集合是由所有这样的有序对(a，b)组成的：其中a∈A，b∈B，记作A×B。**

  A×B={(a，b)|a∈A& b∈B}。
- **“×”为笛卡儿乘运算符。A×B读作A叉乘B。**
- **⑴ A×B≠B×A。**

  ⑵ (A×B)×C≠A×(B×C)。

  ⑶ A×A≠A。

  ⑷ A× *Φ*= *Φ*。

笛卡儿积(Cartesian product)

⑸ **A×(B∪C)=(A×B)∪(A×C)。**

⑹ **(B∪C)×A=(B×A)∪(A×C)。**

⑺ **A×(B∩C)=(A×B)∩(A×C)。**

⑻ **(B∩C)×A=(B×A)∩(C×A)。**

⑼ **A×(B-C)=(A×B)-(A×C)。**

⑽ **(B-C)×A=(B×A)-(C×A)。**

⑾ 当**A、B**为有穷集时，**|A×B|=|A|*|B|。**

幂集(power set)

- A**幂集(power set)**是一个集合，该集合是由**A**的所有子集组成的，记作2^A。
- $2^A=\{B|B\subseteq A\}$。
- 2^A读作**A**的幂集。

幂集(power set)

(1) $\Phi \in 2^{A}$。

(2) $\Phi \subseteq 2^{A}$。

(3) $\Phi \subset 2^{A}$。

(4) $2^{\Phi}=\{\Phi\}$。

(5) $A \in 2^{A}$。

(6) 如果A是有穷集，则$|2^{A}|=2^{|A|}$。

(7) $2^{A \cap B}=2^{A} \cap 2^{B}$。

(8) 如果$A \subseteq B$，则$2^{A} \subseteq 2^{B}$。

补集(complementary set)

A是论域U上的一个集合，A**补集**是由U中的、不在A中的所有元素组成的集合，记作

$$\overline{A} = U - A$$

$$\overline{\Phi} = U$$

$$\overline{U} = \Phi$$

补集(complementary set)

如果**A⊆B**，则 $\overline{B} \subseteq \overline{A}$ 。

$A \cup \overline{A} = U$。

$A \cap \overline{A} = \Phi$ 。

$B = \overline{A} \Leftrightarrow A \cup B = U \,\&\, A \cap B = \Phi$ 。

$\overline{A \cap B} = \overline{A} \cup \overline{B}$ 。

$\overline{A \cup B} = \overline{A} \cap \overline{B}$ 。

1.2 关系

- 二元关系
- 递归定义与归纳证明
- 关系的闭包

1.2.1 二元关系(binary relation)

- **二元关系**
  - 任意的R⊆A×B，R是A到B的**二元关系**。
  - (a，b) ∈R，也可表示为：aRb。
  - A称为**定义域**(domain)，B称为**值域**(range)。
  - 当A=B时，则称R是A上的二元关系。
- **二元关系的性质**
  - 自反(reflexive)性、反自反(irreflexive)性、对称(symmetric)性、反对称(asymmetric)性、传递(transitive)性。

1.2.1 二元关系(binary relation)

- **三歧性**
  - 自反性、对称性、传递性。
- **等价关系**(equivalence relation)
  - 具有三歧性的二元关系称为**等价关系**。

1.2.1 二元关系(binary relation)

- **等价类** (equivalence class)

  S的满足如下要求的划分：S_1、S_2、S_3、…、S_n…称为S关于R的等价划分，S_i称为等价类。

  ⑴ $S= S_1 \cup S_2 \cup S_3 \cup \ldots \cup S_n \cup \ldots$;

  ⑵ 如果$i \neq j$，则$S_i \cap S_j = \Phi$;

  ⑶ 对任意的i，S_i中的任意两个元素a、b，aRb恒成立;

  ⑷ 对任意的i，j，$i \neq j$，S_i中的任意元素a和S_j中的任意元素b，aRb恒不成立

1.2.1 二元关系(binary relation)

- **指数**(index)
  - 把**R**将**S**分成的等价类的个数称为是**R**在**S**上的**指数**。如果**R**将**S**分成有穷多个等价类，则称**R**具有有穷指数；如果**R**将**S**分成无穷多个等价类，则称**R**具有无穷指数。
  - 给定集合**S**上的一个等价关系**R**，**R**就确定了**S**的一个等价分类，当给定另一个不同的等价关系时，它会确定**S**的一个新的等价分类。

1.2.1 二元关系(binary relation)

- **关系的合成 (composition)**

  设$R_1 \subseteq A \times B$是A到B的关系、$R_2 \subseteq B \times C$是B到C的关系，R_1与R_2的**合成**R_1R_2是A到C的关系：$R_1R_2=\{(a, c) \mid \exists(a, b) \in R_1$且$(b, c) \in R_2$。

1.2.1 二元关系(binary relation)

(1) $R_1R_2 \neq R_2R_1$。

(2) $(R_1R_2)R_3=R_1(R_2R_3)$。 (结合率)

(3) $(R_1 \cup R_2)R_3=R_1R_3 \cup R_2R_3$。 (右分配率)

(4) $R_3(R_1 \cup R_2)=R_3R_1 \cup R_3R_2$。 (左分配率)

(5) $(R_1 \cap R_2)R_3 \subseteq R_1R_3 \cap R_2R_3$。

(6) $R_3(R_1 \cap R_2) \subseteq R_3R_1 \cap R_3R_2$。

1.2.1 二元关系(binary relation)

1. 关系这一个概念用来反映对象——集合元素之间的联系和性质
2. 二元关系则是反映两个元素之间的关系，包括某个元素的某种属性。
3. 对二元关系的性质，要强调全称量词是对什么样的范围而言的。

1.2.2 等价关系与等价类（略）

1.2.3 关系的合成（略）

1.2.4 递归定义与归纳证明

- 递归定义(recursive definition)
  - 又称为归纳定义(inductive definition)，它来定义一个集合。
  - 集合的递归定义由三部分组成：
    - 基础(basis)：用来定义该集合的最基本的元素。
    - 归纳(induction)：指出用集合中的元素来构造集合的新元素的规则。
    - 极小性限定：指出一个对象是所定义集合中的元素的充要条件是它可以通过有限次的使用基础和归纳条款中所给的规定构造出来。

1.2.4 递归定义与归纳证明

- 归纳证明
  - 与递归定义相对应。
  - 归纳证明方法包括三大步：
    - 基础(**basis**)：证明最基本元素具有相应性质。
    - 归纳(**induction**)：证明如果某些元素具有相应性质，则根据这些元素用所规定的方法得到的新元素也具有相应的性质。
    - 根据归纳法原理，所有的元素具有相应的性质。

1.2.4 递归定义与归纳证明

- **定义 1-17**

设**R**是**S**上的关系，我们递归地定义R^n的幂：

⑴ $R^0=\{(a，a)|a\in S\}$。

⑵ $R^i=R^{i-1}R$ $(i=1,2,3,4,5,\ldots)$。

1.2.4 递归定义与归纳证明

例1-17 著名的斐波那契(Fibonacci)数的定义

⑴ 基础：**0**是第一个斐波那契数，**1**第二个斐波那契数；

⑵ 归纳：如果**n**是第**i**个斐波那契数，**m**是第**i+1**个斐波那契数，则**n+m**是第**i+2**个斐波那契数，这里**i**为大于等于**1**的正整数。

⑶ 只有满足**(1)**和**(2)**的数才是斐波那契数

0，1，1，2，3，5，8，13，21，34，55，…

1.2.4 递归定义与归纳证明

例1-18 算术表达式

⑴ 基础：常数是算术表达式，变量是算术表达式；

⑵ 归纳：如果E_1、E_2是表达式，则 $+E_1$、$-E_1$、E_1+E_2、 E_1-E_2 、E_1*E_2 、E_1/E_2、E_1**E_2、$Fun(E_1)$是算术表达式。其中Fun为函数名。

⑶ 只有满足(1)和(2)的才是算术表达式。

1.2.4 递归定义与归纳证明

例1-19 对有穷集合**A**，证明$|2^A|=2^{|A|}$。

证明：

设**A**为一个有穷集合, 施归纳于$|A|$：

⑴ 基础： 当$|A|=0$时,$|2^A|=|\{\Phi\}|=1$。

⑵ 归纳：假设$|A|=n$时结论成立，这里$n\geqslant 0$，往证当$|A|=n+1$时结论成立

$$2^A=2^B\cup\{C\cup\{a\}|C\in 2^B\}$$

$$2^B\cap\{C\cup\{a\}|C\in 2^B\}=\Phi$$

1.2.4 递归定义与归纳证明

$$|2^A|=|2^B \cup \{C \cup \{a\}|C \in 2^B\}|$$
$$=|2^B|+|\{C \cup \{a\}|C \in 2^B\}|$$
$$=|2^B|+|2^B|$$
$$=2*|2^B|$$
$$=2*2^{|B|}$$
$$=2^{|B|+1}$$
$$=2^{|A|}$$

(3) 由归纳法原理，结论对任意有穷集合成立。

1.2.4 递归定义与归纳证明

例1-20 表达式的前缀形式是指将运算符写在前面，后跟相应的运算对象。如：$+E_1$的前缀形式为$+E_1$，E_1+E_2的前缀形式为$+E_1E_2$，E_1*E_2的前缀形式为$*E_1E_2$，E_1**E_2的前缀形式为$** E_1$，$Fun(E_1)$ 的前缀形式为$FunE_1$。

证明例**1-18**所定义的表达式可以用这里定义的前缀形式表示。

1.2.4 递归定义与归纳证明

- 递归定义给出的概念有利于归纳证明。在计算机科学与技术学科中，有许多问题可以用递归定义描述或者用归纳方法进行证明，而且在许多时候，这样做会带来许多方便。
- 主要是掌握**递归定义与归纳证明**的叙述格式。

1.2.5 关系的闭包

- **闭包(closure)**
  - 设**P**是关于关系的性质的集合，关系**R**的**P**闭包**(closure)**是包含**R**并且具有**P**中所有性质的最小关系。
- **正闭包(positive closure)**

(1)$R \subseteq R^+$。

(2)如果**(a，b)，(b，c)**$\in R^+$ 则**(a，c)**$\in R^+$。

(3)除**(1)**、**(2)**外，R^+不再含有其他任何元素。

1.2.5 关系的闭包

- **传递闭包(transitive closure)**
  - 具有传递性的闭包。
  - $\mathbf{R}^+$具有传递性。
- 可以证明，对任意二元关系**R**，

$$\mathbf{R}^+ = \mathbf{R} \cup \mathbf{R}^2 \cup \mathbf{R}^3 \cup \mathbf{R}^4 \cup \ldots$$

- 而且当**S**为有穷集时：

$$\mathbf{R}^+ = \mathbf{R} \cup \mathbf{R}^2 \cup \mathbf{R}^3 \cup \ldots \cup \mathbf{R}^{|S|}$$

1.2.5 关系的闭包

- **克林闭包(Kleene closure) R^***

  (1) $R^0 \subseteq R^*, R \subseteq R^*$。

  (2) 如果(a，b)，(b，c)$\in R^*$ 则(a，c)$\in R^*$。

  (3) 除(1)、(2)外，R^*不再含有其他任何元素。

- **自反传递闭包(reflexive and transitive closure)**

  R^*具有自反性、传递性。

1.2.5 关系的闭包

- 可以证明，对任意二元关系**R**，

  $R^* = R^0 \cup R^+$

  $R^* = R^0 \cup R \cup R^2 \cup R^3 \cup R^4 \cup \ldots$

- 而且当**S**为有穷集时：

  $R^* = R^0 \cup R \cup R^2 \cup R^3 \cup \ldots \cup R^{|S|}$

1.2.5 关系的闭包

- R_1、R_2是S上的两个二元关系

  (1) $\Phi^+ = \Phi$。

  (2) $(R_1^+)^+ = R_1^+$。

  (3) $(R_1^*)^* = R_1^*$。

  (4) $R_1^+ \cup R_2^+ \subseteq (R_1 \cup R_2)^+$。

  (5) $R_1^* \cup R_2^* \subseteq (R_1 \cup R_2)^*$。

1.3 图

- 数学家欧拉(L.Euler)解决著名的哥尼斯堡七桥。
- 直观地讲，图是由一些点和一些连接两点的边组成。
- 含无方向的边的图为无向图，含带有方向的边的图为有向图。

1.3.1 无向图

- **无向图(undirected graph)**
  - 设**V**是一个非空的有穷集合，**E⊆V×V**，**G=(V，E)**称为**无向图(undirected graph)**。其中**V**中的元素称为**顶点(vertex**或**node)**，**V**称为顶点集，**E**中的元素称为**无向边(undirected edge)**，**E**为无向边集。
- **图表示**
  - **V**中称为顶点**v**的元素用标记为**v**的小圈表示，**E**中的元素**(v_1，v_2)**用标记为**v_1，v_2**的顶点之间的连线表示。

1.3.1 无向图

- **路(path)**
  - 如果对于$0 \leqslant i \leqslant k-1$，$k \geqslant 1$，均有$(v_i, v_{i+1}) \in E$，则称$v_0$，$v_1$，…，$v_k$是$G=(V, E)$的一条长为k的路。
- **回路或圈(cycle)**
  - 当路v_0，v_1，…，v_k中$v_0=v_k$时，v_0，v_1，…，v_k叫做一个回路或圈(cycle)。

1.3.1 无向图

- **顶点的度数**
  - 对于**v∈V，|{v|(v，w)∈E}|**称为无向图**G=(V，E)**的顶点**v**的度数，记作**deg(v)**。
  - 对于任何一个图，图中所有顶点的度数之和为图中边的**2**倍。

$$\sum_{v \in V} \deg(v) = 2^* \mid E \mid$$

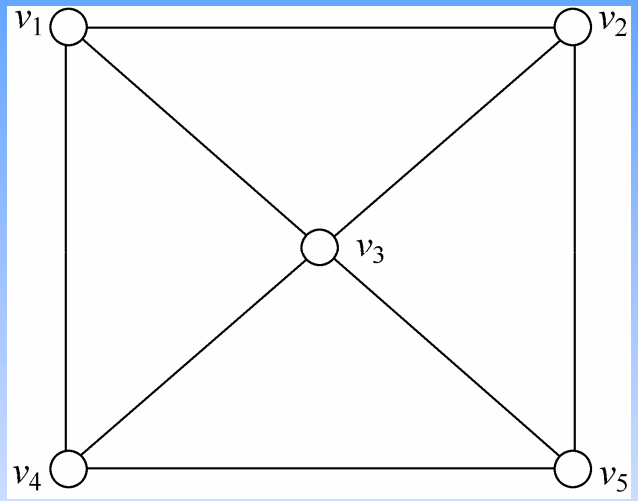
$\deg(v_1)=3$

$\deg(v_2)=3$

$\deg(v_3)=4$

$\deg(v_4)=3$

$\deg(v_5)=3$



$\deg(v_1)+\deg(v_2)+\deg(v_3)+\deg(v_4)+\deg(v_5)=16$

1.3.1 无向图

- **连通图**
  - 如果对于∀**v**，**w**∈**V**，**v**≠**w**，**v**与**w**之间至少有一条路存在，则称**G=(V，E)**是**连通图**。
  - 图**G**是连通的充要条件是**G**中存在一条包含图的所有顶点的路。

1.3.2 有向图

- 有向图(directed graph)
  - G=(V，E)。
  - V：顶点(vertex或node)集。
  - $(v_1，v_2)\in E$：顶点v_1到顶点v_2的有向边(directed edge)，或弧(arc)，v_1称为前导(predecessor)，v_2称为后继(successor)。
- 有向路(directed path)
  - 如果对于$0\leqslant i\leqslant k-1$，$k\geqslant 1$，均有$(v_i，v_{i+1})\in E$，则称$v_0$，$v_1$，…，$v_k$是G的一条长为k的有向路。

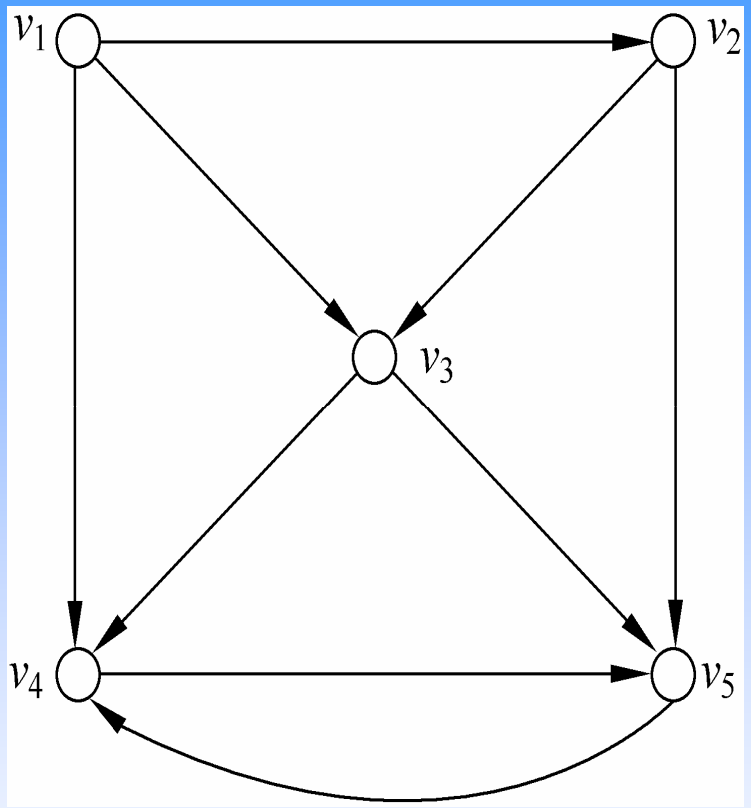
1.3.2 有向图

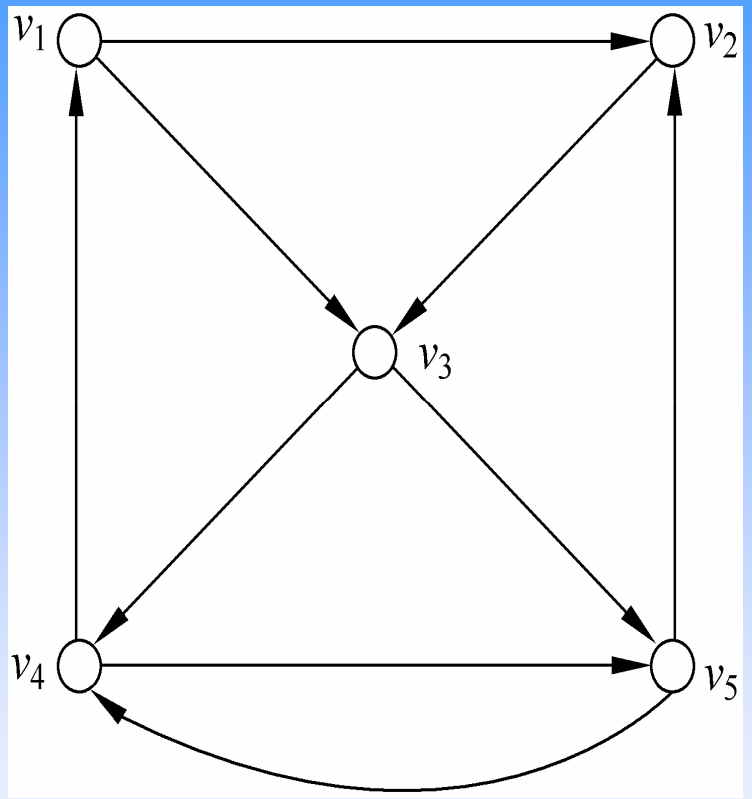
- **有向回路或有向圈(directed cycle)**
  - 对于$0 \leqslant i \leqslant k-1$，$k \geqslant 1$，均有$(v_i, v_{i+1}) \in E$，且$v_0=v_k$，则称$v_0$，$v_1$，…，$v_k$是G的一条长为k的有向路为一个有向回路。
  - 有向回路又叫有向圈。
- 有向图的图表示
  - 图G的图表示是满足下列条件的“图”：其中V中称为顶点v的元素用标记为v的小圈表示，E中的元素(v_1, v_2)用从标记为v_1的顶点到标记为v_2的顶点的弧表示。

1.3.2 有向图

- 顶点的度数
  - **入度(数)：ideg(v)=|{v|(w，v)∈E}|。**
  - **出度(数)：odeg(v)= |{v|(v，w)∈E}|。**
- 对于任何一个有向图，图中所有顶点的入度之和与图中所有顶点的出度之和正好是图中边的个数

$$\sum_{v\in V} i\deg(v) = \sum_{v\in V} o\deg(v) = |E|$$





两个不同的有向图

1.3.3 树

- 满足如下条件的有向图**G=(V，E)**称为一棵**(有序、有向)树(tree)：**
  - **根(root) v：**没有前导，且**v**到树中其他顶点均有一条有向路。
  - 每个非根顶点有且仅有一个前导。
  - 每个顶点的后继按其拓扑关系从左到右排序。

1.3.3 树

•树的基本概念

(1) 顶点也可以成为结点。

(2) 结点的前导为该结点的**父亲(父结点father)**。

(3) 结点的后继为它的**儿子(son)**。

(4) 如果树中有一条从结点v_1到结点v_2的路，则称v_1是v_2的**祖先(ancestor), v2是v1的后代(descendant)**。

(5) 无儿子的顶点叫做**叶子(leaf)**。

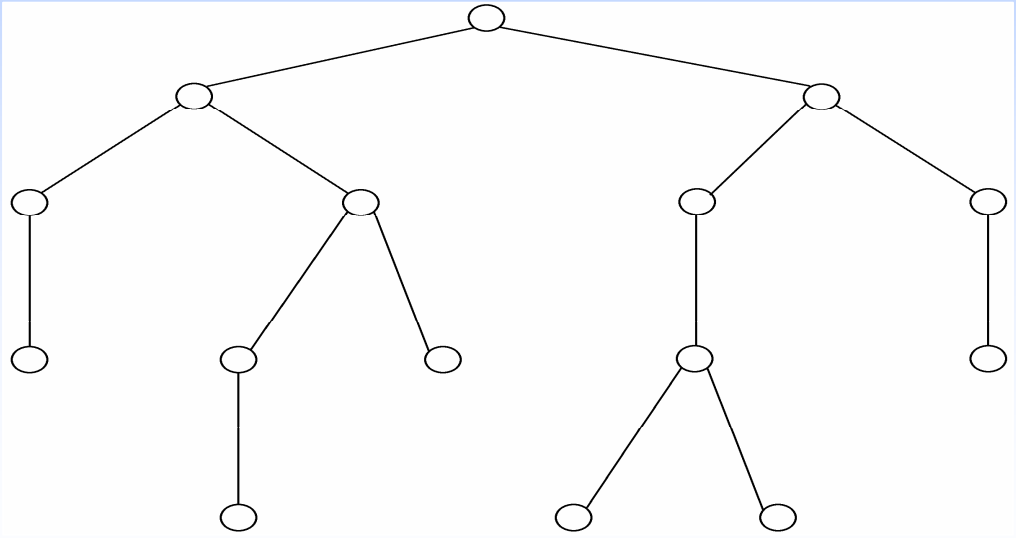
(6) 非叶结点叫做**中间结点(interior)**。

1.3.3 树

- **树的层**
  - 根处在树的第1层(level)。
  - 如果结点v处在第i层(i≥1)，则v的儿子处在第i+1层。
  - 树的最大层号叫做该树的高度(height)。

1.3.3 树

- **二元树**
  - 如果对于∀v∈V，v最多只有2个儿子，则称G=(V，E)为二元树(binary tree)。
  - 对一棵二元树，它的第n层最多有2^{n-1}个结点。一棵n层二元树最多有个2^{n-1}叶子。

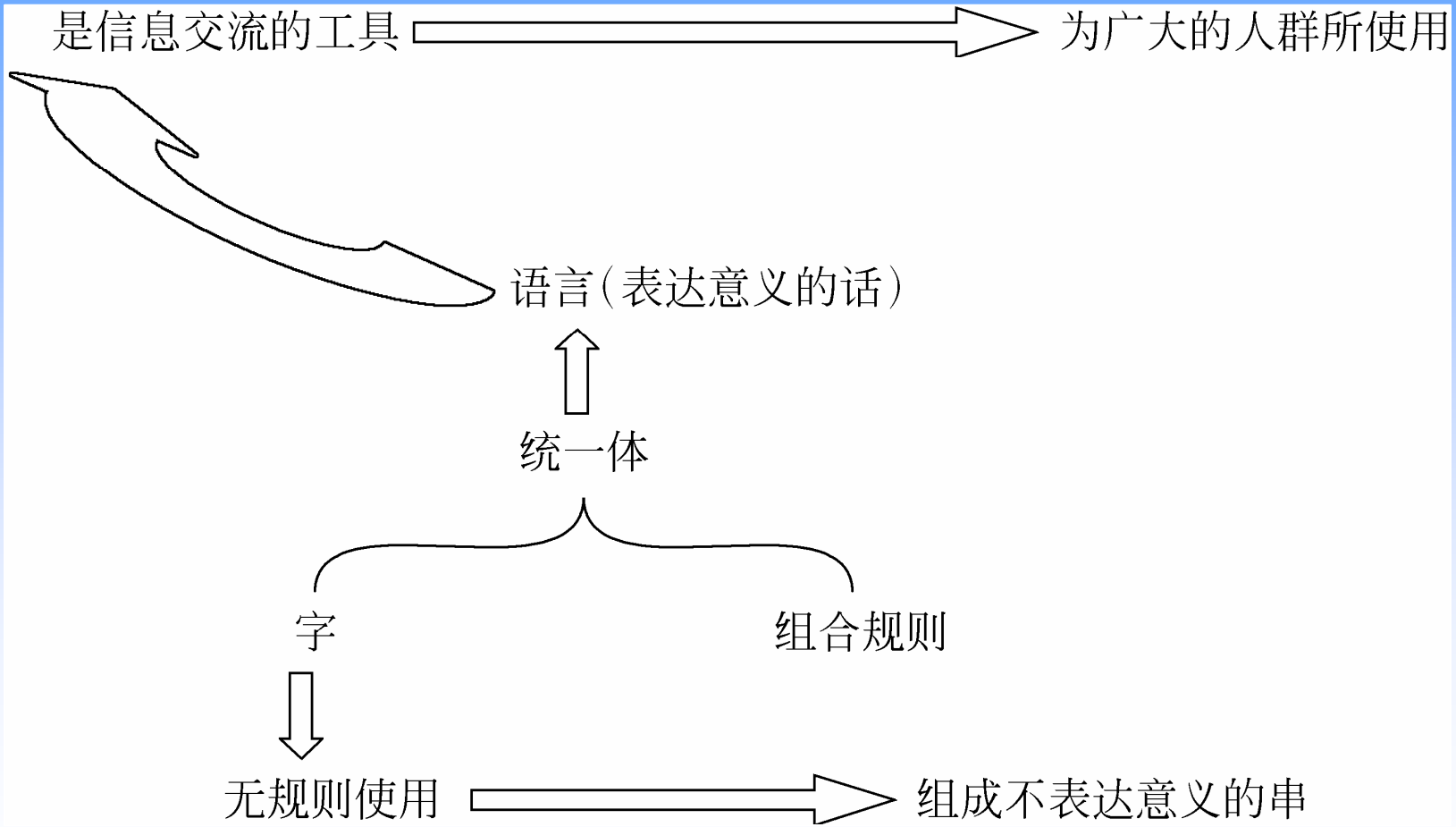


1.4 语言

1.4.1 什么是语言

- 例如：“学大一生是个我”；“我是一个大学生”。
- 语言是一定的群体用来进行交流的工具。
- 必须有着一系列的生成规则、理解(语义)规则。

1.4.1 什么是语言



1.4.1 什么是语言

- 斯大林：从强调语言的作用出发，把语言定义为“为广大的人群所理解的字和组合这些字的方法”。
- 语言学家韦波斯特(Webster) ：为相当大的团体的人所懂得并使用的字和组合这些字的方法的统一体。
- 要想对语言的性质进行研究，用这些定义来建立语言的数学模型是不够精确的。必须有更形式化的定义。

1.4.2 形式语言与自动机理论的产生与作用

- 语言学家乔姆斯基，毕业于宾西法尼亚大学，最初从产生语言的角度研究语言。**1956**年，他将语言**L**定义为一个字母表Σ中的字母组成的一些串的集合：$L \subseteq \Sigma^*$。
- 字母表上按照一定的规则定义一个文法**(grammar)**，该文法所能产生的所有句子组成的集合就是该文法产生的语言。
- **1959**年，乔姆斯基根据产生语言文法的特性，将语言划分成**3**大类。

1.4.2 形式语言与自动机理论的产生与作用

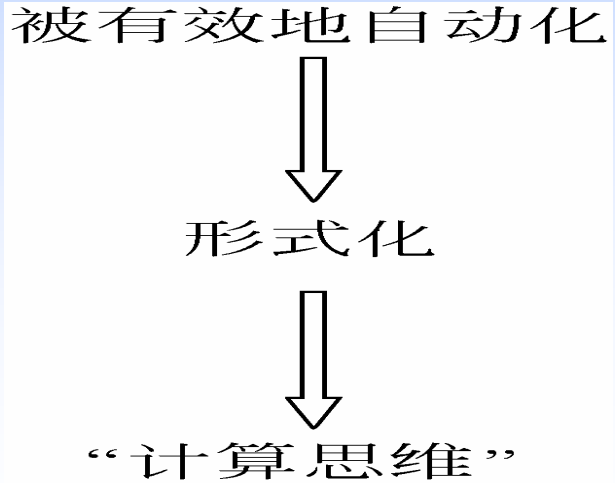
- **1951**年到**1956**年，克林**(Kleene)** 在研究神经细胞中，建立了识别语言的系统——有穷状态自动机。
- **1959**年，乔姆斯基发现文法和自动机分别从生成和识别的角度去表达语言，而且证明了文法与自动机的等价性，这一成果被认为是将形式语言置于了数学的光芒之下，使得形式语言真正诞生了。

1.4.2 形式语言与自动机理论的产生与作用

- **20世纪50年代，巴科斯范式(Backus Nour Form 或 Backus Normal Form，BNF)实现了对高级语言ALGOL-60的成功描述。这一成功，使得形式语言在20世纪60年代得到了大力的发展。尤其是上下文无关文法被作为计算机程序设计语言的文法的最佳近似描述得到了较为深入的研究。**
- 相应的理论用于其他方面。

1.4.2 形式语言与自动机理论的产生与作用

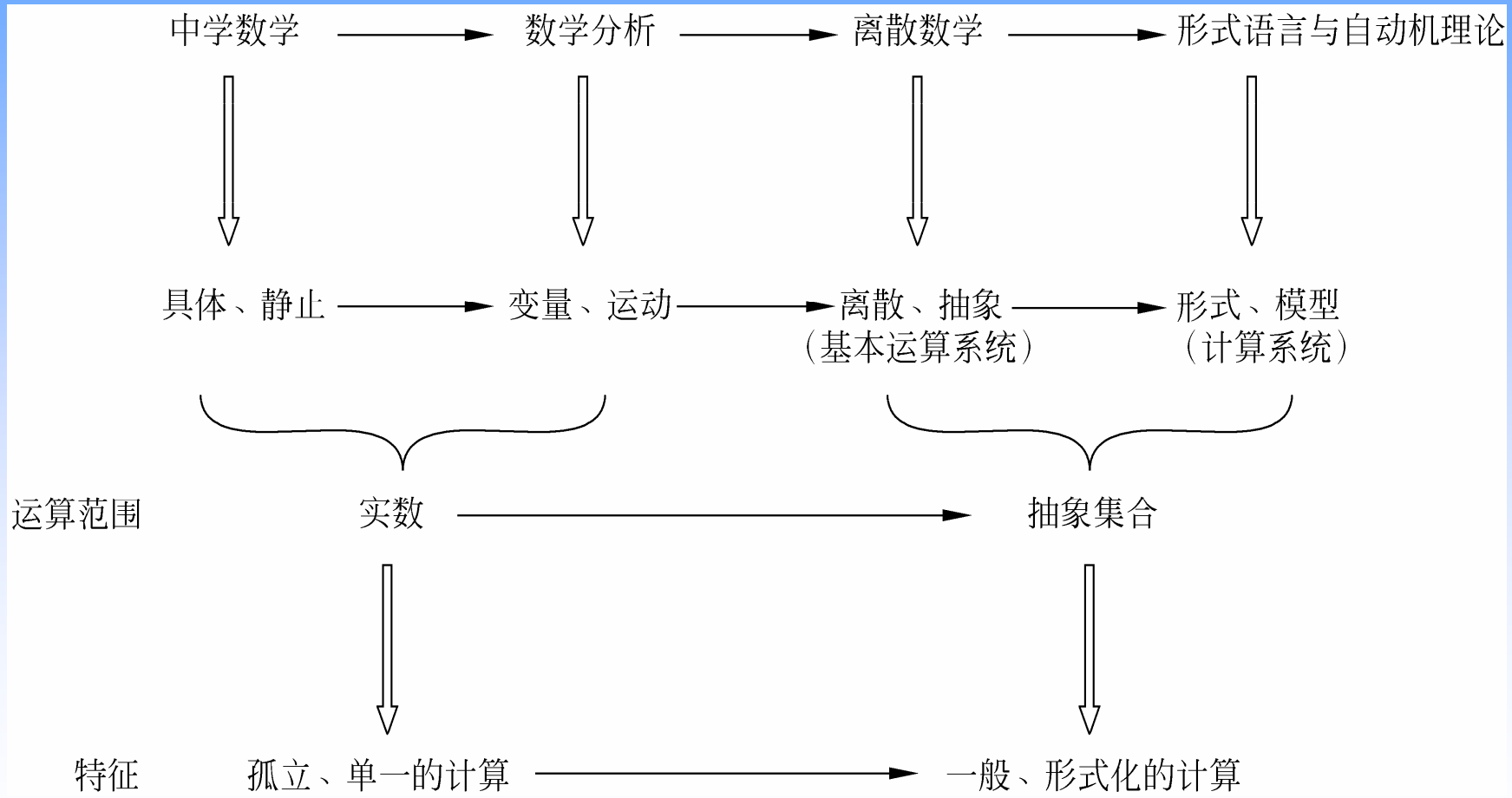
- 形式语言与自动机理论在计算机科学与技术学科的人才的计算思维的培养中占有极其重要的地位。
- 计算学科的主题：“什么能被有效地自动化”。



1.4.2 形式语言与自动机理论的产生与作用

- 计算机科学与技术学科人才专业能力构成
  - “计算思维能力”——抽象思维能力、逻辑思维能力。
  - 算法设计与分析能力。
  - 程序设计与实现能力。
  - 计算机系统的认知、分析、设计和应用能力。

1.4.2 形式语言与自动机理论的产生与作用



1.4.2 形式语言与自动机理论的产生与作用

- 考虑的对象的不同，所需要的思维方式和能力就不同，通过这一系统的教育，在不断升华的过程中，逐渐地培养出了学生的抽象思维能力和对逻辑思维方法的掌握。
- 创新意识的建立和创新能力的培养也在这个教育过程中循序渐进地进行着。
- 内容用于后续课程和今后的研究工作。
- 是进行思维训练的最佳知识载体。
- 是一个优秀的计算机科学工作者必修的一门课程。

1.4.3 基本概念

- 对语言研究的三个方面
  - 表示**(representation)——** 无穷语言的表示。
  - 有穷描述**(finite description) ——**研究的语言要么是有穷的，要么是可数无穷的，这里主要研究可数无穷语言的有穷描述。
  - 结构**(structure)——**语言的结构特征。

1.4.3 基本概念

- **字母表(alphabet)**
  - **字母表**是一个非空有穷集合，字母表中的元素称为该字母表的一个**字母(letter)**。又叫做**符号(symbol)**、或者**字符(character)**。
  - 非空性。
  - 有穷性。
- 例如：

  {a，b，c，d}

  { a，b，c，…，z}

  {0,1}

1.4.3 基本概念

- 字符的两个特性
  - 整体性**(monolith)**，也叫不可分性。
  - 可辨认性**(distinguishable)**，也叫可区分性。
- 例（续）

  {a，a′ ，b，b′ }

  {aa，ab，bb}

  {∞，∧，∨，≥，≤}

1.4.3 基本概念

- 字母表的乘积(product)

  $\Sigma_1\Sigma_2=\{ab|a\in\Sigma_1，\ b\in\Sigma_2\}$

- 例如：

  {0,1}{0,1}={00，01，10，00}

  {0,1}{a，b，c，d}={0a，0b，0c，0d，1a，1b，1c，1d}

  {a，b，c，d}{0,1}={a0，a1，b0，b1，c0，c1，d0，d1}

  {aa，ab，bb}{0,1}={ aa0，aa1，ab0，ab1，bb0，bb1}

1.4.3 基本概念

- 字母表Σ的n次幂

  $\Sigma^0=\{\varepsilon\}$

  $\Sigma^n=\Sigma^{n-1}\Sigma$

  ε是由Σ中的0个字符组成的。

- Σ的正闭包

  $\Sigma^+=\Sigma\cup\Sigma^2\cup\Sigma^3\cup\Sigma^4\cup\ldots$

- Σ的克林闭包

  $\Sigma^*=\Sigma^0\cup\Sigma^+=\Sigma^0\cup\Sigma\cup\Sigma^2\cup\Sigma^3\cup\ldots$

1.4.3 基本概念

- 例如：

  $\{0,1\}^+$=｛0，1，00，01，11，000，001，010，011，100，…｝

  $\{0,1\}^*$=｛ε，0，1，00，01，11，000，001，010，011，100，…｝

  $\{a，b，c，d\}^+$=｛a，b，c，d，aa，ab，ac，ad，ba，bb，bc，bd，…，aaa，aab，aac，aad，aba，abb，abc，…｝

  $\{a，b，c，d\}^*$=｛ε，a，b，c，d，aa，ab，ac，ad，ba，bb，bc，bd，…，aaa，aab，aac，aad，aba，abb，abc,…｝

1.4.3 基本概念

- 结论：

Σ^*={x|x是Σ中的若干个，包括0个字符，连接而成的一个字符串}。

Σ^+={x|x是Σ中的至少一个字符连接而成的字符串}。

1.4.3 基本概念

- **句子(sentence)**

  Σ是一个字母表，$\forall x \in \Sigma^*$，**x**叫做Σ上的一个**句子**。

- 句子相等。

  两个句子被称为相等的，如果它们对应位置上的字符都对应相等。

- **别称**

  字(word)、(字符、符号)行(line)、(字符、符号)串(string)。

1.4.3 基本概念

- **出现(apperance)**
  - **x，y∈Σ*，a∈Σ，句子xay中的a叫做a在该句子中的一个出现。**
  - **当x= ε 时，a的这个出现为字符串xay的首字符**
  - **如果a的这个出现是字符串xay的第n个字符，则y的首字符的这个出现是字符串xay的第n+1个字符。**
  - **当y= ε 时，a的这个出现是字符串xay的尾字符**
  - **例：abaabb。**

1.4.3 基本概念

- **句子的长度(length)**
  - $\forall x \in \Sigma^*$，句子**x**中字符出现的总个数叫做该**句子的长度**，记作**|x|**。
  - 长度为**0**的字符串叫**空句子**，记作 ε 。
- **例如：**

  |abaabb|=6

  |bbaa|=4

  | ε |=0

  |bbabaabbbaa|=11

1.4.3 基本概念

- 注意事项
  - ε 是一个句子。
  - { ε }≠ Φ。这是因为{ ε }不是一个空集，它是含有一个空句子 ε 的集合。|{ ε }|=1，而 |Φ|=0。

1.4.3 基本概念

- **并置(concatenation)**
  - **x，y**$\in \Sigma^*$，**x，y的并置**是由串**x**直接相接串**y**所组成的。记作**xy**。并置又叫做**连结**。
- 串**x**的**n**次**幂**

  $x^0 = \varepsilon$

  $x^n = x^{n-1}x$

1.4.3 基本概念

- 例如：
  - 对x=001，y=1101

    $x^0=y^0=\varepsilon$

    $x^4=001001001001$

    $y^4=1101110111011101$
  - 对x=0101，y=110110

    $x^2=01010101$

    $y^2=110110110110$

    $x^4=0101010101010101$

    $y^4=110110110110110110110110$

1.4.3 基本概念

- Σ^*上的并置运算性质

  ⑴ 结合律：**(xy)z=x(yz)**。

  ⑵ 左消去律：如果**xy=xz**，则**y=z**。

  ⑶ 右消去律：如果**yx=zx**，则**y=z**。

  ⑷ 惟一分解性：存在惟一确定的a_1，a_2，…，$a_n \in \Sigma$，使得$x= a_1a_2\ldots a_n$。

  ⑸ 单位元素：$\varepsilon x=x \varepsilon =x$。

1.4.3 基本概念

- **前缀与后缀**

设x，y，z，w，v∈∑*，且x=yz，w=yv

(1) y是x的前缀(prefix)。

(2)如果z≠ε，则y是x的真前缀(proper prefix)。

(3) z是x的后缀(suffix);

(4) 如果y≠ε，则z是x的真后缀(proper suffix)。

(5) y是x和w的公共前缀(common Prefix)。

1.4.3 基本概念

- **公共前缀与后缀**

(6)如果x和w的任何公共前缀都是y的前缀，则y是x和w的最大公共前缀。

(7) 如果x=zy，w=vy，则y是x和w的公共后缀(common suffix)。

(8)如果x和w的任何公共后缀都是y的后缀，则y是x和w的最大公共后缀。

1.4.3 基本概念

- **例**

字母表Σ={a，b}上的句子abaabb的前缀、后缀、真前缀和真后缀如下：

前缀：ε，a，ab，aba，abaa，abaab，abaabb

真前缀：ε，a，ab，aba，abaa，abaab

后缀：ε，b，bb，abb，aabb，baabb，abaabb

真后缀：ε，b，bb，abb，aabb，baabb

1.4.3 基本概念

- 结论

⑴ x的任意前缀y有惟一的一个后缀z与之对应，使得x=yz；反之亦然。

⑵ x的任意真前缀y有惟一的一个真后缀z与之对应，使得x=yz；反之亦然。

⑶ |{w|w是x的后缀}|=|{w|w是x的前缀}|。

⑷ |{w|w是x的真后缀}|=|{w|w是x的真前缀}|。

⑸ {w|w是x的前缀}={w|w是x的真前缀}∪{x}，

|{w|w是x的前缀}|=|{w|w是x的真前缀}|+1。

1.4.3 基本概念

- 结论

⑹ **{w|w是x的后缀}={w|w是x的真后缀}∪{x}，**

|{w|w是x的后缀}|=|{w|w是x的真后缀}|+1。

⑺ 对于任意字符串**w**，**w**是自身的前缀，但不是自身的真前缀；**w**是自身的后缀，但不是自身的真后缀。

⑻ 对于任意字符串**w**， ε 是**w**的前缀，且是**w**的真前缀； ε 是**w**的后缀，且是**w**的真后缀

1.4.3 基本概念

- 约定

⑴ 用小写字母表中较为靠前的字母**a**，**b**，**c**，…表示字母表中的字母。

⑵ 用小写字母表中较为靠后的字母**x**，**y**，**z**，…表示字母表上的句子。

⑶ 用$\mathbf{x}^{\mathbf{T}}$表示**x**的倒序。例如，如果**x=abc**，则$\mathbf{x}^{\mathbf{T}}$**=cba**。

1.4.3 基本概念

- **子串(substring)**
  - **w，x，y，z∈∑*，且w=xyz，则称y是w的子串。**
- **公共子串(common substring)**
  - **t，u，v，w，x，y，z∈∑*，且t=uyv，w=xyz，则称y是t和w的公共子串(common substring)。如果y_1，y_2，……，y_n是t和w的公共子串，且max{$|y_1|$，$|y_2|$，…，$|y_n|$}=$|y_j|$，则称y_j是t和w的最大公共子串。**
  - 两个串的最大公共子串并不一定是惟一的。

1.4.3 基本概念

- **语言(language)**

  $\forall L \subseteq \Sigma^*$，L称为字母表$\Sigma$上的一个**语言(language)**，$\forall x \in L$，x叫做L的一个句子。

- 例：{0，1}上的不同语言

  {00，11}，{0，1}

  {0，1，00，11}，{0，1，00，11，01，10}

  $\{00，11\}^*$，$\{01，10\}^*$，$\{00，01，10，11\}^*$，

  $\{0\}\{0，1\}^*\{1\}$，$\{0，1\}^*\{111\}\{0，1\}^*$

1.4.3 基本概念

- 语言的**乘积(product)**

$L_1 \subseteq \Sigma_1^*$，$L_2 \subseteq \Sigma_2^*$，语言L_1与L_2的**乘积**是一个语言，该语言定义为：

$L_1L_2=\{xy \mid x \in L_1，y \in L_2\}$

是字母表$\Sigma_1 \cup \Sigma_2$上的语言。

1.4.3 基本概念

- 例

⑴ $L_1=\{0，1\}$。

⑵ $L_2=\{00，01，10，11\}$。

⑶ $L_3=\{0，1，00，01，10，11，000，…\}=\Sigma^+$。

⑷ $L_4=\{\varepsilon，0，1，00，01，10，11，000，…\}=\Sigma^*$。

⑸ $L_5=\{0^n|n\geqslant 1\}$。

⑹ $L_6=\{0^n1^n|n\geqslant 1\}$。

⑺ $L_7=\{1^n|n\geqslant 1\}$。

⑻ $L_8=\{0^n1^m|n，m\geqslant 1\}$。

⑼ $L_9=\{0^n1^n0^n|n\geqslant 1\}$。

⑽ $L_{10}=\{0^n1^m0^k|n，m，k\geqslant 1\}$。

⑾ $L_{11}=\{x|x\in\Sigma^+$且x中0和1的个数相同$\}$。

1.4.3 基本概念

- 上述几个语言的部分特点及相互关系

  上述所有语言都是L_4的子集(子语言)；

  L_1，L_2是有穷语言；其他为无穷语言；其中L_1是Σ上的所有长度为1的句子组成的语言，L_2是Σ上的所有长度为2的句子组成的语言；

  L_3，L_4分别是Σ的正闭包和克林闭包；

  $L_5L_7 \neq L_6$，但$L_5L_7 = L_8$；同样$L_9 \neq L_{10}$，但是我们有：$L_6 \subset L_5L_7$，$L_9 \subset L_{10}$。

1.4.3 基本概念

$L_6=\{0^n1^n|n\geqslant 1\}$中的句子中的0和1的个数是相同的，并且所有的0在所有的1的前面，$L_{11}=\{x|x\in\Sigma^+$且x中0和1的个数相同$\}$中的句子中虽然保持着0的个数和1的个数相等，但它并没要求所有的0在所有的1的前面。例如，0101，$1100\in L_{11}$，但是$0101\notin L_6$，$1100\notin L_6$。而对$\forall x\in L_6$，有$x\in L_{11}$。所以，$L_6\subset L_{11}$。

1.4.3 基本概念

$L_1 \subset L_{12}$，$L_2 \subset L_{12}$

$L_5 \subset L_{12}$，$L_6 \subset L_{12}$

$L_7 \subset L_{12}$，$L_8 \subset L_{12}$

$L_9 \subset L_{12}$，$L_{10} \subset L_{12}$

$L_1 \not\subset L_{10}$，$L_2 \not\subset L_{10}$

$L_5 \not\subset L_{10}$，$L_6 \not\subset L_{10}$

$L_7 \not\subset L_{10}$，$L_8 \not\subset L_{10}$

$L_9 \subset L_{10}$，$L_{10} \subset L_{12}$

1.4.3 基本概念

- 例

(1) $\{x|x=x^T，\ x\in\Sigma\}$。

(2) $\{xx^T|x\in\Sigma^+\}$。

(3) $\{xx^T|x\in\Sigma^*\}$。

(4) $\{xwx^T|x，\ w\in\Sigma^+\}$。

(5) $\{xx^Tw|x，\ w\in\Sigma^+\}$。

1.4.3 基本概念

- **幂**

  $\forall L \in \Sigma^*$，L的n次幂是一个语言，该语言定义为

  ⑴ 当$n=0$是，$L^n=\{\varepsilon\}$。

  ⑵ 当$n \geqslant 1$时，$L^n = L^{n-1}L$ 。

- **正闭包**

  $L^+ = L \cup L^2 \cup L^3 \cup L^4 \cup \ldots$

- **克林闭包**

  $L^* = L^0 \cup L \cup L^2 \cup L^3 \cup L^4 \cup \ldots$

1.5 小结

本章简单叙述了一些基础知识，一方面，希望读者通过对本章的阅读，熟悉集合、关系、图、形式语言等相关的一些基本知识点，为以后各章学习作适当的准备。另一方面，也使读者熟悉本书中一些符号的意义。

1.5 小结

(1) 集合：集合的表示、集合之间的关系、集合的基本运算。

(2) 关系：主要介绍了二元关系相关的内容。包括等价关系、等价分类、关系合成、关系闭包。

(3) 递归定义与归纳证明。

1.5 小结

(4) 图：无向图、有向图、树的基本概念。

(5) 语言与形式语言：自然语言的描述，形式语言和自动机理论的出现，形式语言和自动机理论对计算机科学与技术学科人才能力培养的作用

(6) 基本概念：字母表、字母、句子、字母表上的语言、语言的基本运算

第2章 文法

- 对任何语言L，有一个字母表Σ，使得$L \subseteq \Sigma^*$。
- L的具体组成结构是什么样的？
- 一个给定的字符串是否为一个给定语言的句子?如果不是，它在结构的什么地方出了错？进一步地，这个错误是什么样的错？如何更正？……。
- 这些问题对有穷语言来说，比较容易解决。
- 这些问题对无穷语言来说，不太容易解决。
- 语言的有穷描述。

第2章 文法

- 主要内容

  文法的直观意义与形式定义，推导、文法产生的语言、句子、句型；乔姆斯基体系，左线性文法、右线性文法，文法的推导与归约；空语句。

- 重点

  文法、推导、归约、模型的等价性证明。

- 难点

  形式化的概念，文法的构造。

2.1 启示

- 文法的概念最早是由语言学家们在研究自然语言理解中完成形式化。
- 归纳如下句子的描述：

  ⑴ 哈尔滨是美丽的城市。

  ⑵ 北京是祖国的首都。

  ⑶ 集合是数学的基础。

  ⑷ 形式语言是很抽象的。

  ⑸ 教育走在社会发展的前面。

  ⑹ 中国进入**WTO**。

2.1 启示

- **6个句子的主体结构**

  <名词短语><动词短语><句号>

  <名词短语>={哈尔滨，北京，集合，形式语言，教育，中国}

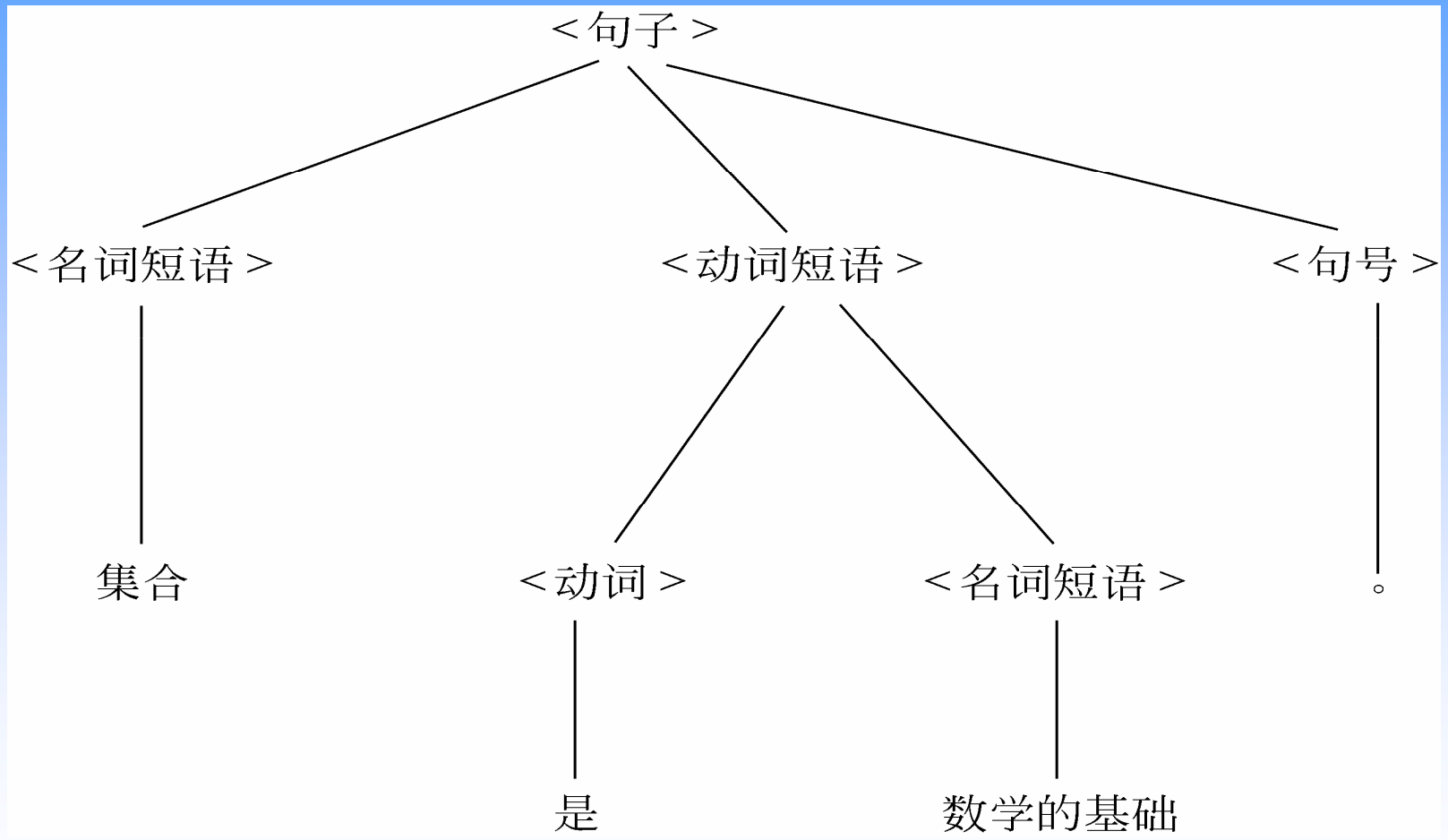
  <动词短语>={是美丽的城市，是祖国的首都，是数学的基础，是很抽象的，走在社会发展的前面，进入**WTO**}

  <句号>={。}

2.1 启示

- <动词短语>可以是<动词><形容词短语> 或者<动词><名词短语> 。
- <名词短语>={北京、哈尔滨、形式语言、中国、教育、集合、WTO、美丽的城市、祖国的首都、数学的基础、社会发展的前面}。
- <动词>={是、走在、进入}。
- <形容词短语>={很抽象的}。
- 把<名词短语><动词短语><句号>取名为<句子>。

2.1 启示



2.1 启示

- 表示成 α → β 形式

  <句子>→<名词短语><动词短语><句号>

  <动词短语>→<动词><形容词短语>

  <动词短语>→<动词><名词短语>

  <动词>→是

2.1 启示

<动词>→走在

<动词>→进入

<形容词短语>→很抽象的

<名词短语>→北京

<名词短语>→哈尔滨

<名词短语>→形式语言

2.1 启示

<名词短语>→中国

<名词短语>→教育

<名词短语>→集合

<名词短语>→ **WTO**

<名词短语>→美丽的城市

<名词短语>→祖国的首都

<名词短语>→数学的基础

<名词短语>→社会发展的前面

<句号>→。

2.1 启示

- 表示一个语言，需要4种东西

(1)形如<名词短语>的“符号”

它们表示相应语言结构中某个位置上可以出现的一些内容。每个“符号”对应的是一个集合，在该语言的一个具体句子中，句子的这个位置上能且仅能出现相应集合中的某个元素。所以，这种“符号”代表的是一个语法范畴。

(2) <句子>

所有的“规则”，都是为了说明<句子>的结构而存在，相当于说，定义的就是<句子>。

2.1 启示

(3)形如北京的“符号”

它们是所定义语言的合法句子中将出现的“符号”。仅仅表示自身，称为终极符号。

(4)所有的“规则”都呈 $\alpha \rightarrow \beta$ 的形式

在产生语言的句子中被使用，称这些“规则”为产生式。

2.2 形式定义

- **文法(grammar)**
- **G=(V，T，P，S)**
  - **V——为变量(variable)**的非空有穷集。∀**A∈V，A**叫做一个**语法变量(syntactic Variable)**，简称为变量，也可叫做**非终极符号(nonterminal)**。它表示一个**语法范畴(syntactic Category)**。所以，本文中有时候又称之为语法范畴。

2.2 形式定义

- **T——为终极符(terminal)的非空有穷集。∀a∈T，a叫做终极符。由于V中变量表示语法范畴，T中的字符是语言的句子中出现的字符，所以，有V∩T= Φ。**
- **S——S∈V，为文法G的开始符号(start symbol)。**

2.2 形式定义

- **P——为产生式(production)**的非空有穷集合。**P**中的元素均具有形式 $\alpha \to \beta$，被称为产生式，读作：α 定义为 β。其中 $\alpha \in (V \cup T)^+$，且 α 中至少有**V**中元素的一个出现。$\beta \in (V \cup T)^*$。α 称为产生式 $\alpha \to \beta$ 的**左部**，β 称为产生式 $\alpha \to \beta$ 的**右部**。产生式又叫做定义式或者语法规则。

2.2 形式定义

- **例 2-1** 以下四元组都是文法。

⑴ **({A}，{0，1}，{A→01，A→0A1，A→1A0}，A)**。

⑵ **({A}，{0，1}，{A→0，A→0A}，A)**。

⑶ **({A，B}，{0，1}，{A→01，A→0A1，A→1A0，B→AB，B→0}，A)**。

⑷ **({A，B}，{0，1}，{A→0，A→1，A→0A，A→1A }，A)**。

2.2 形式定义

⑸ ({S，A，B，C，D},{a，b，c，d,#}，{S→ABCD，S→abc#，A→aaA，AB→aabbB，BC→bbccC，cC→cccC，CD→ccd#，CD→d#，CD→#d}，S)。

⑹ ({S},{a，b}，{S→00S，S→11S，S→00，S→11}，S)。

2.2 形式定义

- 约定

  (1) 对一组有相同左部的产生式

  $\alpha \to \beta_1$，$\alpha \to \beta_2$，…，$\alpha \to \beta_n$

  可以简单地记为：

  $\alpha \to \beta_1 | \beta_2 | \ldots | \beta_n$

  读作：α 定义为 β_1，或者 β_2，…，或者 β_n。并且称它们为 α 产生式。β_1，β_2，…，β_n称为**候选式(candidate)**。

2.2 形式定义

⑵ 使用符号

- 英文字母表较为前面的大写字母，如**A**，**B**，**C**，…表示语法变量；
- 英文字母表较为前面的小写字母，如**a**，**b**，**c**，…表示终极符号；
- 英文字母表较为后面的大写字母，如**X**，**Y**，**Z**，…表示该符号是语法变量或者终极符号；
- 英文字母表较为后面的小写字母，如**x**，**y**，**z**，…表示由终极符号组成的行；
- 希腊字母α，β，γ…表示由语法变量和终极符号组成的行

2.2 形式定义

- **例 2-3** 四元组是否满足文法的要求。
  - **({A，B，C，E}，{a，b，c}，{S→ABC|abc，D→e|a，FB→c，A→A，E → abc| ε }，S)**
  - 4种修改

    (1) ({A，B，C，E，S，D，F}，{a，b，c，e}，{S→ABC|abc，D→e|a，FB→c，A→A，E → abc| ε }，S)。

    (2) ({A，B，C，E，S }，{a，b，c}，{S→ABC|abc，A→A，E → abc| ε }，S)。

    (3) ({A，B，C，E}，{a，b，c}，{ A→A，E → abc| ε }，A)。

    (4) ({A，B，C，E}，{a，b，c}，{ A→A，E → abc| ε }，E)。

2.2 形式定义

- **推导(derivation)**

  设**G=(V，T，P，S)**是一个文法，如果 $\alpha \to \beta \in \mathbf{P}$，$\gamma$，$\delta \in (\mathbf{V} \cup \mathbf{T})^*$，则称 $\gamma\ \alpha\ \delta$ 在**G**中直接推导出 $\gamma\ \beta\ \delta$。

  $$\gamma\ \alpha\ \delta \Rightarrow_G \gamma\ \beta\ \delta$$

  读作：$\gamma\ \alpha\ \delta$ 在文法G中直接推导出 $\gamma\ \beta\ \delta$。

  “直接推导”可以简称为**推导**(derivation)，也称推导为**派生**。

2.2 形式定义

- **归约(reduction)**
  - $\gamma\ \alpha\ \delta \Rightarrow_G \gamma\ \beta\ \delta$
  - 称 $\gamma\ \beta\ \delta$ 在文法**G**中直接归约成 $\gamma\ \alpha\ \delta$。在不特别强调归约的直接性时，“直接归约”可以简称为**归约**。

2.2 形式定义

1. 推导与归约表达的意思的异同。
2. 推导与归约和产生式不一样。所以，$\Rightarrow_G$和$\rightarrow$所表达的意思不一样。
3. 推导与归约是一一对应的。
4. 推导与归约的作用。

2.2 形式定义

$\Rightarrow_G$、$\Rightarrow_G^+$、$\Rightarrow_G^*$当成$(V \cup T)^*$上的二元关系。

(1) $\alpha \Rightarrow_G^n \beta$：表示 α 在**G**中经过**n**步推导出 β；β 在**G**中经过**n**步归约成 α。即，存在 $\alpha_1, \alpha_2, \ldots, \alpha_{n-1} \in (V \cup T)^*$使得 $\alpha \Rightarrow_G \alpha_1, \alpha_1 \Rightarrow_G \alpha_2, \ldots, \alpha_{n-1} \Rightarrow_G \beta$。

(2) 当**n=0**时，有 $\alpha = \beta$。即 $\alpha \Rightarrow_G^0 \alpha$。

(3) $\alpha \Rightarrow_G^+ \beta$：表示 α 在**G**中经过至少**1**步推导出 β；β 在**G**中经过至少**1**步归约成 α。

2.2 形式定义

(4) $\alpha \Rightarrow_G^* \beta$：表示 α 在**G**中经过若干步推导出 β；β 在**G**中经过若干步归约成 α。

分别用$\Rightarrow$、$\Rightarrow^+$、$\Rightarrow^*$、$\Rightarrow^n$代替

$\Rightarrow_G$、$\Rightarrow_G^+$、$\Rightarrow_G^*$、$\Rightarrow_G^n$。

2.2 形式定义

- **例 2-4 设G=({A}，{a}，{A→a|aA}，A)**

A ⇒ a<u>A</u> 使用产生式**A→aA**

⇒ aa<u>A</u> 使用产生式**A→aA**

⇒ aaa<u>A</u> 使用产生式**A→aA**

⇒ aaaa<u>A</u> 使用产生式**A→aA**

…

⇒a…a<u>A</u> 使用产生式**A→aA**

⇒a…aa 使用产生式**A→a⇒**

2.2 形式定义

$A \Rightarrow aA$	使用产生式$A \to aA$
$\Rightarrow aaA$	使用产生式$A \to aA$
$\Rightarrow aaaA$	使用产生式$A \to aA$
$\Rightarrow aaaaA$	使用产生式$A \to aA$
…	
$\Rightarrow a\ldots aA$	使用产生式$A \to aA$
$\Rightarrow a\ldots aaA$	使用产生式$A \to aA$

2.2 形式定义

$AAaaA\underline{A}A \Rightarrow A\underline{A}aaAaAA$ 使用产生式$A \to aA$

$\Rightarrow AaAaaAa\underline{A}A$ 使用产生式$A \to aA$

$\Rightarrow \underline{A}aAaaAaaA$ 使用产生式$A \to a$

$\Rightarrow aaAaaAaa\underline{A}$ 使用产生式$A \to a$

$\Rightarrow aa\underline{A}aaAaaa$ 使用产生式$A \to a$

$\Rightarrow aaa\underline{A}aaAaaa$ 使用产生式$A \to aA$

$\Rightarrow aaaaaa\underline{A}aaa$ 使用产生式$A \to a$

$\Rightarrow aaaaaaaaaa$ 使用产生式$A \to a$

2.2 形式定义

- **例 2-5** 设G=({S，A，B}，{0，1}，{S→A|AB，A→0|0A，B→1|11}，S)

对于n≥1，

$A \Rightarrow^n 0^n$	首先连续n-1次使用产生式； A→0A，最后使用产生式A→0；
$A \Rightarrow^n 0^nA$	连续n次使用产生式A→0A；
$B \Rightarrow 1$	使用产生式B→1；
$B \Rightarrow 11$	使用产生式B→11。

2.2 形式定义

语法范畴A代表的集合L(A)={0， 00， 000，0000，……}={0^n|n≥1};

语法范畴B代表的集合L(B)={1，11}

语法范畴S代表的集合L(S)=L(A)∪L(A)L(B)

={0， 00， 000， 0000， …}∪{0， 00， 000，0000，…}{1，11}

={0，00，000，0000，…}∪

∪{01，001，0001，00001，…}∪

∪{011，0011，00011，000011，…}

2.2 形式定义

例2-6 设$G=(\{A\},\{0,1\},\{A\to01,A\to0A1\},A)$，

$A \Rightarrow^{n} 0^nA1^n$ $n\geqslant0$

$0^nA1^n \Rightarrow 0^{n+1}A1^{n+1}$ $n\geqslant0$

$0^nA1^n \Rightarrow 0^{n+1}1^{n+1}$ $n\geqslant0$

$0^nA1^n \Rightarrow^{i} 0^{n+i}A1^{n+i}$ $n\geqslant0,\ i\geqslant0$

$0^nA1^n \Rightarrow^{i} 0^{n+i}1^{n+I}$ $n\geqslant0,\ i\geqslant0$

$0^nA1^n \Rightarrow^{*} 0^mA1^m$ $n\geqslant0,\ m\geqslant n$

$0^nA1^n \Rightarrow^{+} 0^m1^m$ $n\geqslant0,\ m\geqslant n+1$

$0^nA1^n \Rightarrow^{+} 0^mA1^m$ $n\geqslant0,\ m\geqslant n+1$

$0^nA1^n \Rightarrow^{+} 0^m1^m$ $n\geqslant0,\ m\geqslant n+1$

2.2 形式定义

- 几点结论
  - 对任意的$x \in \Sigma^+$，我们要使语法范畴D代表的集合为$\{x^n | n \geqslant 0\}$，可用产生式组$\{D \rightarrow \varepsilon | xD\}$来实现。
  - 对任意的x，$y \in \Sigma^+$，我们要使语法范畴D代表的集合为$\{x^n y^n | n \geqslant 1\}$，可用产生式组$\{D \rightarrow xy | xDy\}$来实现。
  - 对任意的x，$y \in \Sigma^+$，我们要使语法范畴D代表的集合为$\{x^n y^n | n \geqslant 0\}$，可用产生式组$\{D \rightarrow \varepsilon | xDy\}$来实现。

2.2 形式定义

- **语言(language)**

  $L(G)=\{w \mid w \in T^{*}$ 且 $S \Rightarrow^{*} w\}$

- **句子(sentence)**

  $\forall w \in L(G)$，w称为G产生的一个**句子**。

- **句型(sentential form)**

  $G=(V, T, P, S)$，对于 $\forall \alpha \in (V \cup T)^{*}$，如果 $S \Rightarrow^{*} \alpha$，则称 α 是G产生的一个**句型**。

2.2 形式定义

- 句子**w**是从**S**开始，在**G**中可以推导出来的终极符号行，它**不含**语法变量。
- 句型 α 是从**S**开始，在**G**中可以推导出来的符号行，它**可能含有**语法变量。
- **例 2-7** 给定文法**G=({S，A，B，C，D},{a，b，c，d,#}，{S→ABCD|abc#，A→aaA，AB→aabbB，BC→bbccC，cC→cccC，CD→ccd#，CD→d#，CD→#d}，S)**

2.2 形式定义

S ⇒ ABCD	使用产生式**S→ABCD**
⇒aaABCD	使用产生式**A→aaA**
⇒aaaaABCD	使用产生式**A→aaA**
⇒aaaaaabbBCD	使用产生式**AB→aabbB**
⇒aaaaaabbbbccCD	使用产生式**BC→bbccC**
⇒aaaaaabbbbccccCD	使用产生式**cC→cccC**
⇒aaaaaabbbbcccc#d	使用产生式**CD→#d**

2.2 形式定义

S ⇒ ABCD	使用产生式**S→ABCD**
⇒AbbccCD	使用产生式**BC→bbccC**
⇒aaAbbccCD	使用产生式**A→aaA**
⇒aaAbbccccd#	使用产生式**CD→ccd#**
⇒aaaaAbbccccd#	使用产生式**A→aaA**
⇒aaaaaaAbbccccd#	使用产生式**A→aaA**
⇒aaaaaaaaAbbccccd#	使用产生式**A→aaA**

2.2 形式定义

- 例 2-8 构造产生标识符的文法

G=({<标识符>，<大写字母>，<小写字母>，<阿拉伯数字>}，{0，1，…，9，A，B，C，…，Z，a，b，c，…，z}，P, <标识符>)

P={ <标识符>→<大写字母>|<小写字母>，

<标识符>→<标识符><大写字母>|<标识符><小写字母>|<标识符><阿拉伯数字>，

<大写字母>→A|B|C|D|E|F|G|H|I|J|K|L|M|N|O，

<大写字母>→ P|Q|R|S|T|U|V|W|X|Y|Z，

<小写字母>→a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z ，

<阿拉伯数字>→0|1|2|3|4|5|6|7|8|9 }

2.2 形式定义

G′ =({<标识符>，<头>，<尾>}，{0，1，2，…，9，A，B，C，…，Z，a，b，c，…，z}，P′，<标识符>)

P′ ={<标识符>→<头><尾>，

<头>→A|B|C|D|E|F|G|H|I|J|K|L|M|N

<头>→ O|P|Q|R|S|T|U|V|W|X|Y|Z，

<头>→a|b|c|d|e|f|g|h|i|j|k|l|m|n

<头>→ o|p|q|r|s|t|u|v|w|x|y|z ，

2.2 形式定义

<尾>→ ε |0<尾>|1<尾>|2<尾>|3<尾>|4<尾>|5<尾>,
<尾>→ 6<尾>|7<尾>|8<尾>|9<尾> ，
<尾>→A<尾>|B<尾>|C<尾>|D<尾>|E<尾>|F<尾>,
<尾>→ G<尾>|H<尾>|I<尾>|J<尾>|K<尾> ，
<尾>→L<尾>|M<尾>|N<尾>|O<尾>|P<尾>|Q<尾>,
<尾>→ R<尾>|S<尾>|T<尾>|U<尾>|V<尾> ，
<尾>→ W<尾>|X<尾>|Y<尾>|Z<尾>|a<尾>|b<尾>,
<尾>→ c<尾>|d<尾>|e<尾>|f<尾>|g<尾> ，
<尾>→h<尾>|i<尾>|j<尾>|k<尾>|l<尾>|m<尾>,
<尾>→ n<尾>|o<尾>|p<尾>|q<尾>|r<尾> ，
<尾>→s<尾>|t<尾>|u<尾>|v<尾>|w<尾>|x<尾>
<尾>→ y<尾>|z<尾> }

2.2 形式定义

<标识符> ⇒<标识符><阿拉伯数字>

⇒ <标识符>**3**

⇒ <标识符><阿拉伯数字>**3**

⇒ <标识符>**23**

⇒ <标识符><小写字母>**23**

⇒ <标识符>**n23**

⇒ <标识符><阿拉伯数字>**n23**

⇒ <标识符>**8n23**

⇒ <标识符><小写字母>**8n23**

⇒ <标识符>**d8n23**

⇒ <小写字母>**d8n23**

⇒ **id8n23**

2.2 形式定义

标识符> ⇒<头><尾>

⇒**i**<尾>

⇒**id**<尾>

⇒**id8**<尾>

⇒**id8n**<尾>

⇒**id8n2**<尾>

⇒**id823**<尾>

⇒**id8n23**

2.2 形式定义

- 使用符号使文法更简洁

G′′=({<标识符>},{b，a，d},{<标识符>→b|a|<标识符>b|<标识符>a|<标识符>d}, <标识符>)

G′′′=({L}，{b，a，d}，{L→b|a|Lb|La|Ld}, L)

G′′′′=({L，H，T}，{ b，a，d }，{L→HT，H→b|a，T→ ε |bT|aT|dT}, L)

2.3 文法的构造

- **例2-9** 构造文法G，使L(G)={0，1，00，11}
  - 将文法的开始符号定义为这4个句子。
    - G_1=({S}，{0，1}，{S→0，S→1，S→00，S→11}，S)
  - 先用变量A表示0，用变量B表示1。
    - G_2=({S，A，B}，{0，1}，{S→A，S→B，S→AA，S→BB，A→0，B→1}，S)
  - 基于G_2，考虑“规范性”问题。
    - G_3=({S，A，B}，{0，1}，{S→0，S→1，S→0A，S→1B，A→0，B→1}，S)

2.3 文法的构造

- 可以在V、T中增加一些元素，以获得“不同的”文法。
  - G_4=({S，A，B，C}，{0，1，2}，{S→A，S→B，S→AA，S→BB，A→0，B→1}，S)
  - G_5=({S，A，B，C}，{0，1，2}，{S→A，S→B，S→AA，S→BB，A→0，B→1，CACS→21，C→11，C→2}，S)

$L(G_1)= L(G_2)= L(G_3)= L(G_4)= L(G_5)$

2.3 文法的构造

- **等价(equivalence)**
  - 设有两个文法G_1和G_2，如果$L(G_1)= L(G_2)$，则称G_1与G_2等价。
- 约定
  - 对一个文法，只列出该文法的所有产生式，且所列第一个产生式的左部是该文法的开始符号。

2.3 文法的构造

G_1：$S \to 0|1|00|11$

G_2：$S \to A|B|AA|BB$，$A \to 0$，$B \to 1$

G_3：$S \to 0|1|0A|1B$，$A \to 0$，$B \to 1$

G_4：$S \to A|B|AA|BB$，$A \to 0$，$B \to 1$

G_5：$S \to A|B|BB$，$A \to 0$，$B \to 1$，$CACS \to 21$，$C \to 11$，$C \to 2$

2.3 文法的构造

- 例 2-10
- $L=\{0^n|n\geqslant 1\}$

  G_6：$S\rightarrow 0|0S$
- $L=\{0^n|n\geqslant 0\}$

  G_7：$S\rightarrow \varepsilon|0S$
- $L=\{0^{2n}1^{3n}|n\geqslant 0\}$

  G_8：$S\rightarrow \varepsilon|00S111$

2.3 文法的构造

- **例2-11** 构造文法G_9，使$L(G_9)=\{w|w\in\{a，b，…，z\}^+\}$。

  G_9：$S\rightarrow A|AS$

  $A\rightarrow a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z$

- 用$S\rightarrow A|AS$生成 A^n。
- 不可以用$A\rightarrow a|b|c|…|z$表示。
  $A\rightarrow a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z$
- 不可以用$A\rightarrow a^8$表示$A\rightarrow aaaaaaaa$。
- 不能用$A\rightarrow a^n$　表示A可以产生任意多个a。

2.3 文法的构造

- **例2-12** 构造文法G_{10}，使 $L(G_{10})=\{ww^T|w\in\{0，1，2，3\}^+\}$。

文法

$S\rightarrow HE$

$H\rightarrow 0|1|2|3|0H|1H|2H|3H$

$E\rightarrow 0|1|2|3|E0|E1|E2|E3$

难以生成$L(G_{10})$。

2.3 文法的构造

- **$\{ww^T|w\in\{0，1，2，3\}^+\}$**的句子的特点

  设**$w=a_1a_2\ldots a_n$**，从而有**$w^T=a_n\ldots a_2\ a_1$**，故**w $w^T=a_1a_2\ldots a_na_n\ldots a_2\ a_1$**

  满足**f(w w^T，i)=f(w w^T，|w w^T|-i+1)**。

  递归地定义**L**

  - ⑴ 对**$\forall a\in\{0，1，2，3\}$**，**$aa\in L$**；
  - ⑵ 如果**$x\in L$**，则对**$\forall a\in\{0，1，2，3\}$**，**$axa\in L$**；
  - ⑶ **L**中不含不满足**(1)**、**(2)**任何其他的串。

2.3 文法的构造

根据递归定义中的第一条，有如下产生式组：

S→00 | 11 | 22 | 33

再根据递归定义第二条，又可得到如下产生式组：

S→0S0 | 1S1 | 2S2 | 3S3

从而，

G_{10}：S→00 | 11 | 22 | 33 | 0S0 | 1S1 | 2S2 | 3S3

2.3 文法的构造

- **例2-13** 构造文法G_{12}，使$L(G_{12})$={w|w是我们习惯的十进制有理数}。

G_{12}：$S \rightarrow R \mid +R \mid -R$

$R \rightarrow N \mid B$

$B \rightarrow N.D$

$N \rightarrow 0 \mid AM$

$D \rightarrow 0 \mid MA$

$A \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$M \rightarrow \varepsilon \mid 0M \mid 1M \mid 2M \mid 3M \mid 4M \mid 5M \mid 6M \mid 7M \mid 8M \mid 9M$

2.3 文法的构造

- **例2-14** 构造产生算术表达式的文法：

⑴ 基础：常数是算术表达式，变量是算术表达式；

⑵ 归纳：如果E_1、E_2是表达式，则 $+E_1$、$-E_1$、E_1+E_2、E_1-E_2、E_1*E_2、E_1/E_2、E_1**E_2、$Fun(E_1)$是算术表达式。其中Fun为函数名。

⑶ 只有满足(1)和(2)的才是算术表达式。

G_{13}：$E \rightarrow id|c|+E|-E|E+E|E-E|E*E|E/E|E**E|Fun(E)$

2.3 文法的构造

- **例2-15** 构造产生语言$\{a^nb^nc^n|n\geqslant 1\}$的文法。

文法$G=(\{S_1\},\{a,b\},\{S_1\to ab\mid aS_1b\},S_1)$

产生的语言$\{a^nb^n\mid n\geqslant 1\}$形式上看起来与语言$\{a^nb^nc^n\mid n\geqslant 1\}$比较接近。

$G=(\{S_2\},\{c\},\{S_2\to c\mid cS_2\},S_2)$产生的语言是$\{c^n|n\geqslant 1\}$。

$\{a^nb^nc^n\mid n\geqslant 1\}\neq\{a^nb^n\mid n\geqslant 1\}\{c^n\mid n\geqslant 1\}$

2.3 文法的构造

文法

$\mathbf{S \rightarrow S_1 S_2}$

$\mathbf{S_1 \rightarrow ab \mid aS_1b}$

$\mathbf{S_2 \rightarrow c \mid cS_2}$

不能产生语言

$\mathbf{\{a^n b^n c^n \mid n \geqslant 1\}}$

而是产生语言

$\{a^n b^n \mid n \geqslant 1\}\{c^n \mid n \geqslant 1\}$

2.3 文法的构造

文法

G：S→abc | aSbc

产生的语言为：

$\{a^n(bc)^n \mid n \geqslant 1\}$

焦点：交换**b**和**c**的位置。

2.3 文法的构造

G_{14}：$S \to aBC \mid aSBC$，

$CB \to BC$

$aB \to ab$

$bB \to bb$

$bC \to bc$

$cC \to cc$

G_{14}'：$S \to abc \mid aSBc$，

$bB \to bb$

$cB \to Bc$

文法的乔姆斯基体系

- 文法G=(V，T，P，S)
- G叫做0型文法(type 0 grammar)，也叫做短语结构文法(phrase structure grammar, PSG)。
- L(G)叫做0型语言。也可以叫做短语结构语言(PSL)、递归可枚举集(recursively enumerable，r.e.)。

文法的乔姆斯基体系

- G是0型文法。
- 如果对于$\forall \alpha \to \beta \in P$，均有$|\beta| \geqslant |\alpha|$成立，则称**G为1型文法(type 1 grammar)**，或上下文有关文法**(context sensitive grammar,CSG)**。
- **L(G)**叫做**1型语言(type 1 language)**或者上下文有关语言**(context sensitive language ,CSL)**。

文法的乔姆斯基体系

- **G是1型文法**
- 如果对于$\forall \alpha \rightarrow \beta \in P$，均有$|\beta| \geqslant |\alpha|$，并且$\alpha \in V$成立，则称**G为2型文法(type 2 grammar)，或上下文无关文法(context free grammar,CFG)。**
- **L(G)叫做2型语言(type 2 language)或者上下文无关语言(context free language ,CFL)。**

文法的乔姆斯基体系

- **G是2型文法**
- 如果对于$\forall \alpha \to \beta \in P$，$\alpha \to \beta$均具有形式

$$A \to w$$

$$A \to wB$$

其中A，$B \in V$，$w \in T^+$。则称G为**3型文法(type 3 grammar)**，也可称为**正则文法(regular grammar ,RG)**或者正规文法。$L(G)$叫做**3型语言(type 3 language)**，也可称为**正则语言**或者**正规语言(regular language ,RL)**。

文法的乔姆斯基体系

⑴ 如果一个文法**G**是**RG**，则它也是**CFG**、**CSG**和短语结构文法。反之不一定成立。

⑵ 如果一个文法**G**是**CFG**，则它也是**CSG**和短语结构文法。反之不一定成立。

⑶ 如果一个文法**G**是**CSG**，则它也是短语结构文法。反之不一定成立。

相应地：

⑷ **RL**也是**CFL**、**CSL**和短语结构语言。反之不一定成立。

文法的乔姆斯基体系

⑸ **CFL**也是**CSL**和短语结构语言。反之不一定成立。

⑹ **CSL**也是短语结构语言。反之不一定成立

⑺ 当文法G是CFG时，L(G)却可以是RL。

⑻ 当文法G是CSG时，L(G)可以是RL、CSL。

⑼ 当文法G是短语结构文法时，L(G)可以是RL、CSL和CSL。

文法的乔姆斯基体系

定理 2-1 **L是RL的充要条件是存在一个文法，该文法产生语言L，并且它的产生式要么是形如：A→a的产生式，要么是形如A→aB的产生式。其中A、B为语法变量，a为终极符号。**

- 证明 ：
  - 充分性：设有G′，L(G′)=L，且G′的产生式的形式满足定理要求。这种文法就是RG。所以，G′产生的语言就是RL，故L是RL。

文法的乔姆斯基体系

- 必要性

  构造：用产生式组：

  $\mathbf{A \rightarrow a_1 A_1}$

  $\mathbf{A_1 \rightarrow a_2 A_2}$

  …

  $\mathbf{A_{n-1} \rightarrow a_n}$

  代替产生式

  $\mathbf{A \rightarrow a_1 a_2 \ldots a_n}$

文法的乔姆斯基体系

- 用产生式组

  $A \rightarrow a_1 A_1$

  $A_1 \rightarrow a_2 A_2$

  …

  $A_{n-1} \rightarrow a_n B$

  代替产生式

  $A \rightarrow a_1 a_2 \ldots a_n B$

文法的乔姆斯基体系

- 证明$L(G')=L(G)$。

  施归纳于推导的步数，证明一个更一般的结论：对于$\forall A \in V$，$A \Rightarrow_G^+ x \Leftrightarrow A \Rightarrow_{G'}^+ x$。因为$S \in V$，所以结论自然对S成立。

文法的乔姆斯基体系

- 几点注意事项

⑴ 我们是按照**RG**的一般定义来构造一个与之等价的文法的，这与读者以前熟悉的根据一个具体的对象构造另一个对象是不同的。在这里，可以使用的是非常一般的条件——一个一般模型。这也是这类问题的证明所要求的。而且在本书的后面，将会有更多这样的情况。

文法的乔姆斯基体系

(2) 为了证明一个特殊的结论，可以通过证明一个更为一般的结论来完成。这从表面上好像是增加了我们要证明的内容，但实际上它会使我们能够更好地使用归纳假设，以便顺利地获得我们所需要的结论。

文法的乔姆斯基体系

(3) 施归纳于推导的步数是证明本书中不少问题的较为有效的途径。有时我们还会对字符串的长度施归纳。

本证明的主要部分含两个方面，首先是构造，然后对构造的正确性进行证明。这种等价性证明的思路是非常重要的。

文法的乔姆斯基体系

- **线性文法(liner grammar)**
  - 设**G=(V，T，P，S)**，如果对于$\forall \alpha \rightarrow \beta \in P$，$\alpha \rightarrow \beta$ 均具有如下形式：
  - **$A \rightarrow w$**
  - **$A \rightarrow wBx$**
  - 其中**$A，B \in V$，$w，x \in T^*$**，则称**G为线性文法**。
- **线性语言(liner language)**
  - **L(G)**叫做**线性语言**

文法的乔姆斯基体系

- **右线性文法(right liner grammar)**
  - 设**G=(V，T，P，S)**，如果对于$\forall \alpha \to \beta \in P$，$\alpha \to \beta$ 均具有如下形式：
  - **$A \to w$**
  - **$A \to wB$**
  - 其中**$A，B \in V$，$w，x \in T^*$**，则称**G**为**线性文法**。
- **右线性语言(right liner language)**
  - **L(G)**叫做右**线性语言**。

文法的乔姆斯基体系

- **左线性文法(left liner grammar)**
  - 设**G=(V，T，P，S)**，如果对于$\forall \alpha \to \beta \in P$，$\alpha \to \beta$ 均具有如下形式：
  - **$A \to w$**
  - **$A \to Bw$**
  - 其中**$A，B \in V$，$w，x \in T^*$**，则称**G**为**线性文法**。
- **左线性语言(left liner language)**
  - **L(G)**叫做右**线性语言**。

文法的乔姆斯基体系

定理2-2 L是一个左线性语言的充要条件是存在文法G，G中的产生式要么是形如：$A \rightarrow a$的产生式，要么是形如$A \rightarrow Ba$的产生式，使得L(G)=L。其中A、B为语法变量，a为终极符号。

文法的乔姆斯基体系

定理2-3 左线性文法与右线性文法等价。

- 按照定理2-1的证明经验，要想证明本定理，需要完成如下工作：
  - 对任意右线性文法G，我们能够构造出对应的左线性文法G′，使得L(G′)=L(G)；
  - 对任意左线性文法G，我们能够构造出对应的右线性文法G′，使得L(G′)=L(G)。

文法的乔姆斯基体系

- **例 2-17** 语言**{0123456}**的左线性文法和右线性文法的构造。
- 右线性文法

G_r：$S_r \rightarrow 0A_r$

$A_r \rightarrow 1B_r$

$B_r \rightarrow 2C_r$

$C_r \rightarrow 3D_r$

$D_r \rightarrow 4E_r$

$E_r \rightarrow 5F_r$

$F_r \rightarrow 6$

文法的乔姆斯基体系

- **0123456**在文法G_r中的推导

$S_r \Rightarrow 0A_r$ 使用产生式$S_r \to 0A_r$

$\Rightarrow 01B_r$ 使用产生式$A_r \to 1B_r$

$\Rightarrow 012C_r$ 使用产生式$B_r \to 2C_r$

$\Rightarrow 0123D_r$ 使用产生式$C_r \to 3D_r$

$\Rightarrow 01234E_r$ 使用产生式$D_r \to 4E_r$

$\Rightarrow 012345F_r$ 使用产生式$E_r \to 5F_r$

$\Rightarrow 0123456$ 使用产生式$F_r \to 6$

文法的乔姆斯基体系

- 左线性文法

G_l：$S_l \rightarrow A_l 6$

$A_l \rightarrow B_l 5$

$B_l \rightarrow C_l 4$

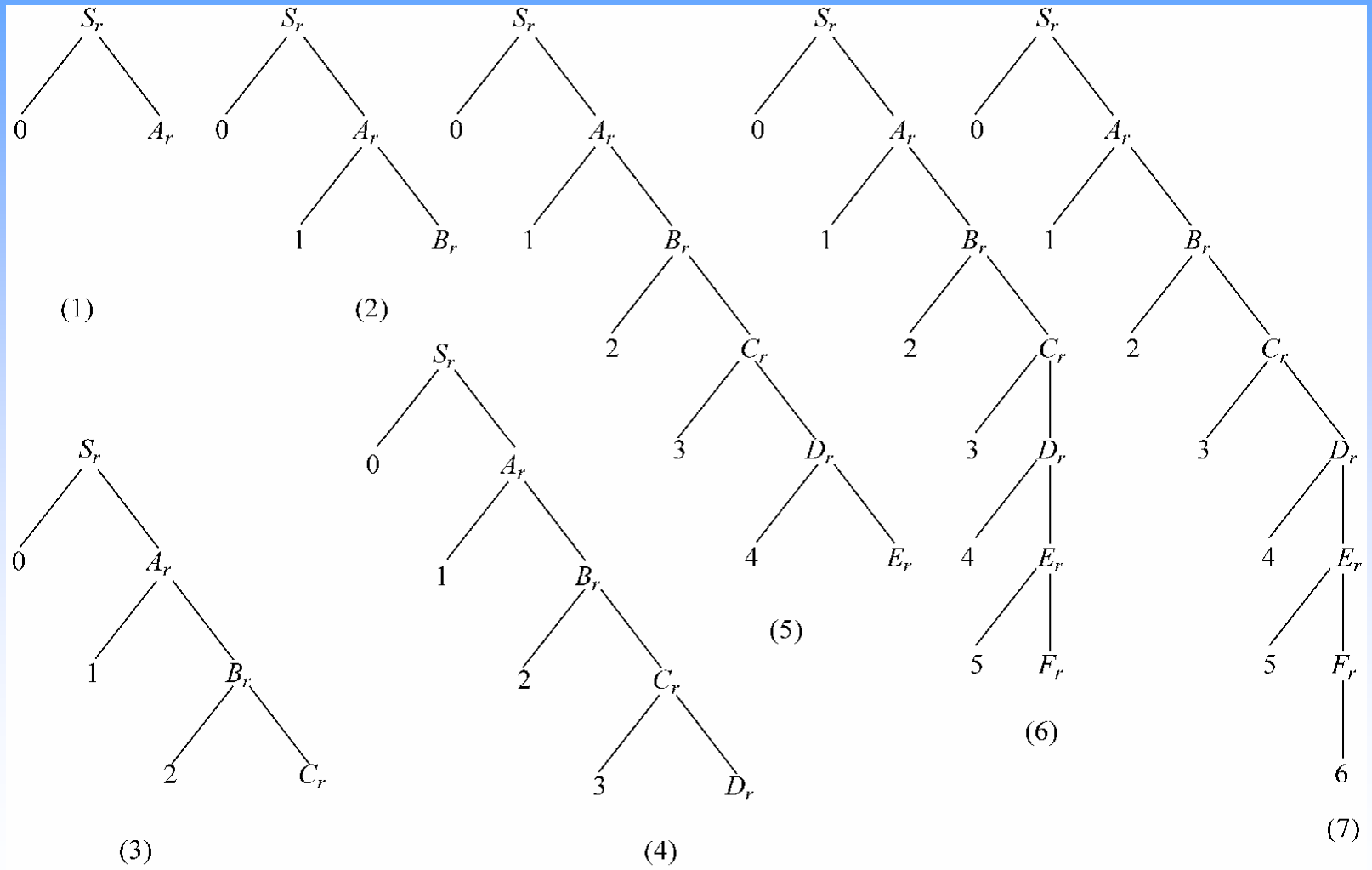
$C_l \rightarrow D_l 3$

$D_l \rightarrow E_l 2$

$E_l \rightarrow F_l 1$

$F_l \rightarrow 0$

文法的乔姆斯基体系



文法的乔姆斯基体系

- **0123456**在文法G_l中的推导

$S_l \Rightarrow A_l 6$　　使用产生式$S_l \rightarrow A_l 6$

$\Rightarrow B_l 56$　　使用产生式$A_l \rightarrow B_l 5$

$\Rightarrow C_l 456$　　使用产生式$B_l \rightarrow C_l 4$

$\Rightarrow D_l 3456$　　使用产生式$C_l \rightarrow D_l 3$

$\Rightarrow E_l 23456$　　使用产生式$D_l \rightarrow E_l 2$

$\Rightarrow F_l 1234456$　　使用产生式$E_l \rightarrow F_l 1$

$\Rightarrow 0123456$　　使用产生式$F_l \rightarrow 0$

文法的乔姆斯基体系

- **0123456被归约成文法G_l的开始符号S**

$\underline{0}$123456

$\Leftarrow \underline{F_l}$1234456 使用产生式$F_l \rightarrow 0$

$\Leftarrow \underline{E_l}$23456 使用产生式$E_l \rightarrow F_l 1$

$\Leftarrow \underline{D_l}$3456 使用产生式$D_l \rightarrow E_l 2$

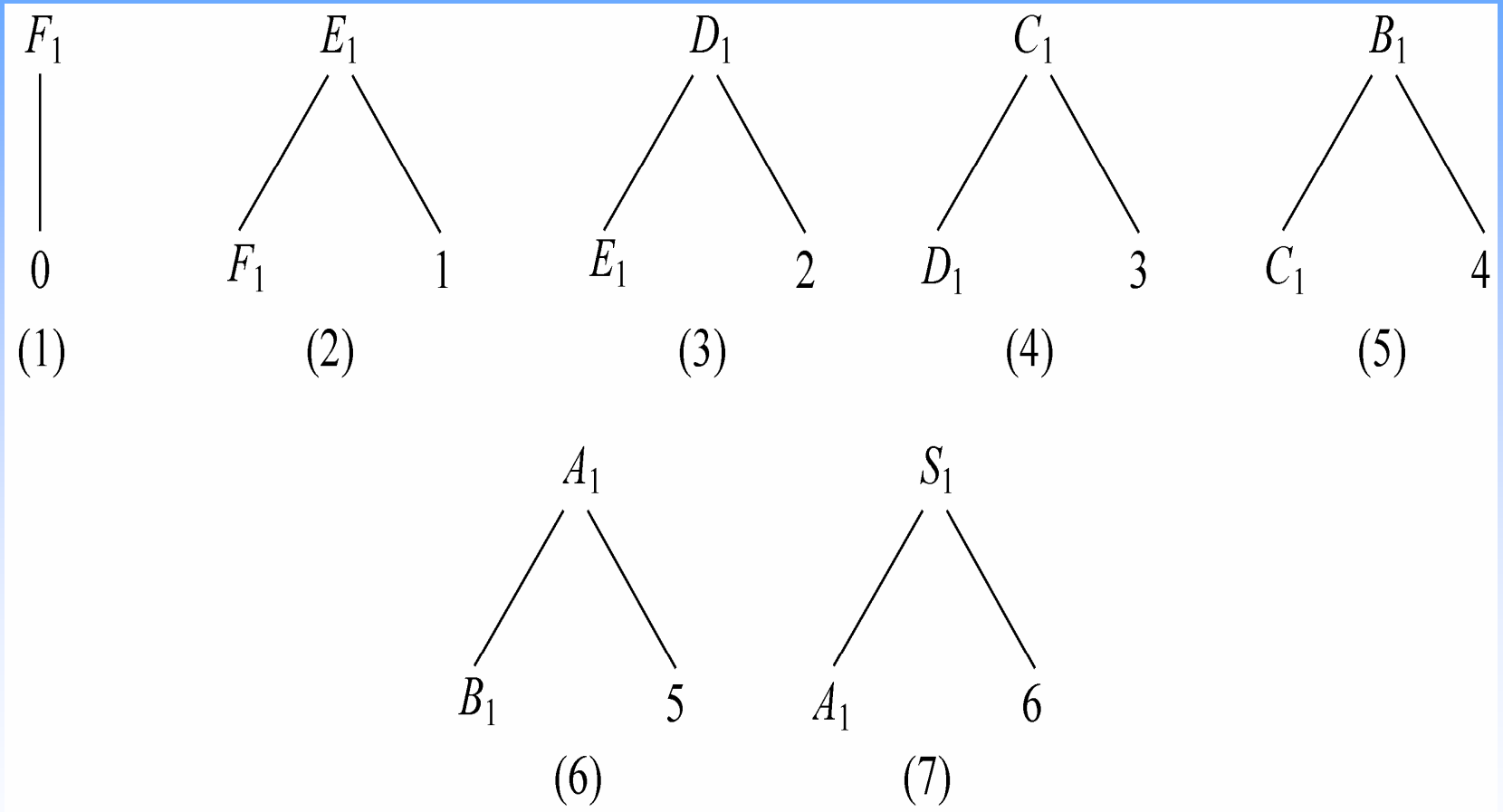
$\Leftarrow \underline{C_l}$456 使用产生式$C_l \rightarrow D_l 3$

$\Leftarrow \underline{B_l}$56 使用产生式$B_l \rightarrow C_l 4$

$\Leftarrow \underline{A_l}$6 使用产生式$A_l \rightarrow B_l 5$

$\Leftarrow S_l$ 使用产生式$S_l \rightarrow A_l 6$

文法的乔姆斯基体系



文法的乔姆斯基体系

定理 2-4 左线性文法的产生式与右线性文法的产生式混用所得到的文法不是**RG**。

证明：设有文法

G_{15}：S→0A

A→S1|1

不难看出，**$L(G_{15})=\{0^n1^n|n\geqslant 1\}$**。我们构造不出**RG G**，使得**$L(G)= L(G_{15})=\{0^n1^n|n\geqslant 1\}$**。因为**$L(G_{15})=\{0^n1^n|n\geqslant 1\}$**不是**RL**。所以，**$G_{15}$**不是**RG**。有关该语言不是**RL**的证明我们将留到研究**RL**的性质时去完成。

2.5 空语句

- 形如**A→** ε 的产生式叫做空产生式，也可叫做 ε 产生式。
- 在**RG、CFG、CSG**中，都不能含有空产生式。所以，任何**CSL**中都不含有空语句 ε 。从而**CFL**和**RL**中都不能含空语句 ε 。
- 空语句 ε 在一个语言中的存在并不影响该语言的有穷描述的存在性，甚至除了为生成空语句 ε 外，空产生式可以不被用于语言中其他任何句子的推导中。

2.5 空语句

- 允许CSL、CFL、RL包含空语句 ε 后，还会给我们进行问题的处理提供一些方便。
- 允许在**RG、CFG、CSG**中含有空产生式
- 允许**CSL、CFL、RL**包含空语句 ε 。

2.5 空语句

定理2-5 设**G=(V，T，P，S)**为一文法，则存在与**G**同类型的文法**G′=(V′，T，P′，S′)**，使得**L(G)=L(G′)**，但**G′**的开始符号**S′**不出现在**G′**的任何产生式的右部。

证明：

构造

当文法G=(V，T，P，S)的开始符号S不出现在P中的任何产生式的右部时，G就是所求

$G'=(V\cup\{S'\},T,P',S')$

$\mathbf{P'=P\cup\{S'\to\alpha\mid S\to\alpha\in P\}}$

2.5 空语句

- 证明$L(G)\subseteq L(G')$

  对任意的$x\in L(G')$，由文法产生的语言的定义知，在G'中存在如下推导：

  $S' \Rightarrow \alpha$　　使用产生式$S' \to \alpha$

  $\Rightarrow^{*} x$　　使用P'中的除$S' \to \alpha$以外的其他产生式。

  在推导$\alpha \Rightarrow^{*} x$中使用的产生式都是P中的产生式。因此，推导$\alpha \Rightarrow^{*} x$在G中仍然成立。

2.5 空语句

由**P′** 的定义知，必有**S**→α ∈**P**

所以，

S⇒α　　使用**P**中的产生式**S**→α

⇒*x　　使用**P**中的产生式

故，

L(G′)⊆**L(G)**

2.5 空语句

设G中存在如下推导：

$S \Rightarrow \alpha$　　使用P中的产生式$S \to \alpha$

$\Rightarrow^* x$　　使用P中的产生式

由$P' = P \cup \{S' \to \alpha \mid S \to \alpha \in P\}$，知道$G'$中，

$S' \Rightarrow \alpha$　　使用产生式$S' \to \alpha$

$\Rightarrow^* x$　　使用P'所包含的P中的其他产生式。

故，$L(G) \subseteq L(G')$。

2.5 空语句

设G=(V，T，P，S)是一个文法，如果S不出现在G的任何产生式的右部，则：

⑴ 如果G是CSG，则仍然称G=(V，T，P∪{S→ε}，S)为CSG；G产生的语言仍然称为CSL。

⑵ 如果G是CFG，则仍然称G=(V，T，P∪{S→ε}，S)为CFG；G产生的语言仍然称为CFL。

⑶ 如果G是RG，则仍然称G=(V，T，P∪{S→ε}，S)为RG。G产生的语言仍然称为RL。

2.5 空语句

定理2-6 下列命题成立：

⑴ 如果L是CSL，则L∪{ε}仍然是CSL。

⑵ 如果L是CFL，则L∪{ε}仍然是CFL。

⑶ 如果L是RL，则L∪{ε}仍然是RL。

2.5 空语句

证明：对第1个命题：设L是CSL，则存在一个CSG G=(V，T，P，S)，使得L(G)=L。由定理2-5-1，我们不妨假设S不出现在G的任何产生式的右部。

取：

$G' =(V，T，P \cup \{S \rightarrow \varepsilon\}，S)$

由于S不出现在G的任何产生式的右部，所以，$S \rightarrow \varepsilon$ 不可能出现在任何长度不为0的句子的推导中。

2.5 空语句

易证：

L(G′)=L(G)∪{ε}。

由于**G′**是**CSG**，所以，**L(G)∪{ε}**是**CSL**。

同理可证第2和第3个命题。

2.5 空语句

定理2-7 下列命题成立

⑴ 如果L是CSL，则L-{ ε }仍然是CSL。

⑵ 如果L是CFL，则L-{ ε }仍然是CFL。

⑶ 如果L是RL，则L-{ ε }仍然是RL。

2.5 空语句

证明：对第1个命题：设L是CSL，则存在一个CSG G=(V，T，P，S)，使得L(G)=L。如果 ε ∉L，则L-{ ε }=L，所以， L-{ ε }是CSL。

如果 ε ∈L，由定理2-5-1，我们不妨假设S不出现在G的任何产生式的右部。取：

G′ =(V，T，P-{S→ ε }，S)

2.5 空语句

由于S不出现在G的任何产生式的右部，所以，如果L(G)中存在长度非0的句子，$S\rightarrow\varepsilon$ 不可能出现在它们的推导中。也就是说，将$S\rightarrow\varepsilon$ 从G中去掉后，不会影响L(G)中任何长度非0的句子的推导。容易证明：

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

由于G′是CSG，所以，$L(G)-\{\varepsilon\}$是CSL。

同理可证其他两个命题。

2.5 空语句

- 对于任意文法**G=(V，T，P，S)**，对于**G**中的其他变量**A**，出现形如**A→ε**的产生式是不会改变文法产生的语言的类型的，而且这样一来，对我们进行文法的构造等工作还提供了很多方便。所以，我们约定：对于**G**中的任何变量**A**，在需要的时候，可以出现形如**A→ε**的产生式。

2.6 小结

本章讨论了语言的文法描述。首先介绍文法的基本定义和推导、归约、文法定义的语言、句子、句型、文法的等价等重要概念。讨论了如何根据语言的特点、通过用语法变量去表示适当的集合（语法范畴）的方法进行文法构造，并按照乔姆斯基体系，将文法划分成**PSG**、**CSG**、**CFG**、**RG**等**4**类。在这些文法中，线性文法是一类重要的文法。

2.6 小结

(1) 文法**G=(V，T，P，S)**。任意**A∈V**表示集合**L(A)={w | w∈T*且A ⇒* w}**。

(2) 推导与归约。文法中的推导是根据文法的产生式进行的。如果 $\alpha \to \beta \in \mathbf{P}$，$\gamma$，$\delta \in (\mathbf{V} \cup \mathbf{T})^*$，则称 $\gamma\ \alpha\ \delta$ 在**G**中直接推导出 $\gamma\ \beta\ \delta$：$\gamma\ \alpha\ \delta \Rightarrow_G \gamma\ \beta\ \delta$；也称 $\gamma\ \beta\ \delta$ 在文法**G**中直接归约成 $\gamma\ \alpha\ \delta$。

2.6 小结

⑶ 语言、句子和句型。文法G产生的语言$L(G)=\{w \mid w\in T^{*}$且$S \Rightarrow^{*} w\}$，$w\in L(G)$为句子。一般地，由开始符号推出来的任意符号行叫做G的句型。

⑷ 一个语言可以被多个文法产生，产生相同语言的文法被称是等价的。

2.6 小结

⑸ 右线性文法的产生式都可以是形如**A→a**和**A→aB**的产生式。左线性文法的产生式都可以是形如**A→a**和**A→Ba**的产生式。左线性文法与右线性文法是等价的。然而，左线性文法的产生式与右线性文法的产生式混用所得到的文法不是正则文法。

第3章 有穷状态自动机

- 主要内容
- 确定的有穷状态自动机(DFA)
  - 作为对实际问题的抽象、直观物理模型、形式定义，DFA接受的句子、语言，状态转移图。
- 不确定的有穷状态自动机(NFA)
  - 定义；
  - NFA与DFA的等价性；

第3章 有穷状态自动机

- 带空移动的有穷状态自动机(ε -NFA)
  - 定义。
  - ε -NFA与DFA的等价性。
- FA是正则语言的识别器
  - 正则文法(RG)与FA的等价性。
  - 相互转换方法。
  - 带输出的有穷状态自动机。
  - 双向有穷状态自动机。

第3章 有穷状态自动机

- 重点：DFA的概念，DFA、NFA、ε-NFA、RG之间的等价转换思路与方法。
- 难点：对DFA概念的理解，DFA、RG的构造方法，RG与FA的等价性证明。

3.1 语言的识别

- 推导和归约中的回溯问题将对系统的效率产生极大的影响

  S→aA|aB

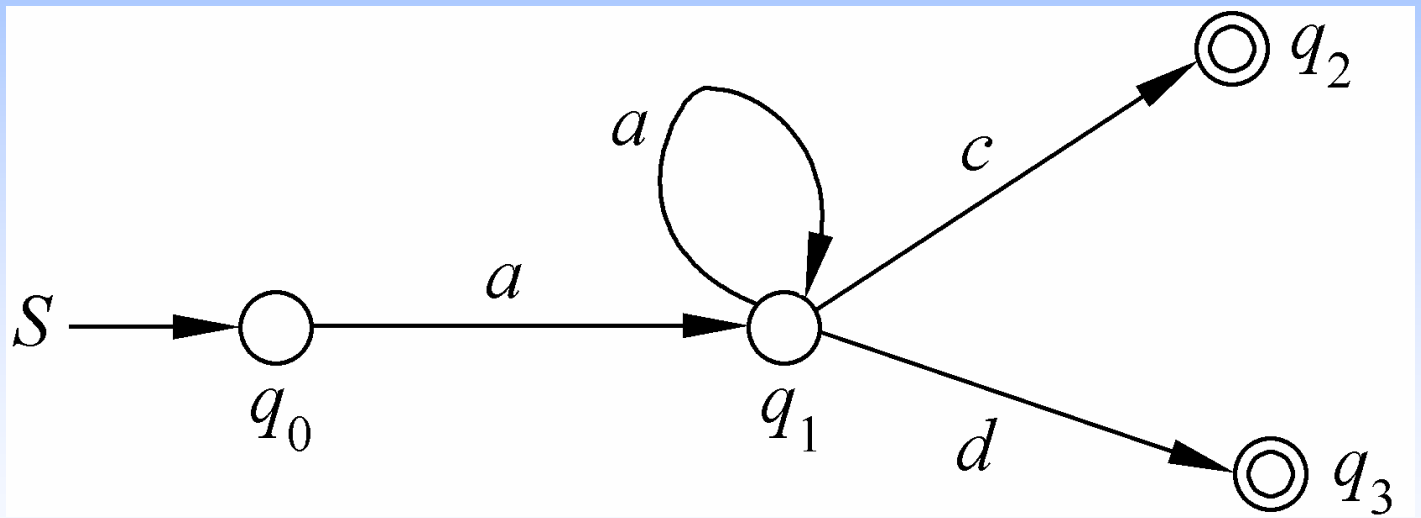
  A→aA|c

  B→aB|d

分析句子**aaac** 的过程中可能需要回溯。

3.1 语言的识别

- 系统识别语言$\{a^nc|n\geqslant 1\}\cup\{a^nd|n\geqslant 1\}$的字符串过程中状态的变化图示如下：



3.1 语言的识别

- 识别系统(模型)
- ⑴ 系统具有有穷个状态，不同的状态代表不同的意义。按照实际的需要，系统可以在不同的状态下完成规定的任务。
- ⑵ 我们可以将输入字符串中出现的字符汇集在一起构成一个字母表。系统处理的所有字符串都是这个字母表上的字符串。

3.1 语言的识别

- ⑶ 系统在任何一个状态(当前状态)下，从输入字符串中读入一个字符，根据当前状态和读入的这个字符转到新的状态。当前状态和新的状态可以是同一个状态，也可以是不同的状态；当系统从输入字符串中读入一个字符后，它下一次再读时，会读入下一个字符。这就是说，相当于系统维持有一个读写指针，该指针在系统读入一个字符后指向输入串的下一个字符。

3.1 语言的识别

- ⑷ 系统中有一个状态，它是系统的开始状态，系统在这个状态下开始进行某个给定句子的处理。
- ⑸ 系统中还有一些状态表示它到目前为止所读入的字符构成的字符串是语言的一个句子，把所有将系统从开始状态引导到这种状态的字符串放在一起构成一个语言，该语言就是系统所能识别的语言。

3.1 语言的识别

- 相应的物理模型

  一个右端无穷的输入带。

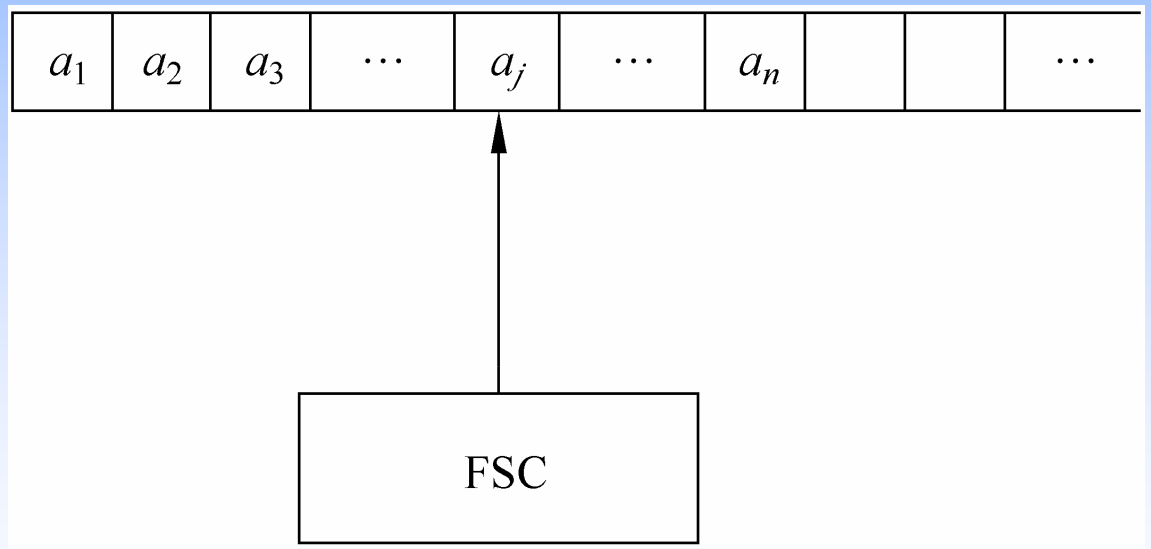
  一个有穷状态控制器**(finite state control,FSC)** 。

  一个读头。

- 系统的每一个动作由三个节拍构成：读入读头正注视的字符；根据当前状态和读入的字符改变有穷控制器的状态；将读头向右移动一格。

3.1 语言的识别

- 有穷状态自动机的物理模型



3.2有穷状态自动机

- **有穷状态自动机(finite automaton,FA)**

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

Q——状态的非空有穷集合。$\forall q \in Q$，**q**称为**M**的一个**状态(state)**。

Σ——**输入字母表(Input alphabet)**。输入字符串都是Σ上的字符串。

q_0——$q_0 \in Q$，是**M**的**开始状态(initial state)**，也可叫做初始状态或者启动状态。

3.2有穷状态自动机

δ——状态**转移函数(transition function)**，有时候又叫做状态转换函数或者移动函数。δ：$Q\times\Sigma\rightarrow Q$，对$\forall(q，a)\in Q\times\Sigma$，δ(q，a)=p表示：M在状态q读入字符a，将状态变成p，并将读头向右移动一个带方格而指向输入字符串的下一个字符。

F——$F\subseteq Q$，是M的**终止状态(final state)**集合。$\forall q\in F$，q称为M的**终止状态**，又称为**接受状态(accept state)**。

3.2有穷状态自动机

- **例 3-1** 下面是一个有穷状态自动机

$$M_1=(\{q_0, q_1, q_2\}, \{0\}, \delta_1, q_0, \{q_2\})$$

其中，$\delta_1(q_0,0)= q_1$，$\delta_1(q_1,0)= q_2$，$\delta_1(q_2,0)= q_1$

用表3-1表示 δ_1。

状态说明	状态	输入字符
		0
开始状态	q_0	q_1
	q_1	q_2
终止状态	q_2	q_1

3.2有穷状态自动机

$M_2=(\{q_0, q_1, q_2, q_3\}, \{0, 1, 2\}, \delta_2, q_0, \{q_2\})$

$\delta_2(q_0,0)= q_1$, $\delta_2(q_1,0)= q_2$

$\delta_2(q_2,0)= q_1$, $\delta_2(q_3,0)= q_3$

$\delta_2(q_0,1)= q_3$, $\delta_2(q_1,1)= q_3$

$\delta_2(q_2,1)= q_3$, $\delta_2(q_3,1)= q_3$

$\delta_2(q_0,2)= q_3$, $\delta_2(q_1,2)= q_3$

$\delta_2(q_2,2)= q_3$, $\delta_2(q_3,2)= q_3$

3.2有穷状态自动机

表3-2 δ_2转换函数

状态说明	状态	输入字符		
		0	1	2
开始状态	q_0	q_1	q_3	q_3
	q_1	q_2	q_3	q_3
终止状态	q_2	q_1	q_3	q_3
	q_3	q_3	q_3	q_3

3.2有穷状态自动机

- 将 δ 扩充为

$$\hat{\delta}: Q \times \Sigma^* \to Q$$

对任意的$\mathbf{q} \in \mathbf{Q}$，$\mathbf{w} \in \Sigma^*$，$\mathbf{a} \in \Sigma$，定义

$$(1)\hat{\delta}(q, \varepsilon) = q$$

$$(2)\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$$

3.2有穷状态自动机

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

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

$$=\delta(q,a)$$

两值相同，不用区分这两个符号。

3.2有穷状态自动机

- **确定的有穷状态自动机**
  - 由于对于任意的$q \in Q$， $a \in \Sigma$， $\delta(q, a)$均有确定的值，所以，将这种FA称为**确定的有穷状态自动机(deterministic finite automaton，DFA)**

3.2有穷状态自动机

- **M接受(识别)的语言**

  对于$\forall x \in \Sigma^*$如果$\delta(q, w) \in F$，则称x被M接受，如果$\delta(q, w) \notin F$，则称M不接受x。

  $$L(M)=\{x \mid x \in \Sigma^* \text{且} \delta(q, w) \in F\}$$

  称为由M接受(识别)的语言

  $$L(M_1)= L(M_2)=\{0^{2n} \mid n \geqslant 1\}$$

- 如果$L(M_1)=L(M_2)$，则称M_1与M_2等价。

3.2有穷状态自动机

- **例 3-2 构造一个DFA，它接受的语言为{x000y|x，y∈{0，1}*}**

q_0——M的启动状态；

q_1——M读到了一个0，这个0可能是子串“000”的第1个0；

q_2——M在q_1后紧接着又读到了一个0，这个0可能是子串“000”的第2个0；

q_3——M在q_2后紧接着又读到了一个0，发现输入字符串含有子串“000”；因此，这个状态应该是终止状态。

3.2有穷状态自动机

- $\delta(q_0,1)=q_0$——M在q_0读到了一个1，它需要继续在q_0“等待”可能是子串“000”的第1个0的输入字符0；
- $\delta(q_1,1)=q_0$——M在刚刚读到了一个0后，读到了一个1，表明在读入这个1之前所读入的0并不是子串“000”的第1个0，因此，M需要重新回到状态q_0，以寻找子串“000”的第1个0；

3.2有穷状态自动机

- $\delta(q_2,1)=q_0$——M在刚刚发现了00后，读到了一个1，表明在读入这个1之前所读入的00并不是子串“000”的前两个0，因此，M需要重新回到状态q_0，以寻找子串“000”的第1个0；
- $\delta(q_3,0)=q_3$——M找到了子串“000”，只用读入该串的剩余部分。
- $\delta(q_3,1)=q_3$——M找到了子串“000”，只用读入该串的剩余部分。

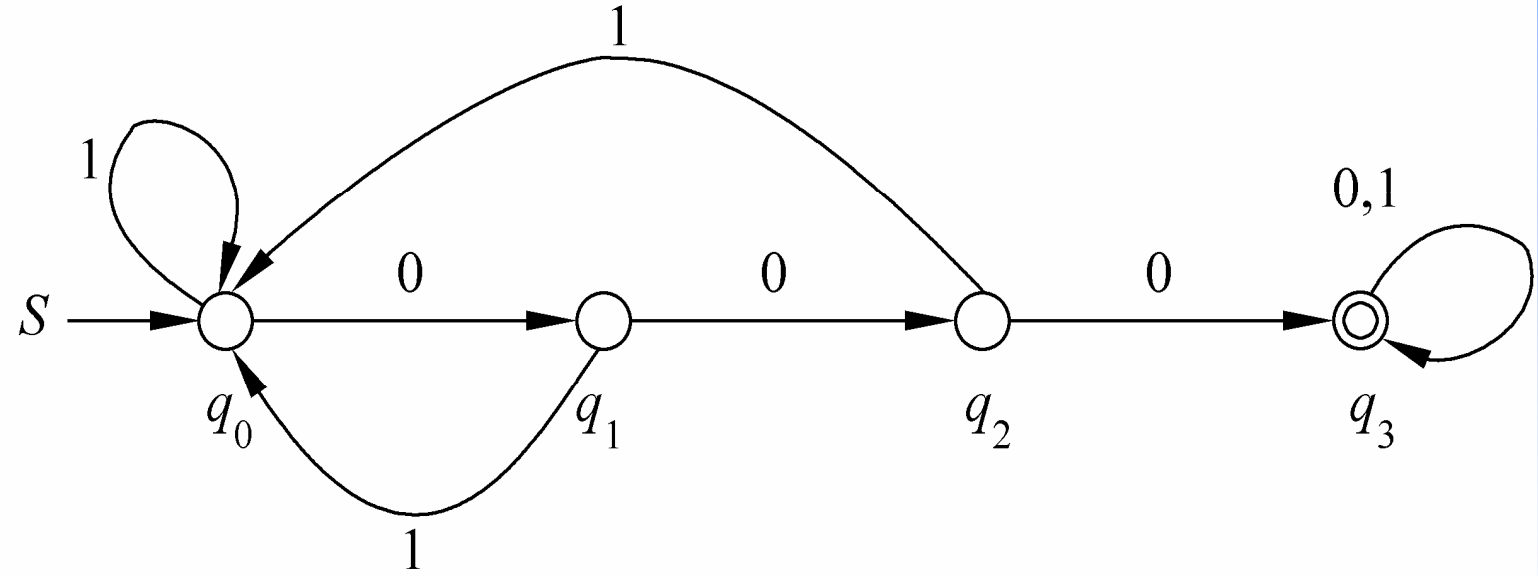
3.2有穷状态自动机

$M=(\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{\delta(q_0,0)=q_1, \delta(q_1,0)=q_2, \delta(q_2,0)=q_3, \delta(q_0,1)=q_0, \delta(q_1,1)=q_0, \delta(q_2,1)=q_0, \delta(q_3,0)=q_3, \delta(q_3,1)=q_3\}, q_0, \{q_3\})$

状态说明	状态	输入字符	
		0	1
开始状态	q_0	q_1	q_0
	q_1	q_2	q_0
	q_2	q_3	q_0
终止状态	q_3	q_3	q_3

3.2有穷状态自动机

- 一种更为直观的表示



3.2有穷状态自动机

- **状态转移图(transition diagram)**

  ⑴ **q∈Q ⇔ q**是该有向图中的一个顶点；

  ⑵ **δ(q，a)=p** ⇔ 图中有一条从顶点**q**到顶点**p**的标记为**a**的弧；

  ⑶ **q∈F** ⇔ 标记为**q**的顶点被用双层圈标出；

  ⑷ 用标有**S**的箭头指出**M**的开始状态。

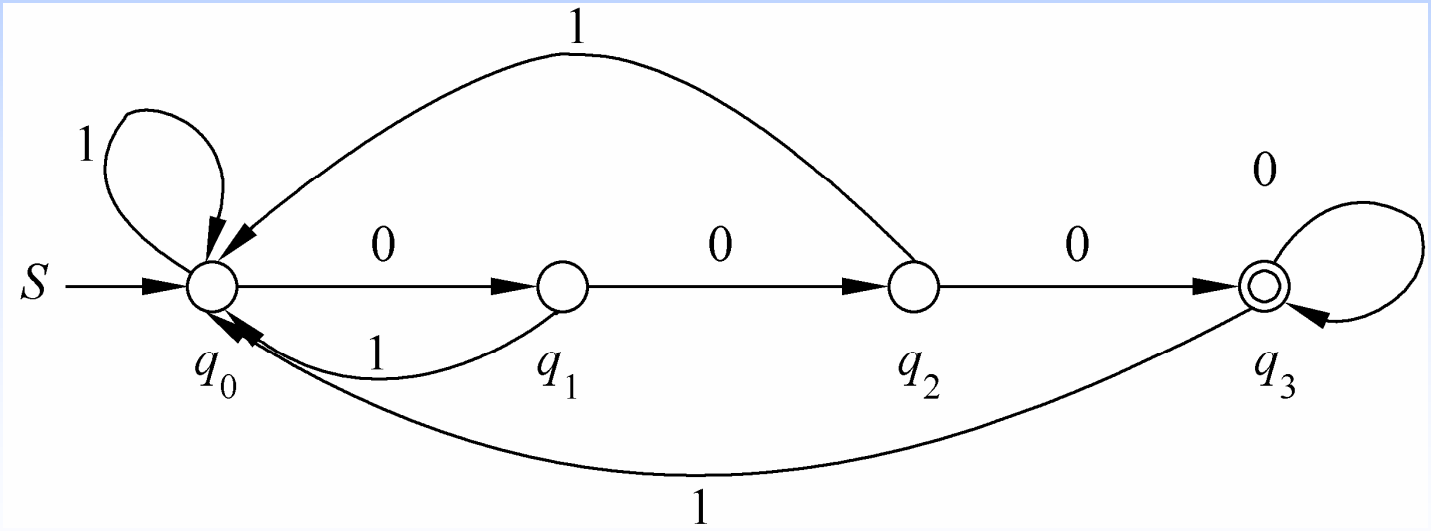
- 状态转移图又可以叫做**状态转换图**。

3.2有穷状态自动机

- 例 3-3构造一个DFA，它接受的语言为{x000|x∈{0，1}*}。
  - 状态q_0读到的0可能是输入字符串的最后三个0的第1个0；
  - 在状态q_1紧接着读到的0可能是输入字符串的最后三个0的第2个0；
  - 在状态q_2紧接着读到的0可能是输入字符串的最后三个0的第3个0；

3.2有穷状态自动机

- 在状态q_3紧接着读到的0也可能是输入字符串的最后三个0的第3个0；
- 如果在状态q_1，q_2，q_3读到的是1，则要重新检查输入串是否以三个0结尾。



3.2有穷状态自动机

- 几点值得注意

(1) 定义FA时，常常只给出FA相应的状态转移图就可以了。

(2) 对于DFA来说，并行的弧按其上的标记字符的个数计算，对于每个顶点来说，它的出度恰好等于输入字母表中所含的字符的个数。

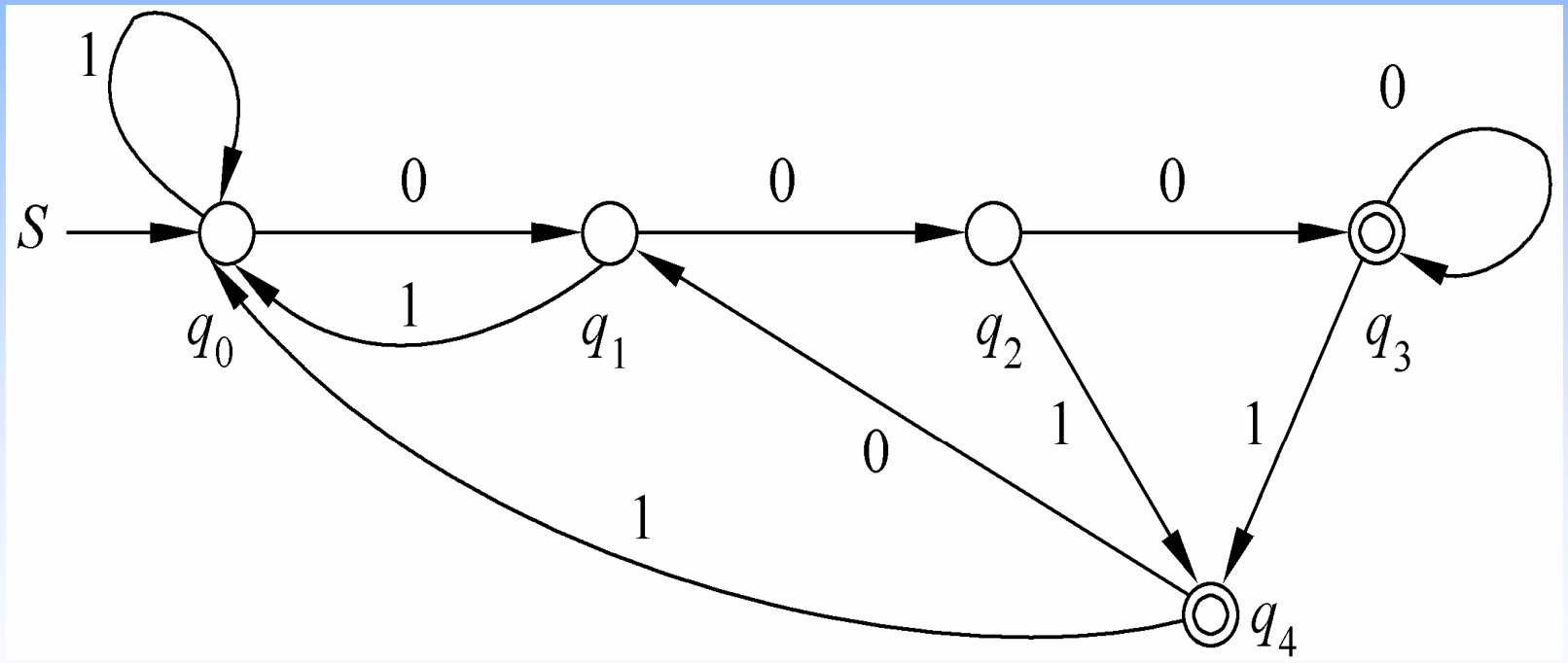
3.2有穷状态自动机

(3) 不难看出，字符串x被FA M接受的充分必要条件是，在M的状态转移图中存在一条从开始状态到某一个终止状态的有向路，该有向路上从第1条边到最后一条边的标记依次并置而构成的字符串x。简称此路的标记为x。

(4) 一个FA可以有多于1个的终止状态。

3.2有穷状态自动机

接受语言{x000|x∈{0，1}*}∪{x001|x∈{0，1}*}的FA



3.2有穷状态自动机

- **即时描述(instantaneous description，ID）**
  - **x，y∈Σ*，δ(q0，x)=q, xqy称为M的一个即时描述，表示xy是M正在处理的一个字符串，x引导M从q0启动并到达状态q，M当前正注视着y的首字符。**
  - **如果xqay是M的一个即时描述，且δ(q，a)=p，则xqay ⊢M xapy。**

3.2有穷状态自动机

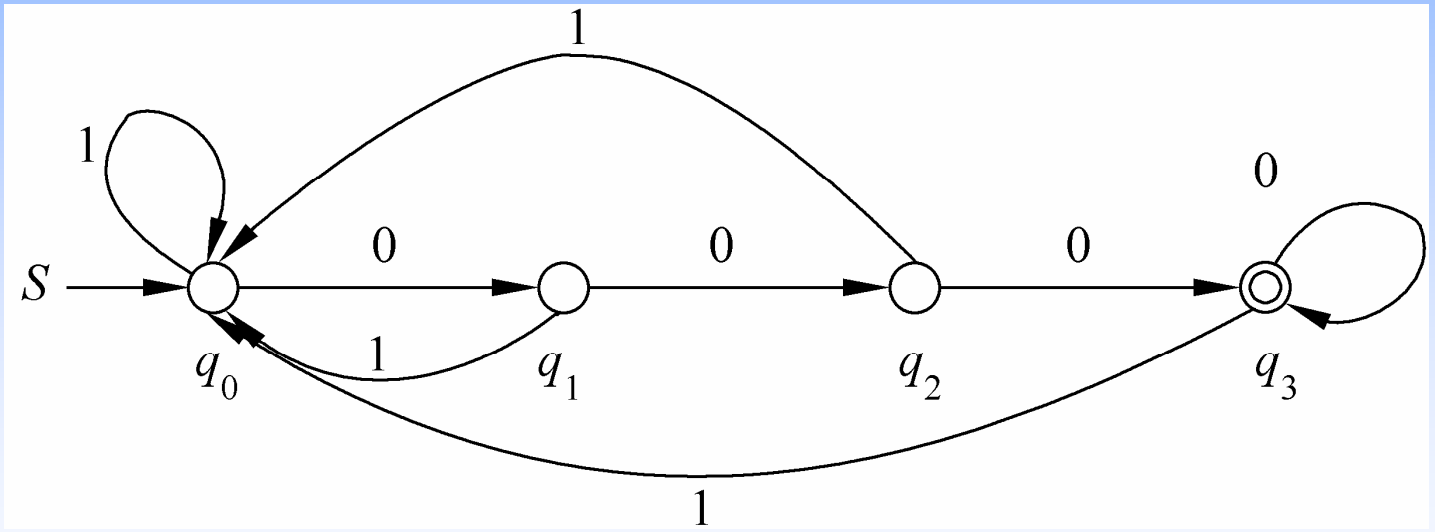
- $\alpha \vdash_M^n \beta$：表示M从即时描述α经过n次移动到达即时描述β。
- M存在即时描述α_1，α_2，……，α_{n-1}，使得
- $\alpha \vdash_M \alpha_1$，$\alpha_1 \vdash_M \alpha_2$，…，$\alpha_{n-1} \vdash_M \beta$
- 当n=0时，有α=β。即$\alpha \vdash_M^0 \alpha$。
- $\alpha \vdash_M^+ \beta$：表示M从即时描述α经过至少1次移动到达即时描述β。
- $\alpha \vdash_M^* \beta$：表示M从即时描述α经过若干步移动到达即时描述β。

3.2有穷状态自动机

- 当意义清楚时，我们将符号 $\vdash_M$、$\vdash_M^n$、$\vdash_M^*$、$\vdash_M^+$中的M省去，分别用 $\vdash$、$\vdash^n$、$\vdash^*$、$\vdash^+$表示。

3.2有穷状态自动机

对下图所示的DFA有如下的ID转换：



3.2有穷状态自动机

$$
\begin{aligned}
q_0 1010010001 &\vdash 1q_0 010010001 \\
&\vdash 10q_1 10010001 \\
&\vdash 101q_0 0010001 \\
&\vdash 1010q_1 010001 \\
&\vdash 10100q_2 10001 \\
&\vdash 101001q_0 0001 \\
&\vdash 1010010q_1 001 \\
&\vdash 10100100q_2 01
\end{aligned}
$$

3.2有穷状态自动机

$$\vdash 1010010000q_3 1$$

$$\vdash 1010010001q_0$$

即 $q_0 1010010001 \vdash^{10} 1010010001q_0$

$q_0 1010010001 \vdash^{+} 1010010001q_0$

$q_0 1010010001 \vdash^{*} 1010010001q_0$

3.2有穷状态自动机

- 对于$x \in \Sigma^*$,

$$q_0x1 \vdash^+ x1q_0$$
$$q_0x10 \vdash^+ x10q_1$$
$$q_0x100 \vdash^+ x100q_2$$
$$q_0x000 \vdash^+ x000q_3$$

3.2有穷状态自动机

能引导FA从开始状态到达q的字符串的集合为：

$$set(q)=\{x \mid x \in \Sigma^{*},\ \delta(q_0, x)=q\}$$

对图3-3所给的DFA 中的所有q，求set(q)。

$set(q_0)=\{x \mid x \in \Sigma^*,\ x=\varepsilon$ 或者x以1结尾$\}$

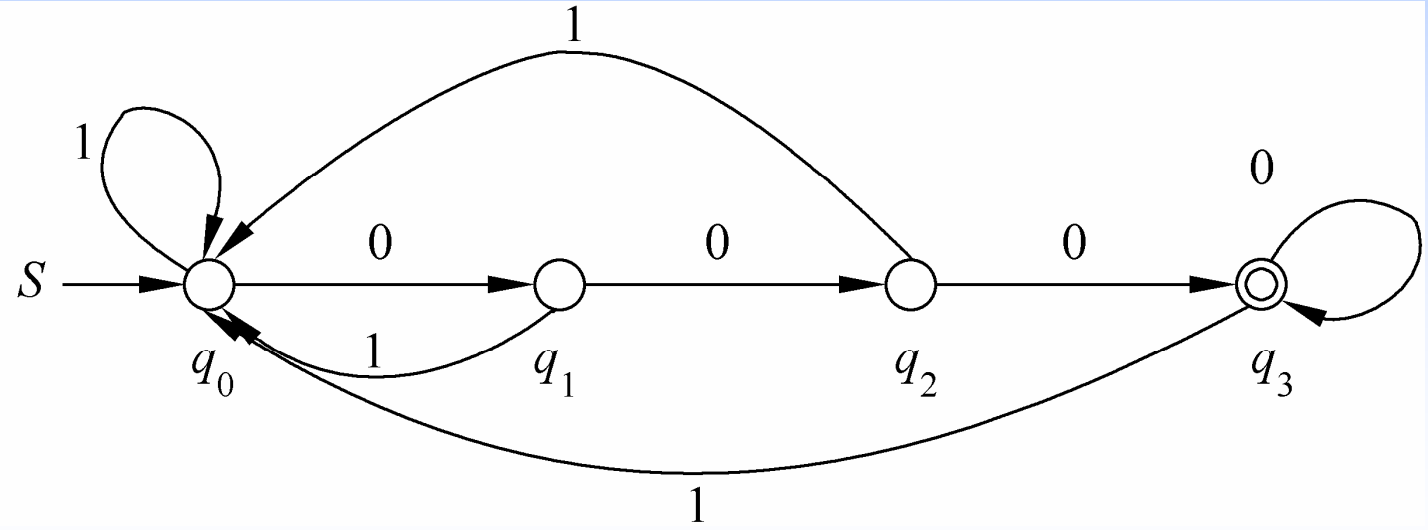
$set(q_1)=\{x \mid x \in \Sigma^*,\ x=0$或者x以10结尾$\}$

$set(q_2)=\{x \mid x \in \Sigma^*,\ x=00$或者x以100结尾$\}$

$set(q_3)=\{x \mid x \in \Sigma^*,$ x以000结尾$\}$

$set(q_4)=\{x \mid x \in \Sigma^*,$ x以001结尾$\}$

这5个集合是两两互不相交。



3.2有穷状态自动机

- 对于任意一个**FA M=(Q，Σ，δ，q_0，F)**我们可以按照如下方式定义关系R_M:
- 对∀**x，y∈Σ*，xR$_M$y ⇔ ∃q∈Q，**使得**x∈set(q)**和**y∈set(q)**同时成立。
- 按照这个定义所得到的关系实际上是Σ*上的一个等价关系。利用这个关系，可以将Σ*划分成不多于**|Q|**个等价类。

3.2有穷状态自动机

- 例 3-4 构造一个DFA，它接受的语言为$\{0^n1^m2^k|n,m,k\geqslant 1\}$。

q_0——M的启动状态；

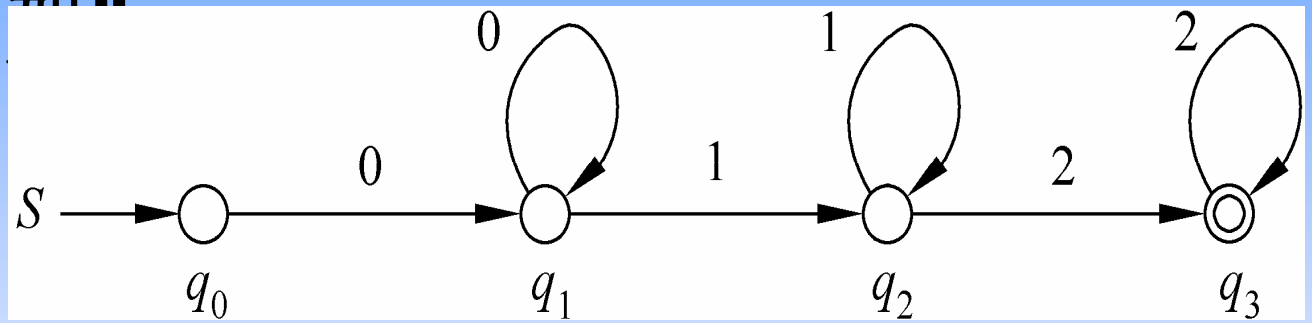
q_1——M读到至少一个0，并等待读更多的0；

q_2——M读到至少一个0后，读到了至少一个1，并等待读更多的1；

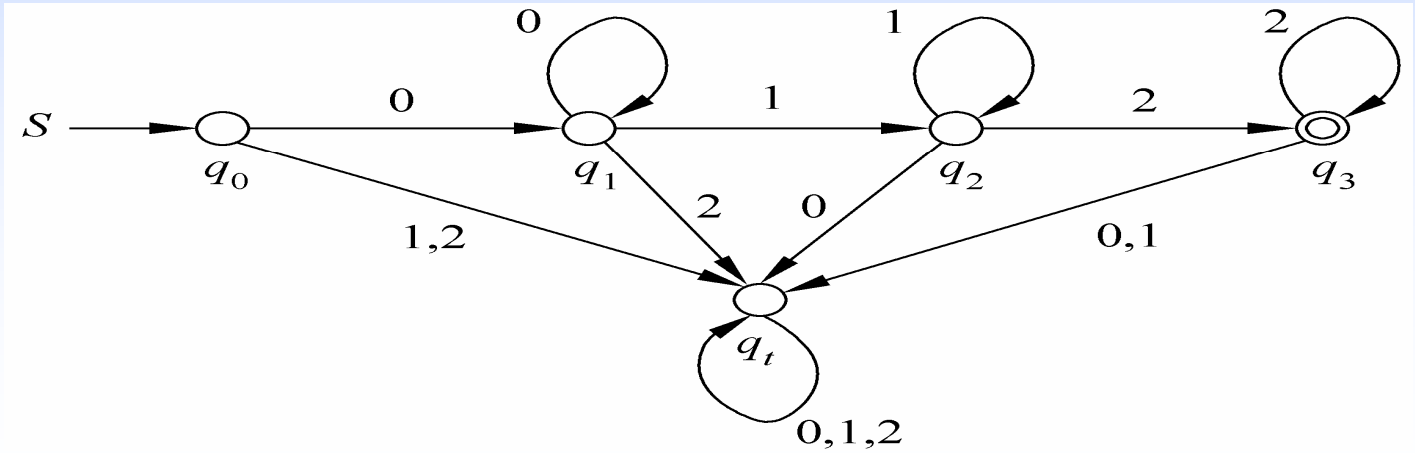
q_3——M读到至少一个0后跟至少一个1后，并且接着读到了至少一个2。

3.2有穷状态自动机

- 先设计“主体框



- 再补充细节



3.2有穷状态自动机

⑴ 当FA一旦进入状态q_t，它就无法离开此状态。所以，q_t相当于一个陷阱状态(trap)。一般地，我们将陷阱状态用作在其他状态下发现输入串不可能是该FA所识别的语言的句子时进入的状态。在此状态下，FA读完输入串中剩余的字符。

3.2有穷状态自动机

⑵ 在构造一个识别给定语言的FA时，用画图的方式比较方便、直观。我们可以先根据语言的主要特征画出该FA的“主体框架”，然后再去考虑画出一些细节要求的内容。

3.2有穷状态自动机

(3) FA的状态具有一定的记忆功能：不同的状态对应于不同的情况，由于FA只有有穷个状态，所以，在识别一个语言的过程中，如果有无穷种情况需要记忆，我们肯定是无法构造出相应的FA的。

3.2有穷状态自动机

- 例 3-5构造一个DFA，它接受的语言为{x|x∈{0，1}*，且当把x看成二进制数时，x模3与0同余}。
- q_0——对应除以3余数为0的x组成的等价类;
- q_1——对应除以3余数为1的x组成的等价类;
- q_2——对应除以3余数为2的x组成的等价类;
- q_0——M的开始状态

3.2有穷状态自动机

- q_s——在此状态下读入0时，有x=0，所以应该进入状态q_0；读入1时，有x=1，所以应该进入状态q_1。即：$\delta(q_s, 0)= q_0$；$\delta(q_s, 1)= q_1$。

3.2有穷状态自动机

- **q_0——能引导M到达此状态的x除以3余0，所以有：x=3*n+0。**
- **读入0时，引导M到达下一个状态的字符串为 x0 ， x0=2*(3*n+0)=3*2*n+0 。 所 以 ，$\delta(q_0, 0)= q_0$;**
- **读入1时，M到达下一个状态的字符串为x1，x1=2*(3*n+0)+1=3*2*n+1。所以，$\delta(q_0, 1)= q_1$;**

3.2有穷状态自动机

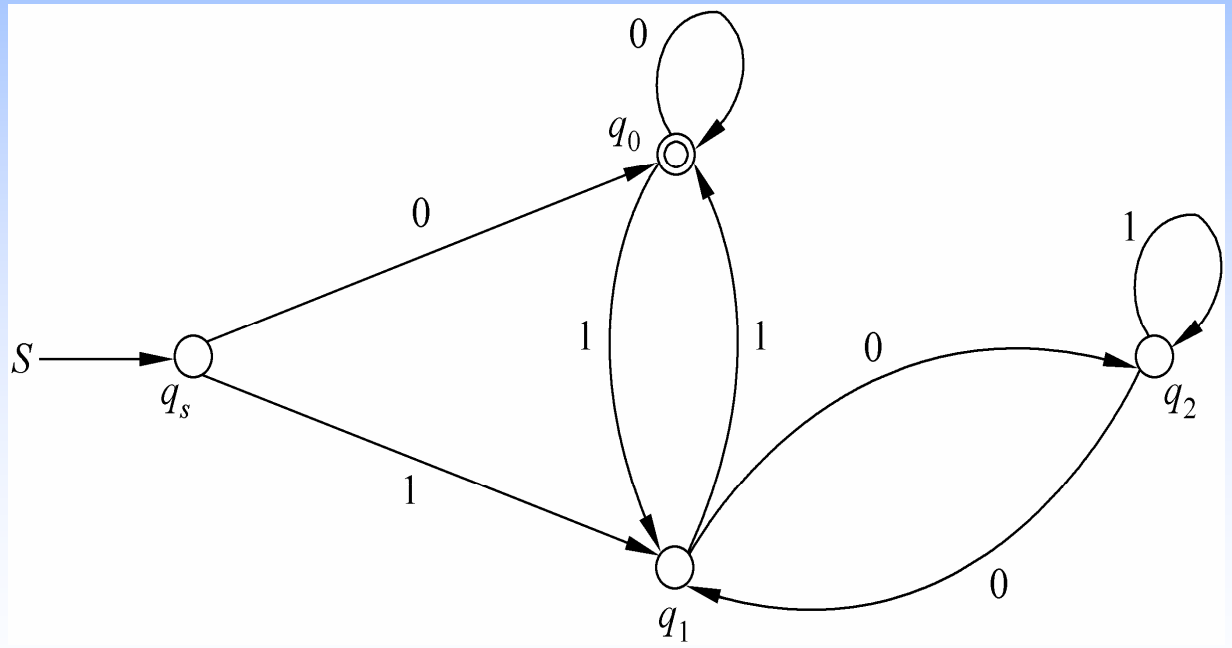
- **q_1——能引导M到达此状态的x除以3余1，所以有：x=3*n+1。**
- **读入0时，引导M到达下一个状态的字符串为x0，x0=2*(3*n+1)=3*2*n+2。所以即：$\delta(q_1, 0)= q_2$;**
- **读入1时，引导M到达下一个状态的字符串为x1，x1=2*(3*n+1)+1=3*2*n+2+1=3*(2*n+1)。所以$\delta(q_1, 1)= q_0$**

3.2有穷状态自动机

- **q_2——能引导M到达此状态的x除以3余2，所以：x=3*n+2。**
- **读入0时，引导M到达下一个状态的字符串为x0，x0=2*(3*n+2)=3*2*n+4=3*(2*n+1)+1 。 所 以 $\delta(q_2, 0)= q_1$;**
- **读入1时，引导M到达下一个状态的字符串为x1，x1=2*(3*n+2)+1=3*2*n+4+1=3*(2*n+1)+2。所以， $\delta(q_2, 1)= q_2$ 。**

3.2有穷状态自动机

- 接受语言{x|x∈{0，1}*，且当把x看成二进制数时，x模3与0同余}的DFA如下：



3.2有穷状态自动机

- **例 3-6** 构造一个DFA，它接受的语言 $L=\{x|x\in\{0，1\}^*$，且对x中任意一个长度不大于5的子串$a_1a_2\ldots a_n$，$a_1+a_2+\ldots+a_n\leqslant 3$，$n\leqslant 5\}$。

- 输入串为 $a_1a_2\ldots a_i\ldots a_{i+4}a_{i+5}\ldots a_m$

3.2有穷状态自动机

- 当i=1，2，3，也就是M读到输入串的第1、2、3个字符时，它需要将这些字符记下来。因为$a_1……a_i$可能需要用来判定输入串的最初4~5个字符组成的子串是否满足语言的要求。
- 当i=4，5，也就是M读到输入串的第4、5个字符时，在$a_1+a_2+……+a_i \leqslant 3$的情况下，M需要将$a_1……a_i$记下来；在$a_1+a_2+……+a_i>3$时，M应该进入陷阱状态$q_t$。

3.2有穷状态自动机

- 当i=6，也就是M读到输入串的第6个字符，此时，以前读到的第1个字符a_1就没有用了，此时它要看$a_2+a_3+\ldots+a_6 \leqslant 3$是否成立，如果成立，M需要将$a_2 \ldots a_6$记下来；在$a_2+a_3+\ldots+a_i>3$时，M应该进入陷阱状态$q_t$。

3.2有穷状态自动机

- 当M完成对子串$a_1a_2\ldots a_i\ldots a_{i+4}$的考察，并发现它满足语言的要求时，M记下来的是$a_i\ldots a_{i+4}$，此时它读入输入串的第i+5个字符$a_{i+5}$，以前读到的第i个字符$a_i$就没有用了，此时它要看$a_{i+1}+a_{i+2}+\ldots+a_{i+5}\leqslant 3$是否成立，如果成立，M需要将$a_{i+1}$，$a_{i+2}$，…，$a_{i+5}$记下来；在$a_{i+1}+a_{i+2}+\ldots+a_{i+5}>3$时，M应该进入陷阱状态$q_t$。

3.2有穷状态自动机

- **M需要记忆的内容有：**
- **什么都未读入——2^0=1种；**
- **记录有1个字符——2^1=2种；**
- **记录有2个字符——2^2=4种；**
- **记录有3个字符——2^3=8种；**
- **记录有4个字符——2^4-1=15种；**
- **记录有5个字符——2^5-6=26种；**
- **记录当前的输入串不是句子——1种。**

3.2有穷状态自动机

状态设置

$q[\varepsilon]$——M还未读入任何字符；

q_t——陷阱状态；

$q[a_1a_2\ldots a_i]$——M记录有i个字符，$1\leqslant i\leqslant 5$。a_1，a_2，…，$a_i\in\{0，1\}$。

取DFA $M=(Q，\{0，1\}，\delta，q[\varepsilon]，F)$

$F=\{q[\varepsilon]\}\cup\{q[a_1a_2\ldots a_i]|\ a_1，a_2，\ldots，a_i\in\{0，1\}$且$1\leqslant i\leqslant 5$且$a_1+a_2+\ldots+a_i\leqslant 3\}$

$Q=\{q_t\}\cup F$

3.2有穷状态自动机

- $\delta(q[\varepsilon], a_1)=q[a_1]$
- $\delta(q[a_1], a_2)=q[a_1a_2]$
- $\delta(q[a_1a_2], a_3)=q[a_1a_2a_3]$

$$\delta(q[a_1a_2a_3], a)=\begin{cases} q[a_1a_2a_3a] & \text{如果} a_1+a_2+a_3+a\leqslant 3 \\ q_t & \text{如果} a_1+a_2+a_3+a>3 \end{cases}$$

3.2有穷状态自动机

$$\delta(q[a_1a_2a_3a_4],\ a)=\begin{cases} q[a_1a_2a_3a_4a] & \text{如果} a_1+a_2+a_3+a_4+a\leqslant 3 \\ q_t & \text{如果} a_1+a_2+a_3+a_4+a>3 \end{cases}$$

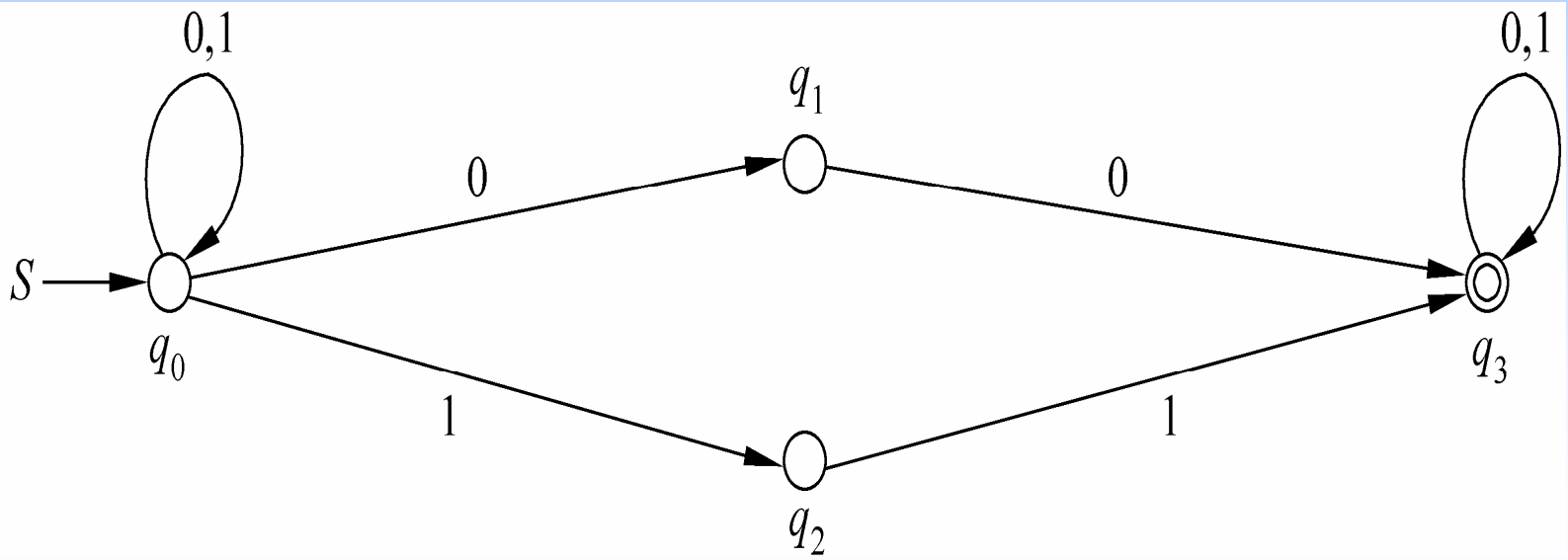
$$\delta(q[a_1a_2a_3a_4a_5],\ a)=\begin{cases} q[a_2a_3a_4a_5a] & a_2+a_3+a_4+a_5+a\leqslant 3 \\ q_t & \text{如果} a_2+a_3+a_4+a_5+a>3 \end{cases}$$

$$\delta(q_t,\ a_1)=q_t$$

3.3 NFA

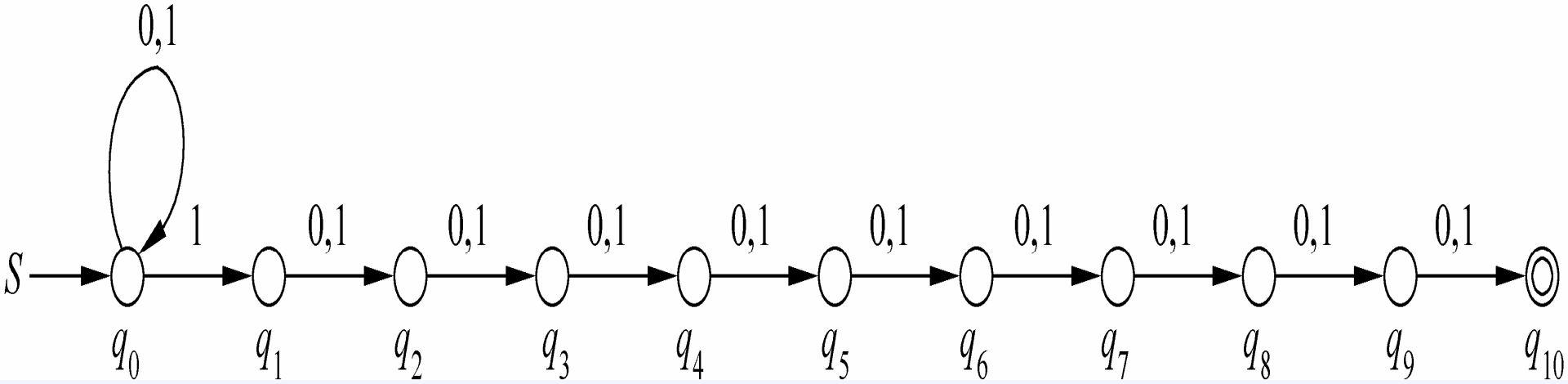
3.3.1 作为对DFA的修改

- 希望是接受{x|x∈{0，1}*，且x含有子串00或11}的FA如下：



3.3.1 作为对DFA的修改

- 希望是接受{x|x∈{0，1}*，且x 的倒数第10个字符为1}的FA如下：



3.3.1 作为对DFA的修改

- 这两个图所给的“FA”与前面我们所定义的FA，即DFA，的区别在于：

  ⑴ 并不是对于所有的(q，a)∈Σ×Q，δ(q，a)都有一个状态与它对应；

  ⑵ 并不是对于所有的(q，a)∈Σ×Q，δ(q，a)只对应一个状态。
- “FA”在任意时刻可以处于有穷多个状态。
- “FA”具有“智能”。

3.3.2 NFA的形式定义

- **不确定的有穷状态自动机(non-deterministic finite automaton ,NFA)**

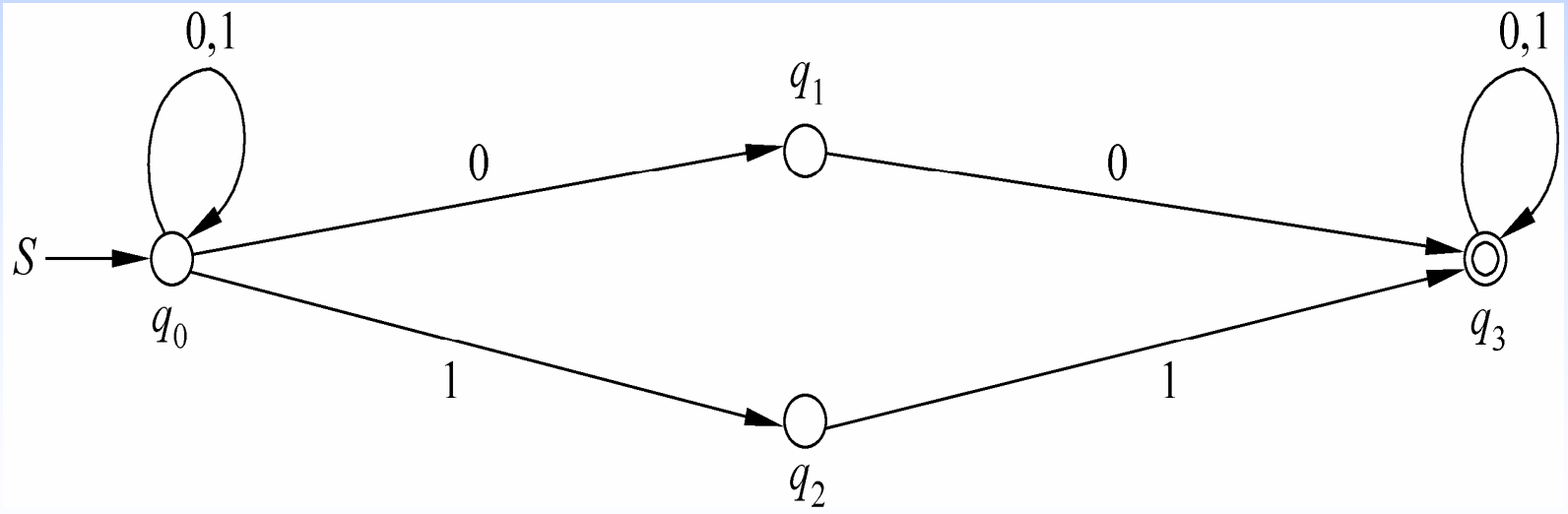
  M是一个五元组

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

  - **Q**、Σ、q_0、**F**的意义同**DFA**。
  - δ：$Q\times\Sigma\rightarrow 2^Q$，对$\forall(q,\ a)\in Q\times\Sigma$，$\delta(q,\ a)=\{p_1,\ p_2,\ \ldots,\ p_m\}$表示**M**在状态**q**读入字符**a**，可以选择地将状态变成p_1、或者p_2、…、或者p_m，并将读头向右移动一个带方格而指向输入字符串的下一个字符。

3.3.2 NFA的形式定义

- FA的状态转移图、FA的状态对应的等价类、FA的即时描述对NFA都有效。
- 接受{x|x∈{0，1}*，且x含有子串00或11}的FA对应的移动函数定义表。

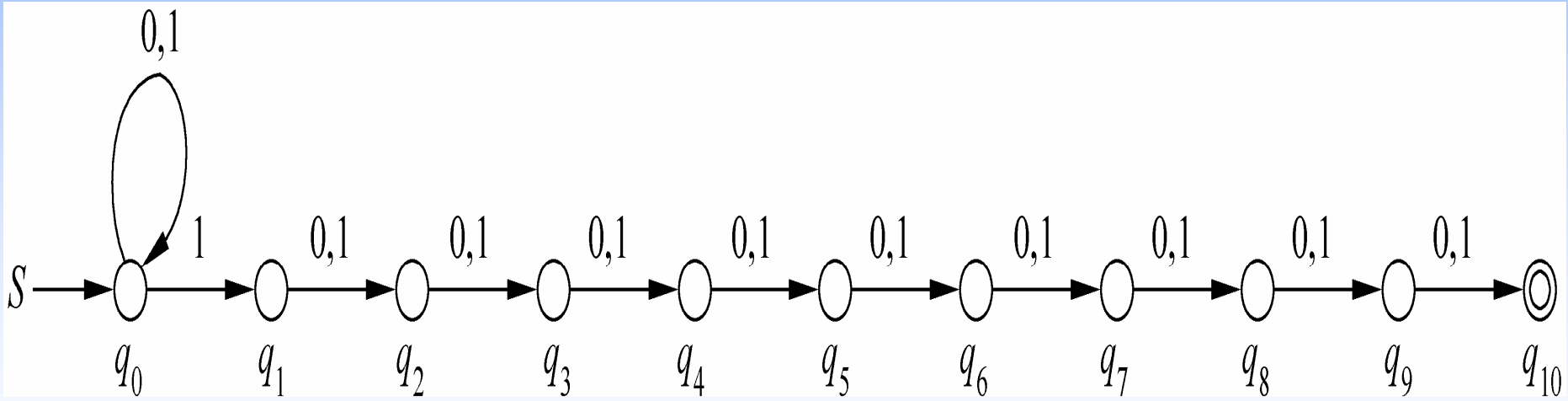


3.3.2 NFA的形式定义

状态说明	状态	输入字符	
		0	1
启动状态	q_0	$\{q_0, q_1\}$	$\{q_0, q_2\}$
	q_1	$\{q_3\}$	Φ
	q_2	Φ	$\{q_3\}$
终止状态	q_3	$\{q_3\}$	$\{q_3\}$

3.3.2 NFA的形式定义

- 接受{x|x∈{0，1}*，且x 的倒数第10个字符为1}的FA对应的移动函数定义表。



3.3.2 NFA的形式定义

状态说明	状态	输入字符	
		0	1
启动状态	q_0	$\{q_0\}$	$\{q_0, q_1\}$
	q_1	$\{q_2\}$	$\{q_2\}$
	q_2	$\{q_3\}$	$\{q_3\}$
	q_3	$\{q_4\}$	$\{q_4\}$
	q_4	$\{q_5\}$	$\{q_5\}$
	q_5	$\{q_6\}$	$\{q_6\}$
	q_6	$\{q_7\}$	$\{q_7\}$
	q_7	$\{q_8\}$	$\{q_8\}$
	q_8	$\{q_9\}$	$\{q_9\}$
	q_9	$\{q_{10}\}$	$\{q_{10}\}$
终止状态	q_{10}	Φ	Φ

3.3.2 NFA的形式定义

- 将 δ 扩充为

$$\hat{\delta}: Q \times \Sigma^* \to 2^Q$$

对任意的**q**∈**Q**，**w**∈Σ*，**a**∈Σ，定义

$$(1)\hat{\delta}(q,\varepsilon) = q$$

$$(2)\hat{\delta}(q,wa) = \{\mathrm{p} \mid \$\mathrm{r} \in (\mathrm{q},\ \mathrm{w}),\ 使得 \mathrm{p} \in \delta(\mathrm{r},\ \mathrm{a})\}$$

3.3.2 NFA的形式定义

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

$$=\{\mathrm{p}\mid\exists \mathrm{r}\in(\mathrm{q},\varepsilon),\mathrm{p}\in\delta(\mathrm{r},\mathrm{a})\}$$

$$=\{\mathrm{p}\mid\exists r\in\{q\},\mathrm{p}\in\delta(\mathrm{r},\mathrm{a})\}$$

$$=\{\mathrm{p}\mid\mathrm{p}\in\delta(\mathrm{q},\mathrm{a})\}$$

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

和关于DFA的结论一样，两值相同，也不用区分这两个符号。

3.3.2 NFA的形式定义

- 进一步扩充 δ 的定义域： δ：
  $2^Q \times \Sigma^* \to 2^Q$。对任意的$P \subseteq Q$，$w \in \Sigma^*$

$$\delta(P, w) = \bigcup_{q \in P} \delta(q, w)$$

3.3.2 NFA的形式定义

- 由于，对$\forall(q, w)\in Q\times\Sigma^*$

$$\delta(\{q\}, w) = \bigcup_{q\in\{q\}} \delta(q, w)$$

$$= \delta(q, w)$$

所以，不一定严格地区分 δ 的第1个分量是一个状态还是一个含有一个元素的集合。

3.3.2 NFA的形式定义

- 对任意的$q \in Q$，$w \in \Sigma^*$，$a \in \Sigma$：

  $\delta(q, wa) = \delta(\delta(q, w), a)$

- 对输入字符串$a_1a_2\ldots a_n$

  $\delta(q, a_1a_2\ldots a_n) = \delta((\ldots\delta(\delta(q, a_1), a_2), \ldots), a_n)$。

3.3.2 NFA的形式定义

- **M接受(识别)的语言**
  - 对于$\forall x \in \Sigma^*$，如果$\delta(q_0, w) \cap F \neq \Phi$，则称x被M接受，如果$\delta(q_0, w) \cap F = \Phi$，则称M不接受x。
  - $L(M)=\{x \mid x \in \Sigma^*$且$\delta(q_0, w) \cap F \neq \Phi\}$，称为由M接受(识别)的语言。

3.3.3 NFA与DFA等价

- 对于一个输入字符，NFA与DFA的差异是前者可以进入若干个状态，而后者只能进入一个惟一的状态。虽然从DFA看待问题的角度来说，NFA在某一时刻同时进入若干个状态，但是，这若干个状态合在一起的“总效果”相当于它处于这些状态对应的一个“综合状态”。因此，我们考虑让DFA用一个状态去对应NFA的一组状态。

3.3.3 NFA与DFA等价

- NFA $M_1=(Q, \Sigma, \delta_1, q_0, F_1)$与DFA $M_2=(Q_2, \Sigma, \delta_2, q_0', F_2)$的对应关系：
  - NFA从开始状态q_0启动，我们就让相应的DFA从状态$[q_0]$启动。所以$q_0'=[q_0]$。
  - 对于NFA 的一个状态组$\{q_1, q_2, \ldots, q_n\}$，如果NFA在此状态组时读入字符a后可以进入状态组$\{p_1, p_2, \ldots, p_m\}$,则让相应的DFA在状态$[q_1, q_2, \ldots, q_n]$读入字符a时，进入状态$[p_1, p_2, \ldots, p_m]$。

3.3.3 NFA与DFA等价

定理 3-1 NFA与DFA等价。

证明：

(1) 构造与M_1等价的DFA M_2。

$M_1=(Q, \Sigma, \delta_1, q_0, F_1)$

$M_2=(Q_2, \Sigma, \delta_2, [q_0], F_2)$

$Q_2=2^Q$

$F_2=\{[p_1, p_2, \ldots, p_m] | \{p_1, p_2, \ldots, p_m\} \subseteq Q \& \{p_1, p_2, \ldots, p_m\} \cap F_1 \neq \Phi\}$

3.3.3 NFA与DFA等价

$\delta_2([q_1, q_2, \ldots, q_n], a)=[p_1, p_2, \ldots, p_m]$

$\Leftrightarrow \delta_1(\{q_1, q_2, \ldots, q_n\}, a)= \{p_1, p_2, \ldots, p_m\}$

(2) 证明 $\delta_1(q_0, x)= \{p_1, p_2, \ldots, p_m\}$

$\Leftrightarrow \delta_2([q_0], x)=[p_1, p_2, \ldots, p_m]$。

设$x \in \Sigma^*$，施归纳于$|x|$

$x= \varepsilon$ ， $\delta_1(q_0, \varepsilon)= \{q_0\}$, $\delta_2([q_0], \varepsilon)=[q_0]$

3.3.3 NFA与DFA等价

设当|x|=n是结论成立。下面证明当|x|=n+1是结论也成立。不妨设x=wa，|w|=n，a∈Σ

$$\delta_1(q_0, wa) = \delta_1(\delta_1(q_0, w), a)$$
$$= \delta_1(\{q_1, q_2, \ldots, q_n\}, a)$$
$$= \{p_1, p_2, \ldots, p_m\}$$

由归纳假设，

$$\delta_1(q_0, w) = \{q_1, q_2, \ldots, q_n\} \Leftrightarrow \delta_2([q_0], w) = [q_1, q_2, \ldots, q_n]$$

3.3.3 NFA与DFA等价

根据δ_2的定义，

$\delta_2([q_1, q_2, \ldots, q_n], a)=[p_1, p_2, \ldots, p_m]$
$\Leftrightarrow \delta_1(\{q_1, q_2, \ldots, q_n\}, a)=\{p_1, p_2, \ldots, p_m\}$

所以，

$$\delta_2([q_0], wa)= \delta_2(\delta_2([q_0], w), a)$$
$$= \delta_2([q_1, q_2, \ldots, q_n], a)$$
$$=[p_1, p_2, \ldots, p_m]$$

3.3.3 NFA与DFA等价

故，如果$\delta_1(q_0, wa)= \{p_1, p_2, \ldots, p_m\}$则必有$\delta_2([q_0], wa)= [p_1, p_2, \ldots, p_m]$。

由上述推导可知，反向的推导也成立。这就是说，结论对$|x|=n+1$也成立。

由归纳法原理，结论对$\forall x \in \Sigma^*$成立。

3.3.3 NFA与DFA等价

(3) 证明$L(M_1)=L(M_2)$

设$x \in L(M_1)$，且$\delta_1(q_0, x)=\{p_1, p_2, \cdots, p_m\}$，

从而$\delta_1(q_0, x) \cap F_1 \neq \Phi$，

这就是说，$\{p_1, p_2, \cdots, p_m\} \cap F_1 \neq \Phi$，

由F_2的定义，$[p_1, p_2, \cdots, p_m] \in F_2$。

3.3.3 NFA与DFA等价

再由(2)知，

$\delta_2([q_0], x)=[p_1, p_2, \cdots, p_m]$

所以，$x \in L(M_2)$。故$L(M_1) \subseteq L(M_2)$。

反过来推，可得$L(M_2) \subseteq L(M_1)$。

从而$L(M_1)=L(M_2)$得证。

综上所述，定理成立。

•例 3-7 图3-9所示的NFA 对应的DFA的状态转移函数如表3-7所示。

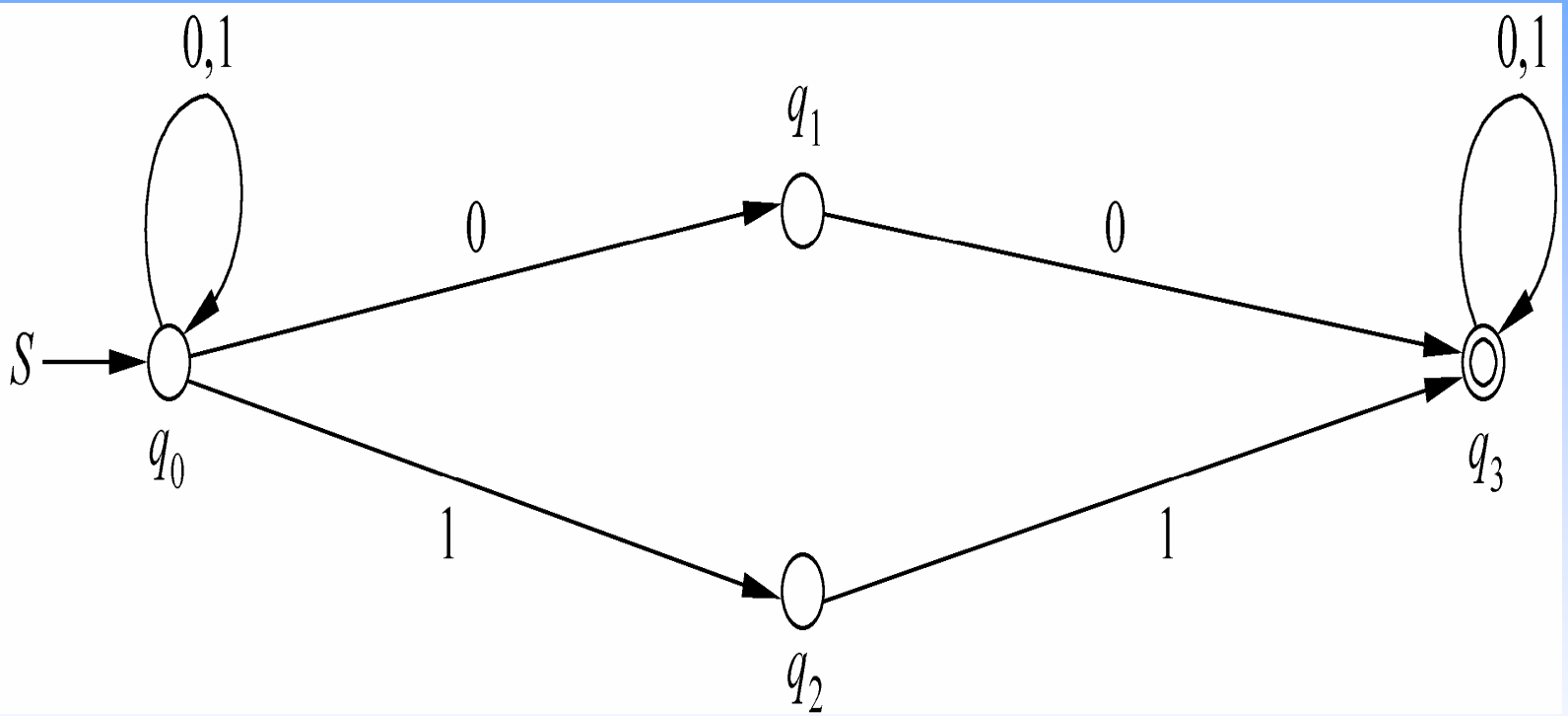


图3-9

表3-7 状态转移函数

状态说明		状态	输入字符	
			0	1
启动	√	$[q_0]$	$[q_0,q_1]$	$[q_0,q_2]$
		$[q_1]$	$[q_3]$	$[\Phi]$
		$[q_2]$	$[\Phi]$	$[q_3]$
终止		$[q_3]$	$[q_3]$	$[q_3]$
	√	$[q_0,q_1]$	$[q_0,q_1,q_3]$	$[q_0,q_2]$
	√	$[q_0,q_2]$	$[q_0,q_1]$	$[q_0,q_2,q_3]$
终止		$[q_0,q_3]$	$[q_0,q_1,q_3]$	$[q_0,q_2,q_3]$
		$[q_1,q_2]$	$[q_3]$	$[q_3]$
终止		$[q_1,q_3]$	$[q_3]$	$[q_3]$
终止		$[q_2,q_3]$	$[q_3]$	$[q_3]$
		$[q_0,q_1,q_2]$	$[q_0,q_1,q_3]$	$[q_0,q_2,q_3]$
终止	√	$[q_0,q_1,q_3]$	$[q_0,q_1,q_3]$	$[q_0,q_2,q_3]$
终止	√	$[q_0,q_2,q_3]$	$[q_0,q_1,q_3]$	$[q_0,q_2,q_3]$
终止		$[q_1,q_2,q_3]$	$[q_3]$	$[q_3]$
终止		$[q_0,q_1,q_2,q_3]$	$[q_0,q_1,q_3]$	$[q_0,q_2,q_3]$
		$[\Phi]$	$[\Phi]$	$[\Phi]$

3.3.3 NFA与DFA等价

- **不可达状态**(inaccessible state)：不存在从$[q_0]$对应的顶点出发，到达该状态对应的顶点的路。我们称此状态从开始状态是不可达的。
- 表3-7中，所有标记“√”的状态是从开始状态可达的，其他是不可达的——无用的。

3.3.3 NFA与DFA等价

- 构造给定NFA等价的DFA策略
  - 先只把开始状态$[q_0]$填入表的状态列中，如果表中所列的状态列有未处理的，则任选一个未处理的状态$[q_1，q_2，…，q_n]$，对Σ中的每个字符a，计算$\delta([q_1，q_2，…，q_n]，a)$，并填入相应的表项中，如果$\delta([q_1，q_2，…，q_n]，a)$在表的状态列未出现过，则将它填入表的状态列。如此重复下去，直到表的状态列中不存在未处理的状态。

3.3.3 NFA与DFA等价

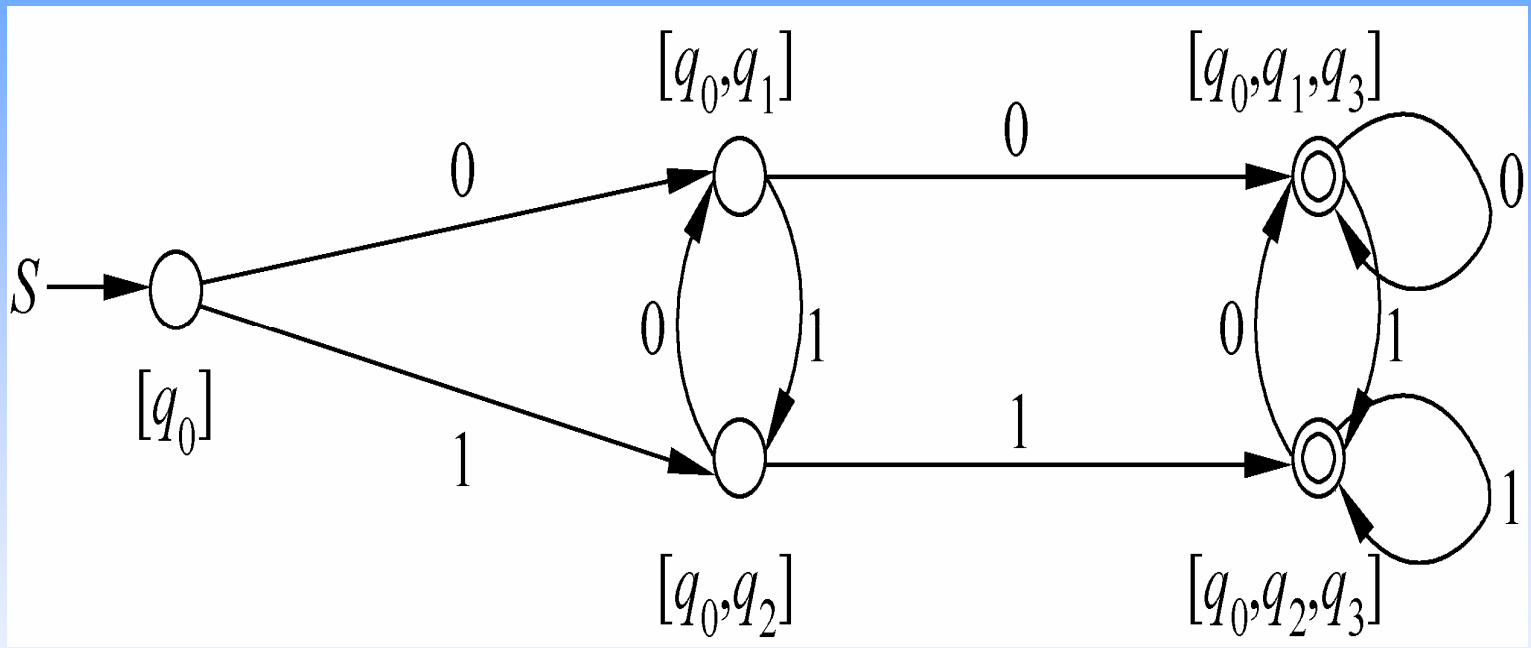
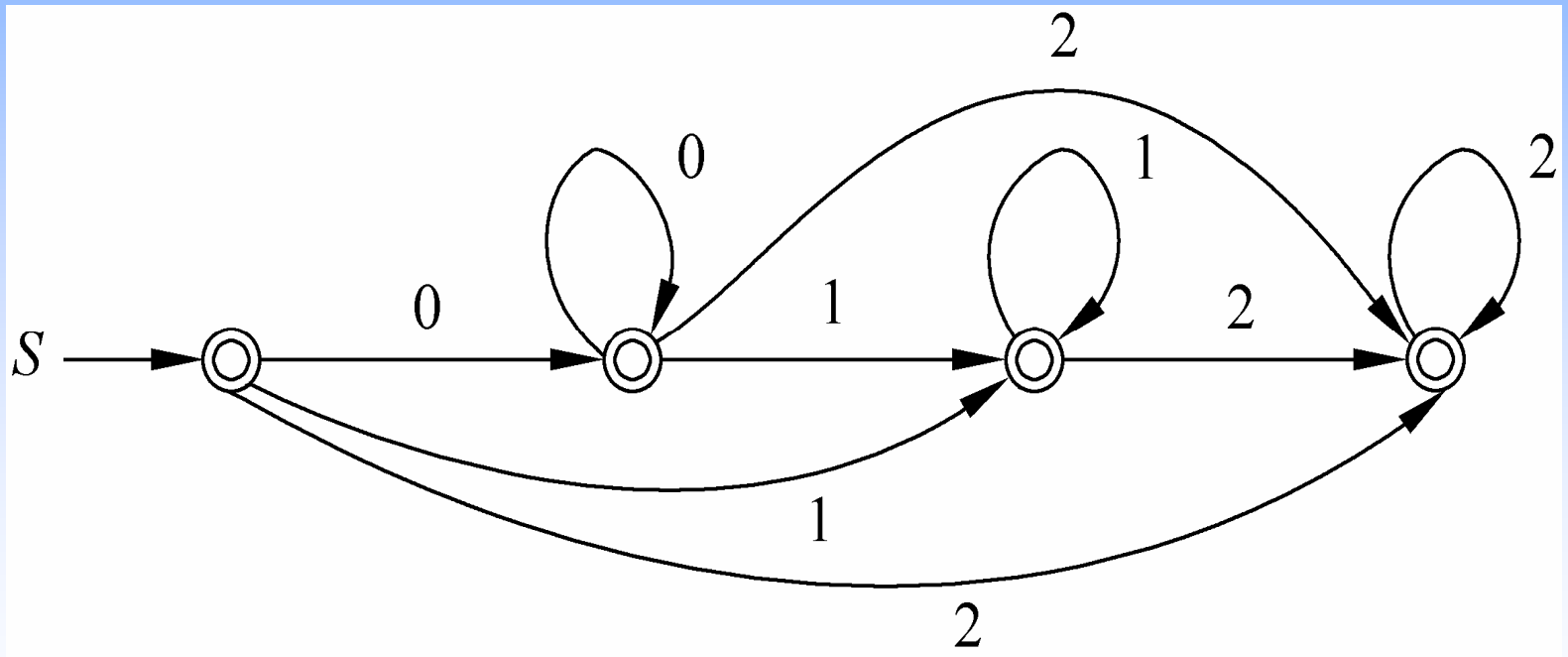


图 3-11　图3-9所示NFA的等价DFA

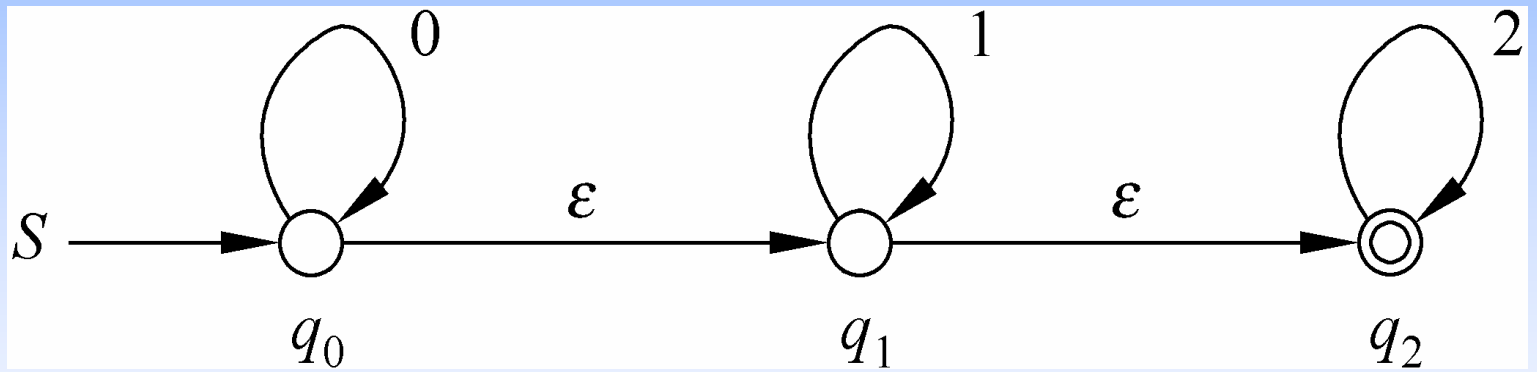
3.4 带空移动的有穷状态自动机

- 接受语言$\{0^n1^m2^k | n，m，k \geqslant 0\}$的NFA



3.4 带空移动的有穷状态自动机

- 接受语言$\{0^n1^m2^k|n，m，k\geqslant 0\}$的NFA是否可以构造成下图所示的“ε-NFA”？



其构造显然比图1-13所示的NFA更容易。当然还希望它们是等价的。

3.4 带空移动的有穷状态自动机

- **带空移动的不确定的有穷状态自动机(non-deterministic finite automaton with ε-moves, ε-NFA)**

M=(Q，Σ，δ，q_0，F)，Q、Σ、q_0、F的意义同DFA。

δ：$Q\times(\Sigma\cup\{\varepsilon\})\rightarrow 2^Q$

3.4 带空移动的有穷状态自动机

- 非空移动
  - $\forall(q, a) \in Q \times \Sigma$
  - $\delta(q, a) = \{p_1, p_2, \ldots, p_m\}$
  - 表示M在状态q读入字符a，可以选择地将状态变成p_1、p_2、…或者p_m，并将读头向右移动一个带方格而指向输入字符串的下一个字符。

3.4 带空移动的有穷状态自动机

- 空移动
  - $\forall q \in Q$
  - $\delta(q, \varepsilon) = \{p_1, p_2, \ldots, p_m\}$
  - 表示M在状态q不读入任何字符，可以选择地将状态变成p_1、p_2、…或者p_m。也可以叫做M在状态q做一个空移动(又可以称为 ε 移动)，并且选择地将状态变成p_1、p_2、…或者p_m。

3.4 带空移动的有穷状态自动机

- 进一步扩充 δ 的定义域： δ：
  $2^Q \times \Sigma^* \to 2^Q$。对任意的$P \subseteq Q$，$w \in \Sigma^*$ 。
- 对任意的$q \in Q$，$w \in \Sigma^*$，$a \in \Sigma$。

(1) ε **-CLOSURE(q)={p|从q到p有一条标记为** ε **的路}。**

$$(2)\varepsilon - CLOSURE\quad(P) = \bigcup_{p \in P} \varepsilon - CLOSURE\quad(p)$$

3.4 带空移动的有穷状态自动机

$$(3)\hat{\delta}(q,\varepsilon)=\varepsilon-CLOSURE(q)$$

$$(4)\hat{\delta}(q,wa)=\varepsilon-CLOSURE(P)$$

$$P=\{p\mid\exists r\in\hat{\delta}(q,w)\text{使得}p\in\delta(r,a)\}$$

$$=\bigcup_{r\in\hat{\delta}(r,w)}\delta(r,a)$$

3.4 带空移动的有穷状态自动机

- 进一步扩展移动函数：$2^Q \times \Sigma \to 2^Q$。
- 对任意$(P, a) \in 2^Q \times \Sigma$。

$$(5)\delta(P,a) = \bigcup_{q \in P} \delta(q,a)$$

$$(6)\hat{\delta}(P,w) = \bigcup_{q \in P} \hat{\delta}(q,w)$$

3.4 带空移动的有穷状态自动机

- 在 ε -NFA中，对任意$a \in \Sigma$，$q \in Q$，

$$\hat{\delta}(q,a) \neq \delta(q,a)$$

需要严格区分。

图3-14 所示 ε -NFA

状态	δ				$\hat{\delta}$			
	ε	0	1	2	ε	0	1	2
q_0	$\{q_1\}$	$\{q_0\}$	Φ	Φ	$\{q_0,q_1,q_2\}$	$\{q_0,q_1,q_2\}$	$\{q_1,q_2\}$	$\{q_2\}$
q_1	$\{q_2\}$	Φ	$\{q_1\}$	Φ	$\{q_1,q_2\}$	Φ	$\{q_1,q_2\}$	$\{q_2\}$
q_2	Φ	Φ	Φ	$\{q_2\}$	$\{q_2\}$	Φ	Φ	$\{q_2\}$

3.4 带空移动的有穷状态自动机

- **M接受(识别)的语言**

对于$\forall \mathbf{x} \in \Sigma^*$，仅当

$$\hat{\delta}(q_0, x) \bigcap F \neq \Phi$$

时，称**x**被**M**接受。

$$L(M) = \{x \mid x \in \Sigma^* 并且 \hat{\delta}(q_0, x) \bigcap F \neq \Phi\}$$

3.4 带空移动的有穷状态自动机

定理 3-2 ε **-NFA与NFA等价。**

证明：设有 ε -NFA $M_1=(Q, \Sigma, \delta_1, q_0, F)$

(1) 构造与之等价的NFA M_2。

取NFA $M_2=(Q, \Sigma, \delta_2, q_0, F_2)$

$$F_2=\begin{cases} F\cup\{q_0\} & \text{如果} F\cap \varepsilon\text{-CLOSURE}(q_0)\neq \Phi \\ F & \text{如果} F\cap \varepsilon\text{-CLOSURE}(q_0)= \Phi \end{cases}$$

$$\delta_2(q,a)=\hat{\delta}_1(q,a)$$

3.4 带空移动的有穷状态自动机

- **(2)** 施归纳于**|x|**，证明对$\forall \mathbf{x} \in \Sigma^+$。

$$\delta_2(q,x) = \hat{\delta}_1(q,x)$$

$$\begin{aligned}
&\delta_2(\mathbf{q_0},\ \mathbf{x}) = \delta_2(\mathbf{q_0},\ \mathbf{wa}) \\
&= \delta_2(\delta_2(\mathbf{q_0},\ \mathbf{w}),\ \mathbf{a}) \\
&= \delta_2((\mathbf{q_0},\ \mathbf{w}),\ \mathbf{a})
\end{aligned}$$

3.4 带空移动的有穷状态自动机

$$= \bigcup_{q \in \hat{\delta}_1(q_0, w)} \delta_2(q, a)$$

$$= \bigcup_{q \in \hat{\delta}_1(q_0, w)} \hat{\delta}_1(q, a)$$

$$= \varepsilon - CLOSURE \left(\bigcup_{q \in \hat{\delta}_1(q_0, w)} \delta_1(q, a) \right)$$

$$= \varepsilon - CLOSURE \left(\{ p \mid \exists q \in \hat{\delta}_1(q_0, w),\ p \in \delta_1(q, a \right.$$

$$= \hat{\delta}_1(q_0, wa)$$

$$= \hat{\delta}_1(q_0, x)$$

3.4 带空移动的有穷状态自动机

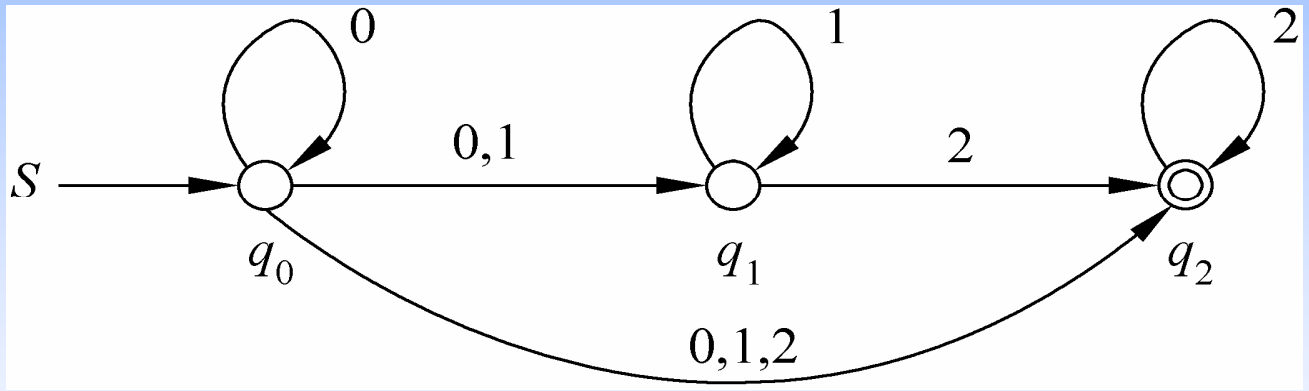
- **(3)** 证明, 对$\forall \mathbf{x} \in \Sigma^+$， $\delta_2(\mathbf{q_0}, \mathbf{x}) \cap \mathbf{F_2} \neq \Phi$ 的充分必要条件是

$$\hat{\delta}_1(q, x) \bigcap F \neq \Phi$$

- (4) 证明 $\varepsilon \in \mathbf{L(M_1)} \Leftrightarrow \varepsilon \in \mathbf{L(M_2)}$ 。

3.4 带空移动的有穷状态自动机

- 例3-4 求与图3-14所示 ε -NFA等价的NFA。



3.5 FA是正则语言的识别器

3.5.1 FA与右线性文法

- **DFA M=(Q，Σ，δ，q_0，F)**，处理句子$a_1a_2\ldots a_n$的特性。

⑴ **M**按照句子$a_1a_2\ldots a_n$中字符的出现顺序，从开始状态q_0开始，依次处理字符a_1、a_2、…、a_n，在这个处理过程中，每处理一的字符进入一个状态，最后停止在某个终止状态。

3.5.1 FA与右线性文法

⑵ 它每次处理且仅处理一个字符：第i步处理输入字符a_i。

⑶ 对应于使用 $\delta(q, a)=p$的状态转移函数的处理，相当于是在状态q完成对a的处理，然后由p接下去实现对后续字符的处理。

⑷ 当 $\delta(q, a)=p \in F$，且a是输入串的最后一个字符时，M完成对此输入串的处理。

3.5.1 FA与右线性文法

$A_0 \Rightarrow a_1A_1$ 对应产生式$A_0 \rightarrow a_1A_1$

$\Rightarrow a_1a_2A_2$ 对应产生式$A_1 \rightarrow a_2A_2$

…

$\Rightarrow a_1a_2\ldots a_{n-1}A_{n-1}$ 对应产生式$A_{n-2} \rightarrow a_{n-1}A_{n-1}$

$\Rightarrow a_1a_2\ldots a_{n-1}a_n$ 对应产生式$A_{n-1} \rightarrow a_n$

3.5.1 FA与右线性文法

$q_0\ a_1a_2\ldots a_{n-1}a_n$

$\vdash a_1q_1\ a_2\ldots a_{n-1}a_n$　　对应 $\delta(q_0, a_1)=q_1$

$\vdash a_1a_2q_2\ldots a_{n-1}a_n$　　对应 $\delta(q_1, a_2)=q_2$

……

$\vdash a_1a_2\ldots a_{n-1}q_{n-1}a_n$　　对应 $\delta(q_{n-2}, a_{n-1})=q_{n-1}$

$\vdash a_1a_2\ldots a_{n-1}a_nq_n$　　对应 $\delta(q_{n-1}, a_n)=q_n$

3.5.1 FA与右线性文法

- 其中q_n为M的终止状态。考虑根据a_1、a_2、…、a_n，让A_0与q_0对应、A_1与q_1对应、A_2与q_2对应、…、A_{n-2}与q_{n-2}对应、A_{n-1}与q_{n-1}对应。这样，就有希望得到满足定理2-4-1的正则文法的推导与DFA的互相模拟的方式。

3.5.1 FA与右线性文法

定理 3-3 FA接受的语言是正则语言。

证明：

(1) 构造。

基本思想是让**RG**的派生对应**DFA**的移动。

设**DFA M=(Q，Σ，δ，q_0，F)**，

取右线性文法 **G=(Q，Σ，P，q_0)**，

$P=\{q\to ap \mid \delta(q，a)=p\}\cup\{q\to a \mid \delta(q，a)=p\in F\}$

3.5.1 FA与右线性文法

（2）证明 L(G)=L(M)-{ ε }。

对于$a_1a_2\ldots a_{n-1}a_n \in \Sigma^+$,

$q_0 \Rightarrow^+ a_1a_2\ldots a_{n-1}a_n$

$\Leftrightarrow q_0 \to a_1q_1$，$q_1 \to a_2q_2$，…，

$q_{n-2} \to a_{n-1}q_{n-1}$，$q_{n-1} \to a_n \in P$

$\Leftrightarrow \delta(q_0, a_1)=q_1$，$\delta(q_1, a_2)=q_2$，…，$\delta(q_{n-2}, a_{n-1})=q_{n-1}$，$\delta(q_{n-1}, a_n)=q_n$，且 $q_n \in F$

$\Leftrightarrow \delta(q_0, a_1a_2\ldots a_{n-1}a_n)= q_n \in F$

$\Leftrightarrow a_1a_2\ldots a_{n-1}a_n \in L(M)$

3.5.1 FA与右线性文法

（3）关于 ε 句子。

如果$q_0 \notin F$，则 $\varepsilon \notin L(M)$，$L(G)=L(M)$。

如果$q_0 \in F$，则由定理2-6和定理2-7，存在正则文法 G'，使得 $L(G')=L(G) \cup \{\varepsilon\}=L(M)$。

综上所述，对于任意DFA M，存在正则文法G，使得$L(G)=L(M)$。

定理得证。

3.5.1 FA与右线性文法

- 例 3-9 与图3-8 所给DFA等价的正则文法

$q_s \rightarrow 0|0q_0|1q_1$

$q_0 \rightarrow 0|0q_0|1q_1$

$q_1 \rightarrow 0q_2|1|1q_0$

$q_2 \rightarrow 0q_1|1q_2$

3.5.1 FA与右线性文法

- 与图3-7 所给的DFA等价的正则文法

$q_0 \rightarrow 0q_1|1q_t|2q_t$

$q_1 \rightarrow 0q_1|1q_2|2q_t$

$q_2 \rightarrow 0q_t|1q_2|2q_3|2$

$q_3 \rightarrow 0q_t|1q_t|2q_3|2$

$q_t \rightarrow 0q_t|1q_t|2q_t$

3.5.1 FA与右线性文法

定理3-4 正则语言可以由**FA**接受。

证明：

（1）构造。

基本思想：让**FA**模拟**RG**的派生。

设**G=(V，T，P，S)**，且 $\varepsilon \notin L(G)$，

取**FA M=(V∪{Z}，T，δ，S，{Z})**，

Z∉V。

3.5.1 FA与右线性文法

- 对$\forall(a，A)\in T\times V$

$$\delta(A，a)=\begin{cases}\{B|A\to aB\in P\}\cup\{Z\} & \text{如果}A\to a\in P\\ \{B|A\to aB\in P\} & \text{如果}A\to a\notin P\end{cases}$$

用$B\in\delta(a，A)$与产生式$A\to aB$对应

用$Z\in\delta(a，A)$与产生式$A\to a$对应。

3.5.1 FA与右线性文法

（2）证明L(M)=L(G)

对于$a_1a_2\ldots a_{n-1}a_n \in T^+$,

$a_1a_2\ldots a_{n-1}a_n \in L(G) \Leftrightarrow S \Rightarrow^+ a_1a_2\ldots a_{n-1}a_n$

$\Leftrightarrow S \Rightarrow a_1A_1 \Rightarrow a_1a_2A_2 \Rightarrow \ldots$

$\Rightarrow a_1a_2\ldots a_{n-1}A_{n-1} \Rightarrow a_1a_2\ldots a_{n-1}a_n$

$\Leftrightarrow S \rightarrow a_1A_1$，$A_1 \rightarrow a_2A_2$，…，

$A_{n-2} \rightarrow a_{n-1}A_{n-1}$，$A_{n-1} \rightarrow a_n \in P$

3.5.1 FA与右线性文法

$\Leftrightarrow A_1 \in \delta(S, a_1)$，$A_2 \in \delta(A_1, a_2)$，…，

$A_{n-1} \in \delta(A_{n-2}, a_{n-1})$，$Z \in \delta(A_{n-1}, a_n)$

$\Leftrightarrow Z \in \delta(S, a_1a_2\ldots a_{n-1}a_n)$

$\Leftrightarrow a_1a_2\ldots a_{n-1}a_n \in L(M)$

对于 ε，按照定理2-5和定理2-6处理。

3.5.1 FA与右线性文法

- 例 3-10 构造与所给正则文法等价的FA：

G_1：$E \rightarrow 0A|1B$

$A \rightarrow 1|1C$

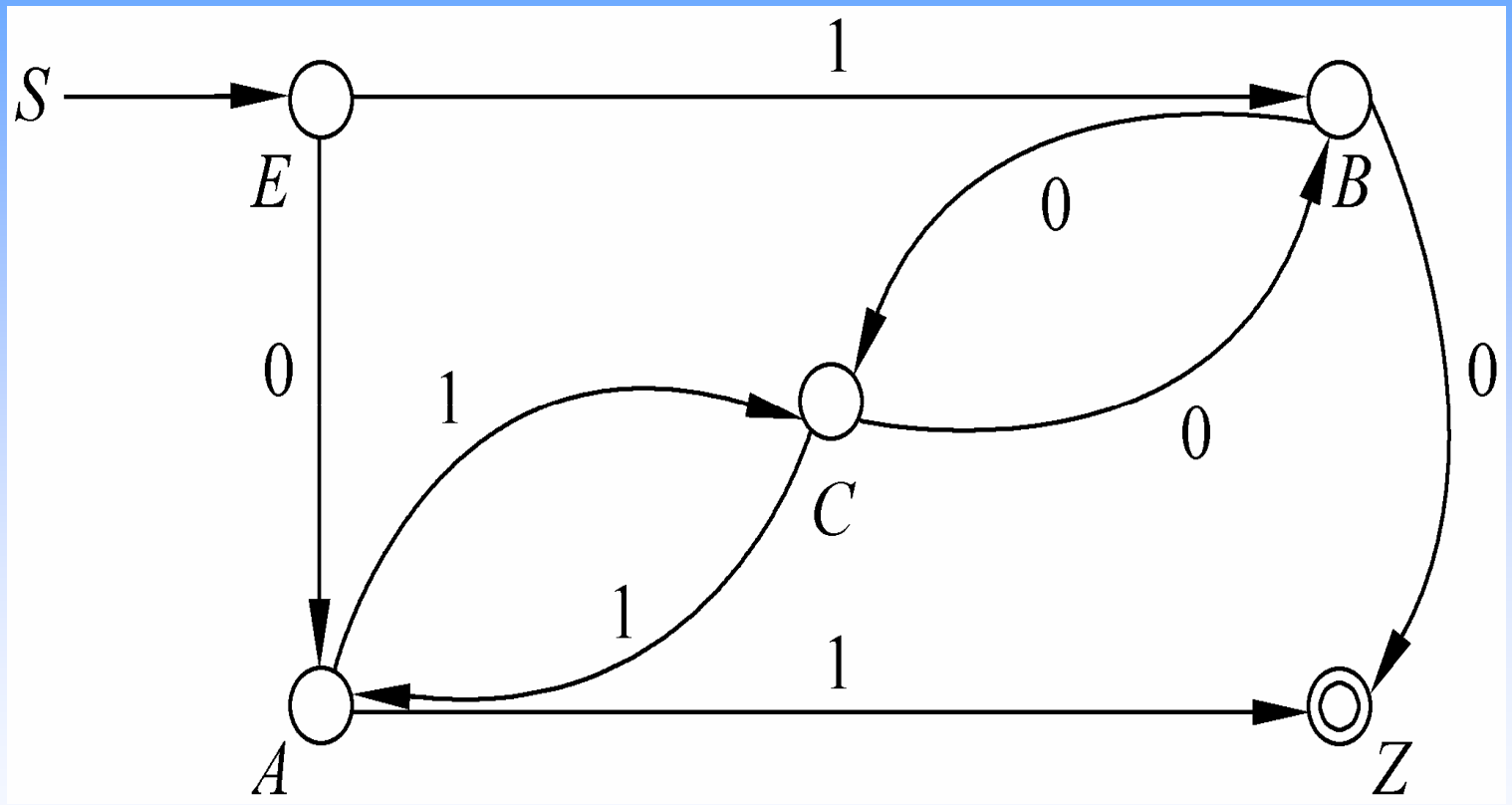
$B \rightarrow 0|0C$

$C \rightarrow 0B|1A$

3.5.1 FA与右线性文法

δ (E，0)={A}	对应E→0A
δ (E，1)={B}	对应E→1B
δ (A，1)={Z,C}	对应A→1|1C
δ (B，0)={Z,C}	对应B→0|0C
δ (C，0)={B}	对应C→0B
δ (C，1)={A}	对应C→1A

3.5.1 FA与右线性文法



3.5.1 FA与右线性文法

推论3-1 FA与正则文法等价。

证明：根据定理3-3和定理3-4即可得到。

3.5.1 FA与左线性文法

- 按照推导来说，句子$a_1a_2\ldots a_{n-1}a_n$中的字符被推导出的先后顺序正好与它们在句子中出现的顺序相反；而按照归约来看，它们被归约成语法变量的顺序则正好与它们在句子中出现的顺序相同：a_1，a_2，…，a_{n-1}，a_n。可见，归约过程与FA处理句子字符的顺序是一致的，所以可考虑依照“归约”来研究FA的构造。

3.5.1 FA与左线性文法

- 对于形如**A→a**的产生式：在推导中，一旦使用了这样的产生式，句型就变成了句子，而且**a**是该句子的第一个字符；按“归约”理解，对句子的第**1**个字符，根据形如**A→a**的产生式进行归约。对应到**FA**中，**FA**从开始状态出发，读到句子的第一个字符**a**，应将它“归约”为**A**。我们如果考虑用语法变量对应**FA**的状态，那么，此时我们需要引入一个开始状态，比如说：**Z**。这样，对应形如**A→a**的产生式，可以定义**A**∈ δ **(Z，a)**。

3.5.1 FA与左线性文法

- 按照上面的分析，对应于形如**A→Ba**的产生式：**FA**应该在状态**B**读入**a**时，将状态转换到**A**。也可以理解为：在状态**B**，**FA**已经将当前句子的、处理过的前缀“归约”成了**B**，在此时它读入**a**时，要将**Ba**归约成**A**，因此，它进入状态**A**。

3.5.1 FA与左线性文法

- 按照“归约”的说法，如果一个句子是文法G产生的语言的合法句子，它最终应该被归约成文法G的开始符号。所以，G的开始符号对应的状态就是相应的FA的终止状态。
- 如何解决好开始符号只有一个，而DFA的终止状态可以有多个的问题。

3.5.1 FA与左线性文法

- 例如：对文法

G_2：$E \rightarrow A0|B1$

$A \rightarrow 1|C1$

$B \rightarrow 0|C0$

$C \rightarrow B0|A1$

- 对应

$\delta(A, 0)=\{E\}$

$\delta(B, 1)=\{E\}$

$\delta(Z, 1)=\{A\}$

$\delta(C, 1)=\{A\}$

$\delta(Z, 0)=\{B\}$

$\delta(C, 0)=\{B\}$

$\delta(B, 0)=\{C\}$

$\delta(A, 1)=\{C\}$

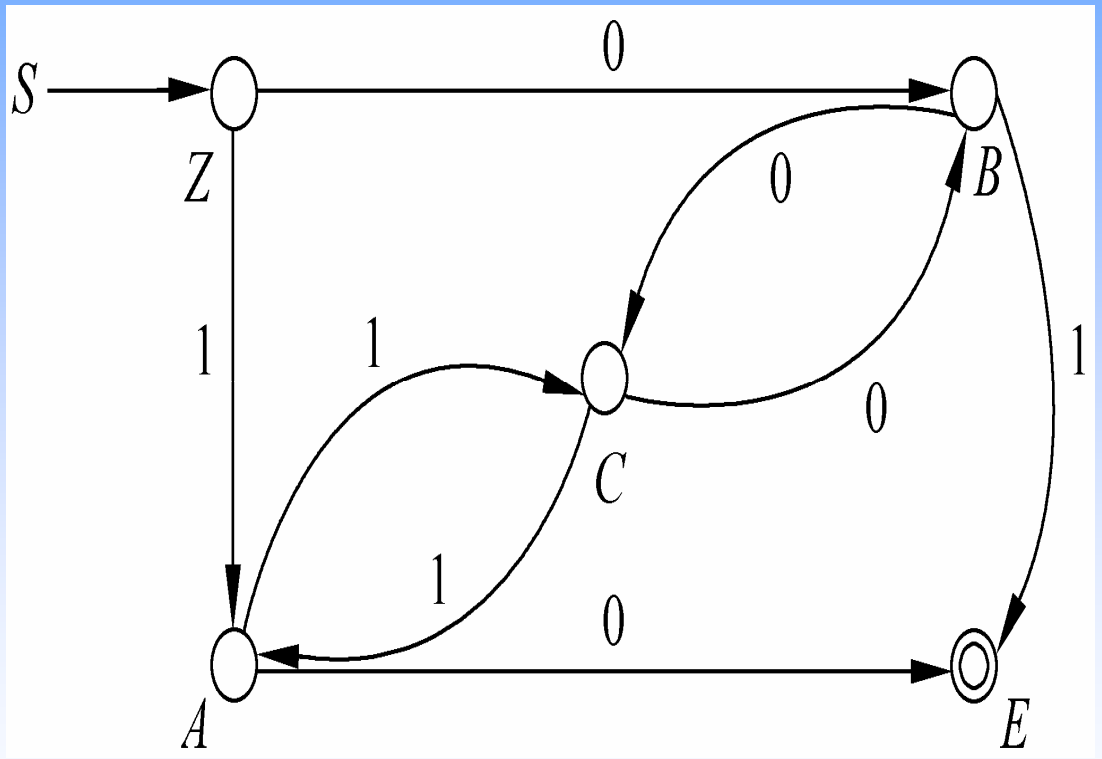
3.5.1 FA与左线性文法

G_2：E→A0|B1

A→1|C1

B→0|C0

C→B0|A1



3.5.1 FA与左线性文法

- **DFA (**的状态转移图**)**作如下“预处理”：

  ⑴ 删除**DFA**的陷阱状态**(**包括与之相关的弧**)**；

  ⑵ 在图中加一个识别状态；

  ⑶ “复制”一条原来到达终止状态的弧，使它从原来的起点出发，到达新添加的识别状态。

3.5.1 FA与左线性文法

⑴ 如果 δ (A，a)=B，则有产生式B→Aa；

⑵ 如果 δ (A，a)=B，且A是开始状态，则有产生式B→a。

定理3-5 左线性文法与FA等价。

3.6 FA的一些变形

3.6.1 双向有穷状态自动机

- 确定的双向有穷状态自动机(two-way deterministic finite automaton，2DFA)

$$M=(Q，\Sigma，\delta，q_0，F)$$

Q、Σ、q_0、F的意义同DFA。

3.6.1 双向有穷状态自动机

- δ：Q×Σ→Q×{L，R，S}，对∀(q，a)∈Q×Σ
  - 如果δ(q，a)= {p，L}，它表示M在状态q读入字符a，将状态变成p，并将读头向左移动一个带方格而指向输入字符串的前一个字符。
  - 如果δ(q，a)= {p，R}，它表示M在状态q读入字符a，将状态变成p，并将读头向右移动一个带方格而指向输入字符串中下一个字符。
  - 如果δ(q，a)= {p，S}，它表示M在状态q读入字符a，将状态变成p，读头保持在原位不动。

3.6.1 双向有穷状态自动机

- M接受的语言

  $L(M)=\{x \mid q_0x \vdash^* xp$且$p \in F\}$ 是2DFA接受的语言。

定理3-6　2DFA与FA等价。

3.6.1 双向有穷状态自动机

- 不确定的双向有穷状态自动机(two-way nondeterministic finite automaton，2NFA)

$$M=(Q，\ \Sigma，\ \delta，\ q_0，\ F)$$

Q、Σ、q_0、F的意义同DFA。

δ：$Q\times\Sigma\rightarrow 2^{Q\times\{L，\ R，\ S\}}$ 。

3.6.1 双向有穷状态自动机

- $\delta(q, a)=\{(p_1, D_1)(p_2, D_2)\ldots, (p_m, D_m)\}$ 表示M在状态q读入字符a，可以选择地将状态变成p_1同时按D_1实现对读头的移动、或者将状态变成p_2同时按D_2实现对读头的移动、……或者将状态变成p_m同时按D_m实现对读头的移动。其中D_1，D_2，…，$D_m \in \{L, R, S\}$，表示的意义与2DFA的定义表示的意义相同。

3.6.1 双向有穷状态自动机

定理3-7　2NFA与FA等价。

3.6.2 带输出的FA

- **Moore机**

$$M=(Q,\ \Sigma,\ \Delta,\ \delta,\ \lambda,\ q_0)$$

Q、Σ、q_0、δ 的意义同**DFA**。

Δ ——**输出字母表**(output alphabet)。

λ：$Q\rightarrow\Delta$ 为输出函数。对$\forall q\in Q$，$\lambda(q)=a$ 表示**M**在状态**q**时输出**a**。

3.6.2 带输出的FA

- 对于$\forall a_1a_2\ldots a_{n-1}a_n \in \Sigma^*$，如果

$\delta(q_0, a_1)=q_1$，$\delta(q_1, a_2)=q_2$，…，$\delta(q_{n-2}, a_{n-1})=q_{n-1}$，$\delta(q_{n-1}, a_n)=q_n$，

则M的输出为

$$\lambda(q_0, a_1)\,\lambda(q_1, a_2)\ldots\lambda(q_{n-1}, a_n)$$

3.6.2 带输出的FA

- **Mealy机**

$$M=(Q,\ \Sigma,\ \Delta,\ \delta,\ \lambda,\ q_0)$$

Δ——输出字母表。

λ：$Q\times\Sigma\rightarrow\Delta$为输出函数。对$\forall(q, a)\in Q\times\Sigma$，$\lambda(q, a)=d$表示M在状态q读入字符a时输出d。

3.6.2 带输出的FA

- 对于$\forall a_1a_2\ldots a_{n-1}a_n \in \Sigma^*$，M的输出串为：

$\lambda(q_0, a_1)\ \lambda(\delta(q_0, a_1), a_2)\ \ldots\ \lambda((\ldots\delta(\delta(q_0, a_1), a_2)\ldots), a_n)$

设 $\delta(q_0, a_1)=q_1$，$\delta(q_1, a_2)=q_2$，…，$\delta(q_{n-2}, a_{n-1})=q_{n-1}$，$\delta(q_{n-1}, a_n)=q_n$，

则M的输出可以表示成：

$$\lambda(q_0, a_1)\ \lambda(q_1, a_2)\ \ldots\ \lambda(q_{n-1}, a_n)$$

3.6.2 带输出的FA

- **Moore**机处理该串时每经过一个状态，就输出一个字符：输出字符和状态一一对应；
- **Mealy**机处理该串时的每一个移动输出一个字符：输出字符和移动一一对应。

3.6.2 带输出的FA

- 如果对于$\forall x \in \Sigma^*$，$T_1(x)=\lambda_1(q_0)T_2(x)$，则Moore机$M_1=(Q_1, \Sigma, \Delta, \delta_1, \lambda_1, q_{01})$与Mealy机$M_2=(Q_2, \Sigma, \Delta, \delta_2, \lambda_2, q_{02})$，是等价的，其中，$T_1(x)$和$T_2(x)$分别表示$M_1$和$M_2$关于x的输出。

定理3-8 Moore机与Mealy机等价。

3.7 小结

本章讨论正则语言的识别器——FA。包括DFA、NFA、 ε-NFA。RG与FA的等价性。简单介绍带输出的FA和双向FA。

(1) FA M是一个五元组，M=(Q，Σ，δ，q_0，F)，它可以用状态转移图表示。

(2) M接受的语言为{x| x∈Σ^*且δ(q，x)∈F}。如果$L(M_1)=L(M_2)$，则称M_1与M_2等价。

(3) FA的状态具有有穷的存储功能。这一特性可以用来构造接受一个给定语言的FA。

3.7 小结

(4) NFA允许M在一个状态下读入一个字符时选择地进入某一个状态，对于$\forall x \in \Sigma^*$，如果$\delta(q_0, x) \cap F \neq \Phi$，则称x被M接受，如果$\delta(q_0, x) \cap F = \Phi$，则称M不接受x。M接受的语言$L(M)=\{x \mid x \in \Sigma^* \text{且} \delta(q_0, x) \cap F \neq \Phi\}$。

3.7 小结

(5) ε-NFA是在NFA的基础上，允许直接根据当前状态变换到新的状态而不考虑输入带上的符号。对$\forall q \in Q$，$\delta(q, \varepsilon) = \{p_1, p_2, \ldots, p_m\}$表示M在状态q不读入任何字符，可以选择地将状态变成p_1、p_2、…或者p_m。这叫做M在状态q做一个空移动。

(6) NFA与DFA等价，ε-NFA与NFA等价，统称它们为FA。

3.7 小结

(7) 根据需要，可以在FA中设置一种特殊的状态——陷阱状态。但是，不可达状态却是应该无条件地删除的。

(8) FA是正则语言的识别模型。分别按照对推导和归约的模拟，可以证明FA和左线性文法、右线性文法等价。

3.7 小结

⑼ **2DFA是FA的又一种变形，它不仅允许读头向前移动，还允许读头向后移动。通过这种扩充，2DFA仍然与FA等价。**

⑽ **Moore机和Mealy机是两种等价的带输出的FA，Moore机根据状态决定输出字符，Mealy机根据移动决定输出字符。**

第4章 正则表达式

- 正则文法擅长语言的产生，有穷状态自动机擅长语言的识别。
- 本章讨论正则语言的正则表达式描述。它在对正则语言的表达上具有特殊的优势，为正则语言的计算机处理提供了方便条件。
  - 简洁、更接近语言的集合表示和语言的计算机表示等。

第4章 正则表达式

- 主要内容
  - 典型RE的构造。
  - 与RE等价FA的构造方法。
  - 与DFA等价的RE的构造。
- 重点
  - RE的概念。
  - RE与DFA的等价性。
- 难点：RE与DFA的等价性证明。

4.1 启示

产生语言$\{a^nb^mc^k|n，m，k\geqslant 1\}\cup$
$\{a^ic^nbxa^m|i\geqslant 0, n\geqslant 1, m\geqslant 2, x$为d和e组成的串$\}$
的正则文法为

$A\rightarrow aA|aB|cE$

$B\rightarrow bB|bC$

$C\rightarrow cC|c$

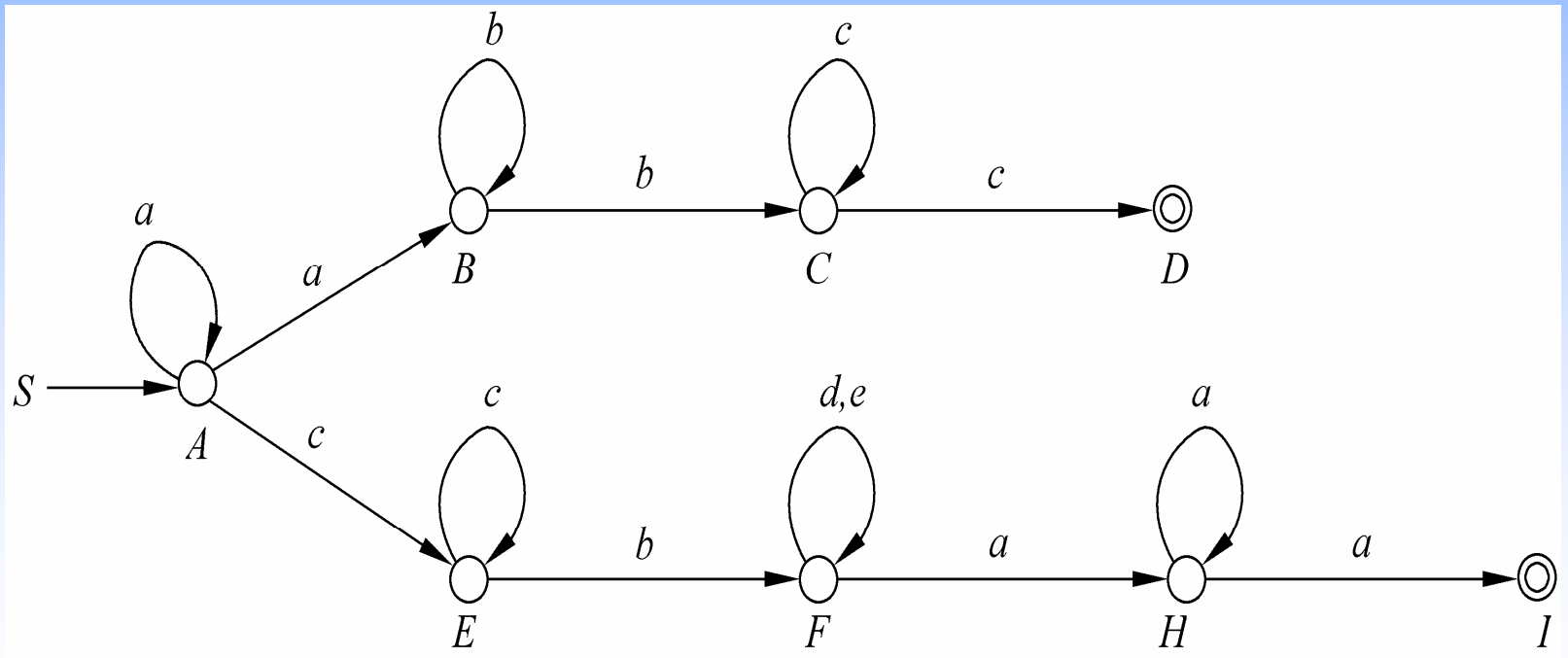
$E\rightarrow cE|bF$

$F\rightarrow dF|eF|aH$

$H\rightarrow aH|a$

4.1 启示

- 接受此语言的NFA M



4.1 启示

- 计算集合 **set(q)**

set(A)={a^n|n≥0}={a}*

set(B)= set(A){a}{b^n|n≥0}

={$a^n ab^m$|m，n≥0}

={a}*{a}{b}*={a}$^+${b}*

set(C)= set(B){b}{c}*

={a}*{a}{b}*{b}{c}*={a}$^+${b}$^+${c}*

set(D)= set(C) {c}={a}$^+${b}$^+${c}*{c}

={a}$^+${b}$^+${c}$^+$

4.1 启示

$set(E)= set(A)\{c\}\{c\}^*$

$=\{a\}^*\{c\}\{c\}^*=\{a\}^*\{c\}^+$

$set(F)= set(E)\{b\}\{d，e\}^*=\{a\}^*\{c\}^+\{b\}\{d，e\}^*$

$set(H)= set(F)\{a\}\{a\}^*=\{a\}^*\{c\}^+\{d，e\}^*\{a\}\{a\}^*$

$=\{a\}^*\{c\}^+\{d，e\}^*\{a\}^+$

$set(I)= set(H)\{a\}=\{a\}^*\{c\}^+\{d，e\}^*\{a\}^+\{a\}$

$L(M)= set(D)\cup set(H)$

$= \{a\}^+\{b\}^+\{c\}^+\cup\{a\}^*\{c\}^+\{d，e\}^*\{a\}^+\{a\}$

4.1 启示

根据集合运算的定义，

{d，e}={d}∪{e}。

从而，

{d，e}*=({d}∪{e})*。

这样可以将**L(M)**写成如下形式：

L(M)= {a}$^+${b}$^+${c}$^+$∪{a}*{c}$^+$({d}∪{e})*{a}$^+${a}

记作：

a$^+$b$^+$c$^+$+a*c$^+$(d+e)*a$^+$a= aa*bb*cc*+a*cc*(d+e)* aaa*

4.2 RE的形式定义

- **正则表达式(regular expression，RE)**

(1) Φ是Σ上的RE，它表示语言Φ；

(2) ε是Σ上的RE，它表示语言$\{\varepsilon\}$；

(3) 对于$\forall a \in \Sigma$，a是Σ上的RE，它表示语言{a}；

4.2 RE的形式定义

⑷ 如果r和s分别是Σ上表示语言R和S的RE，则：

r与s的“和” (r+s)是Σ上的RE，(r+s)表达的语言为R∪S；

r与s的“乘积” (rs)是Σ上的RE，(rs)表达的语言为RS；

r的克林闭包(r^*)是Σ上的RE，(r^*)表达的语言为R^*。

⑸ 只有满足⑴、⑵、⑶、⑷的才是Σ上的RE。

4.2 RE的形式定义

- **例 4-1 设Σ={0，1}**

  ⑴ **0，表示语言{0}；**

  ⑵ **1，表示语言{1}；**

  ⑶ **(0+1)，表示语言{0，1}；**

  ⑷ **(01)，表示语言{01}；**

  ⑸ **((0+1)*)，表示语言{0，1}*；**

  ⑹ **((00)((00)*))，表示语言{00}{00}*；**

4.2 RE的形式定义

⑺ $((((0+1)^*)(0+1))((0+1)^*))$，表示语言$\{0，1\}^+$；

⑻ $((((0+1)^*)000)((0+1)^*))$，表示{0，1}上的至少含有3个连续0的串组成的语言；

⑼ $((((0+1)^*)0)1)$，表示所有以01结尾的0、1字符串组成的语言；

⑽ $(1(((0+1)^*)0))$，表示所有以1开头，并且以0结尾的0、1字符串组成的语言。

4.2 RE的形式定义

- 约定

⑴ r的正闭包r^+表示r与(r^*)的乘积以及(r^*)与r的乘积：

$$r^+=(r(r^*))=((r^*)r)$$

⑵ 闭包运算的优先级最高，乘运算的优先级次之，加运算“+”的优先级最低。所以，在意义明确时，可以省略其中某些括号。

$$((((0+1)^*)000)((0+1)^*))=(0+1)^*000(0+1)^*$$

4.2 RE的形式定义

$$((((0+1)^*)(0+1))((0+1)^*))=(0+1)^*(0+1)(0+1)^*$$

⑶ 在意义明确时，RE r表示的语言记为L(r)，也可以直接地记为r。

⑷ 加、乘、闭包运算均执行左结合规则。

4.2 RE的形式定义

- **相等(equivalence)**
  - r、s是字母表Σ上的一个RE，如果L(r)=L(s)，则称r与s相等，记作r=s 。
  - 相等也称为等价。
- **几个基本结论**

⑴ 结合律：(rs)t=r(st)

(r+s)+t=r+(s+t)

⑵ 分配律：r(s+t)=rs+rt

(s+t)r=sr+tr

4.2 RE的形式定义

⑶ 交换律：　r+s=s+r。

⑷ 幂等律：　r+r=r。

⑸ 加法运算零元素：r+ Φ=r。

⑹ 乘法运算单位元：r ε = ε r=r。

⑺ 乘法运算零元素：r Φ= Φr= Φ。

⑻ L(Φ)= Φ。

⑼ L(ε)={ ε }。

⑽ L(a)={a}。

4.2 RE的形式定义

⑾ L(rs)=L(r)L(s)。

⑿ L(r+s)=L(r)∪L(s)。

⒀ $L(r^*)=(L(r))^*$ 。

⒁ $L(\Phi^*)=\{\varepsilon\}$。

⒂ $L((r+\varepsilon)^*)=L(r^*)$。

⒃ $L((r^*)^*)=L(r^*)$。

⒄ $L((r^*s^*)^*)=L((r+s)^*)$。

⒅ 如果L(r) ⊆L(s)，则r+s=s。

4.2 RE的形式定义

⑲ $L(r^n)=(L(r))^n$ 。

⑳ $r^n r^m=r^{n+m}$ 。

一般地， $r+\varepsilon \neq r$，$(rs)^n \neq r^n s^n$，$rs \neq sr$。

- **幂**

  r是字母表Σ上的RE，r的n次幂定义为

  (1) $r^0=\varepsilon$ 。

  (2) $r^n=r^{n-1}r$。

4.2 RE的形式定义

- 例 4-2 设Σ={0，1}

  00表示语言{00}；

  $(0+1)^*00(0+1)^*$表示所有的至少含两个连续0的0、1串组成的语言；

  $(0+1)^*1(0+1)^9$表示所有的倒数第10个字符为1的串组成的语言；

4.2 RE的形式定义

L((0+1)*011)={x|x 是 以 011 结 尾 的 0 、 1 串};

L(0$^+$1$^+$2$^+$)={0^{n}1^{m}2^k|m，n，k≥1};

L(0*1*2*)={0^{n}1^{m}2^k|m，n，k≥0};

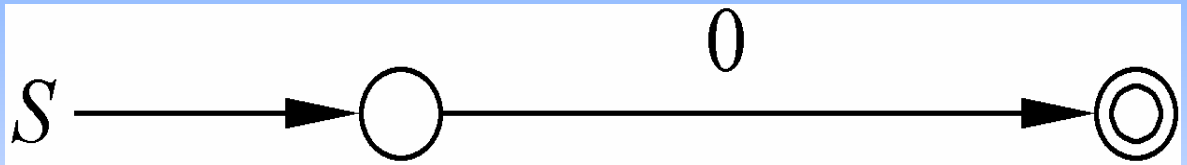
L(1(0+1)*1+0(0+1)*0))={x|x的开头字符与尾字符相同**}**。

4.3 RE与FA等价

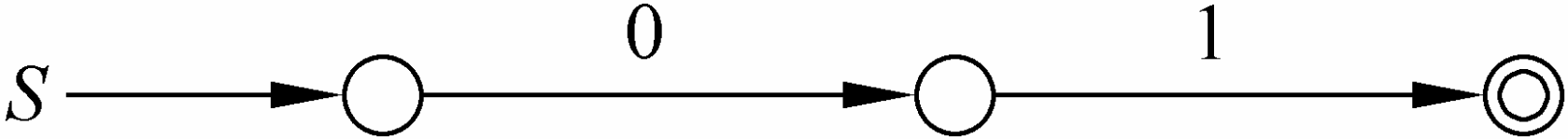
- 正则表达式**r**称为与**FA M**等价，如果**L(r)=L(M)**。
- 从开始状态出发，根据状态之间按照转移所确定的后继关系，依次计算出所给**FA**的各个状态**q**对应的**set(q)**，并且最终得到相应的**FA**接受的语言的**RE**表示。
- 寻找一种比较“机械”的方法，使得计算机系统能够自动完成**FA**与**RE**之间的转换。

4.3.1 RE到FA的等价变换

- 0对应的FA

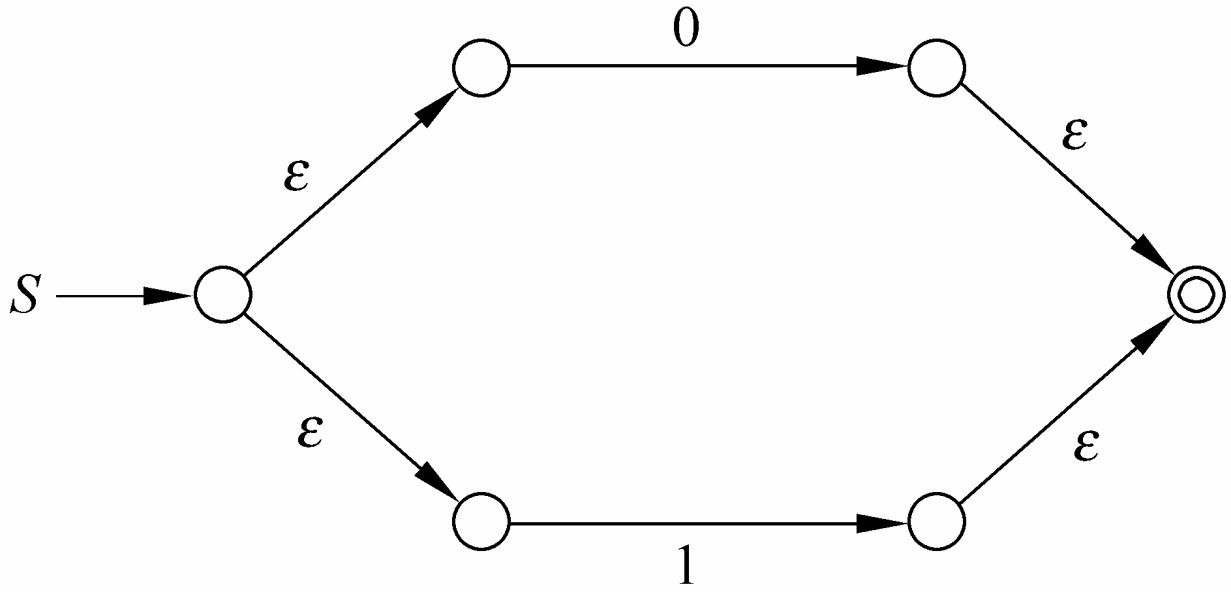


- 01对应的FA



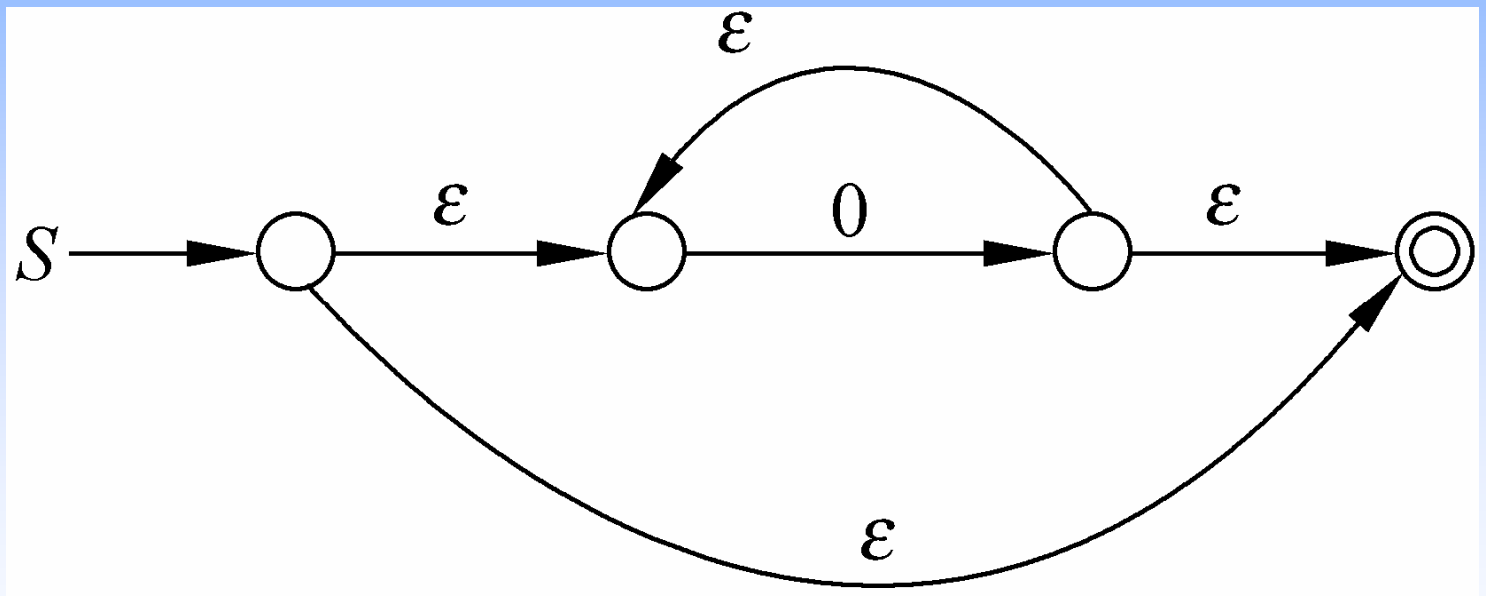
4.3.1 RE到FA的等价变换

- 0+1对应的FA



4.3.1 RE到FA的等价变换

- **0*对应的FA**



4.3.1 RE到FA的等价变换

定理 4-1 RE表示的语言是RL。

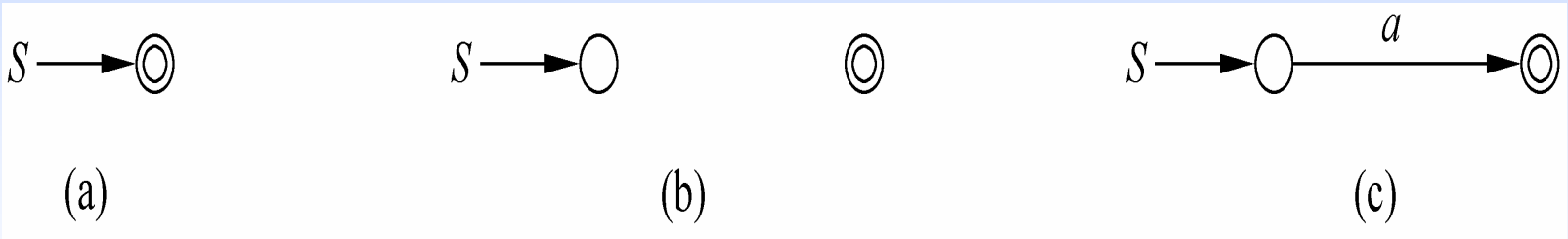
证明：

- 施归纳于正则表达式中所含的运算符的个数**n**，证明对于字母表Σ上的任意正则表达式**r**，存在**FA M**，使得**L(M) = L(r)** 。
  - **M**恰有一个终止状态。
  - **M**在终止状态下不作任何移动。

4.3.1 RE到FA的等价变换

n=0

r= ε　　**r= Φ**　　**r=a**



4.3.1 RE到FA的等价变换

n=k

设结论对于**n=k**时成立，此时有如下**FA**：

$M_1=(Q_1, \Sigma, \delta_1, q_{01}, \{f_1\})$

$M_2=(Q_2, \Sigma, \delta_2, q_{02}, \{f_2\})$

$L(M_1)=L(r_1)$，$L(M_2)=L(r_2)$。

$Q_1 \cap Q_2 = \Phi$。

4.3.1 RE到FA的等价变换

n=k+1

取q_0，$f \notin Q_1 \cup Q_2$，令

$M=(Q_1 \cup Q_2 \cup \{q_0, f\}, \Sigma, \delta, q_0, \{f\})$

① $\delta(q_0, \varepsilon)=\{q_{01}, q_{02}\}$;

② 对$\forall q \in Q_1$，$a \in \Sigma \cup \{\varepsilon\}$，$\delta(q, a)=\delta_1(q, a)$;

对$\forall q \in Q_2$，$a \in \Sigma \cup \{\varepsilon\}$，$\delta(q, a)=\delta_2(q, a)$;

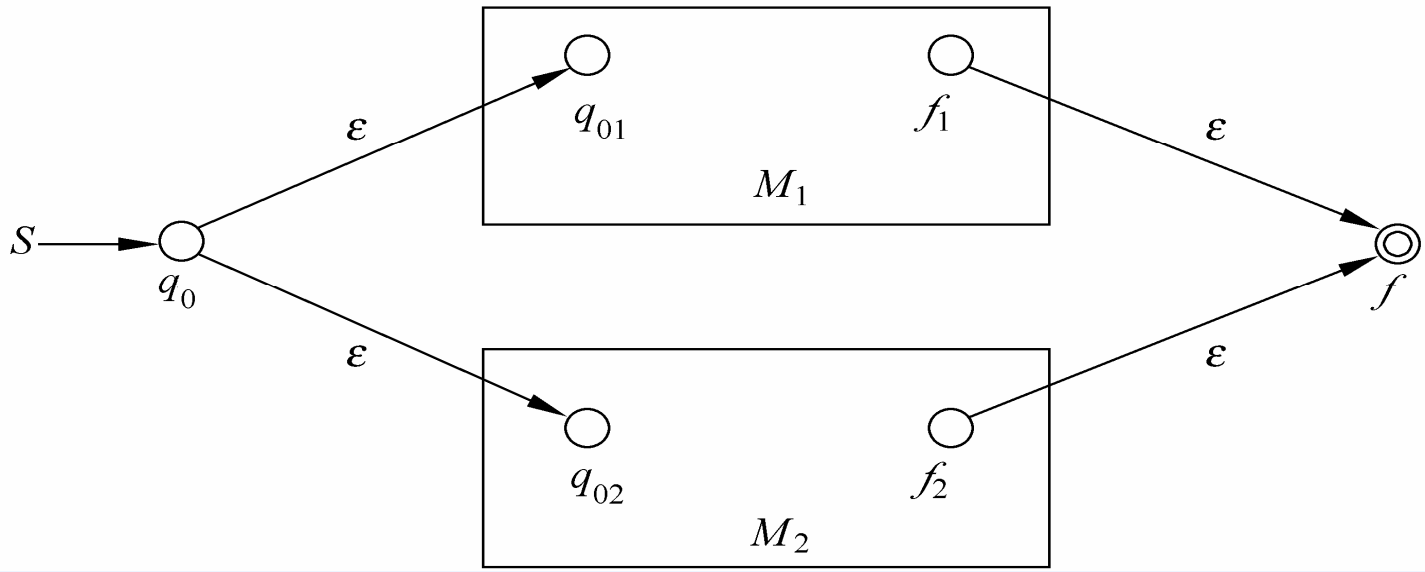
③ $\delta(f_1, \varepsilon)=\{f\}$;

④ $\delta(f_2, \varepsilon)=\{f\}$。

$r=r_1+r_2$

4.3.1 RE到FA的等价变换

n=k+1

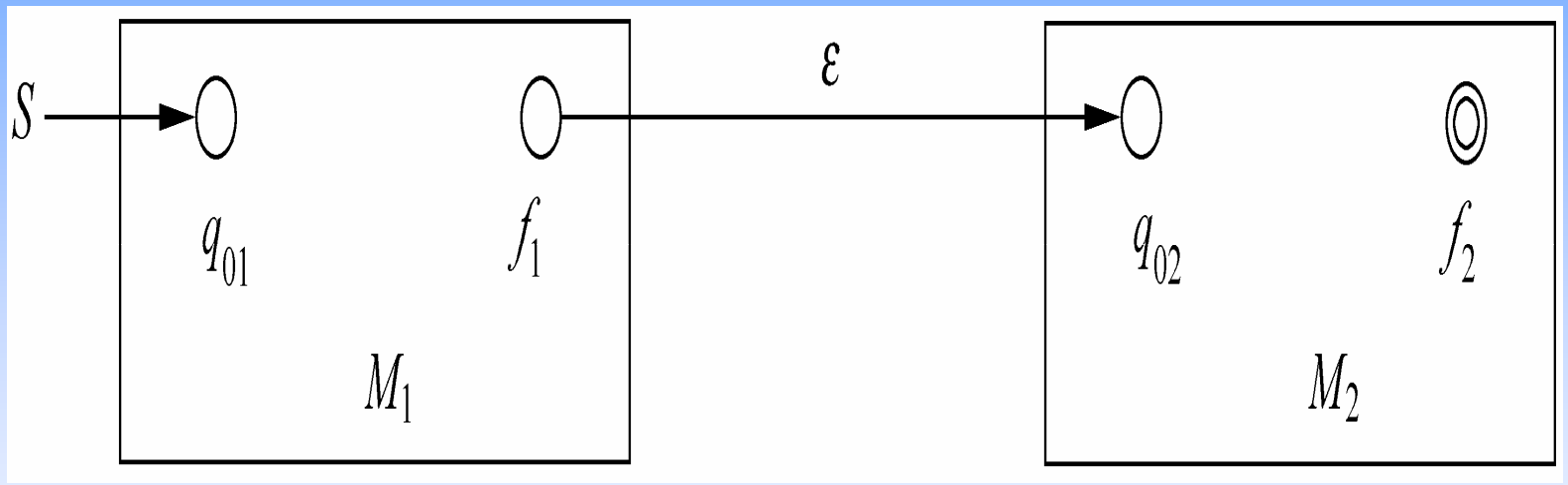


$r=r_1+r_2$

4.3.1 RE到FA的等价变换

- $M=(Q_1 \cup Q_2,\ \Sigma,\ \delta,\ q_{01},\ \{f_2\})$
- ① 对$\forall q \in Q_1-\{f_1\}$，$a \in \Sigma \cup \{\varepsilon\}$
  - $\delta(q,\ a)=\delta_1(q,\ a)$;
- ② 对$\forall q \in Q_2-\{f_2\}$，$a \in \Sigma \cup \{\varepsilon\}$
  - $\delta(q,\ a)=\delta_2(q,\ a)$;
- ③ $\delta(f_1,\ \varepsilon)=\{q_{02}\}$

4.3.1 RE到FA的等价变换



$$r=r_1r_2$$

4.3.1 RE到FA的等价变换

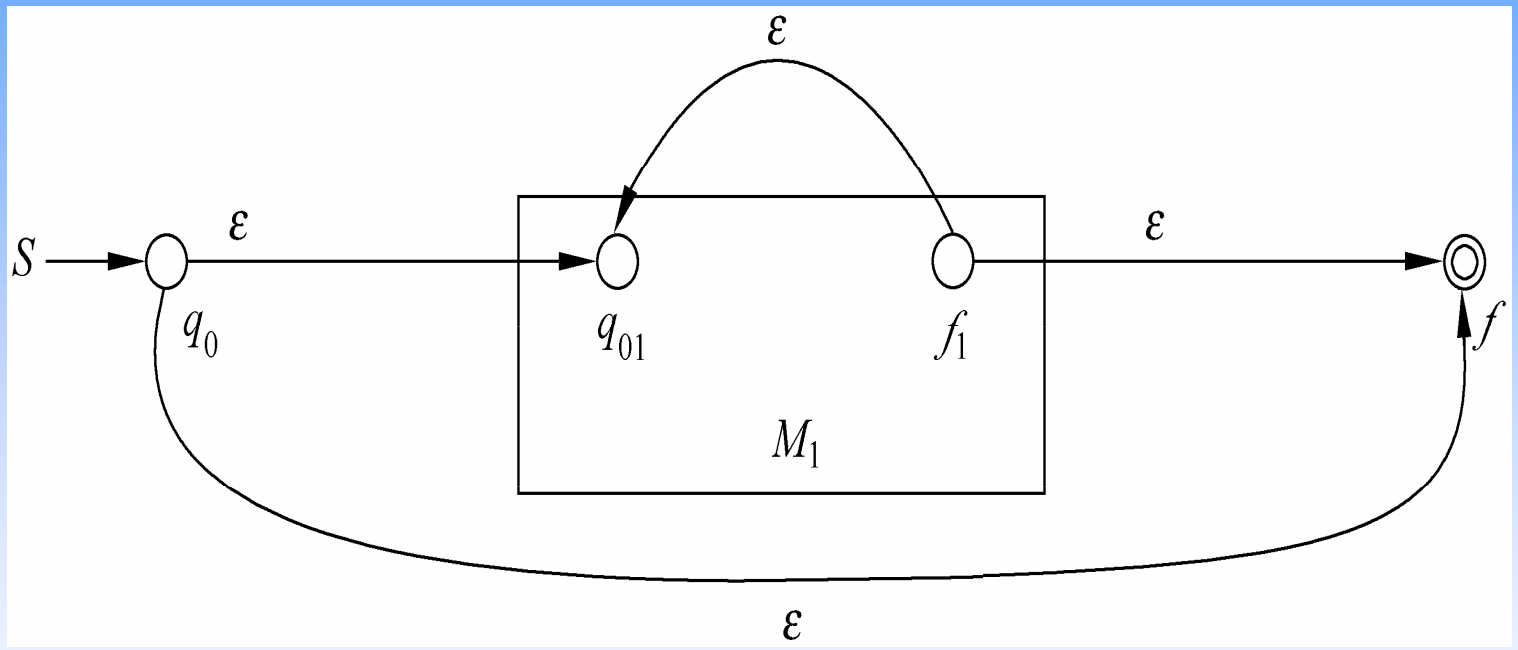
- $M=(Q_1 \cup \{q_0, f\}, \Sigma, \delta, q_0, \{f\})$
- 其中q_0，$f \notin Q_1$，定义δ为

  ① 对$\forall q \in Q_1-\{f_1\}$，$a \in \Sigma$，
  $\delta(q, a)=\delta_1(q, a)$。

  ② $\delta(f_1, \varepsilon)=\{q_{01}, f\}$。

  ③ $\delta(q_0, \varepsilon)=\{q_{01}, f\}$。

4.3.1 RE到FA的等价变换



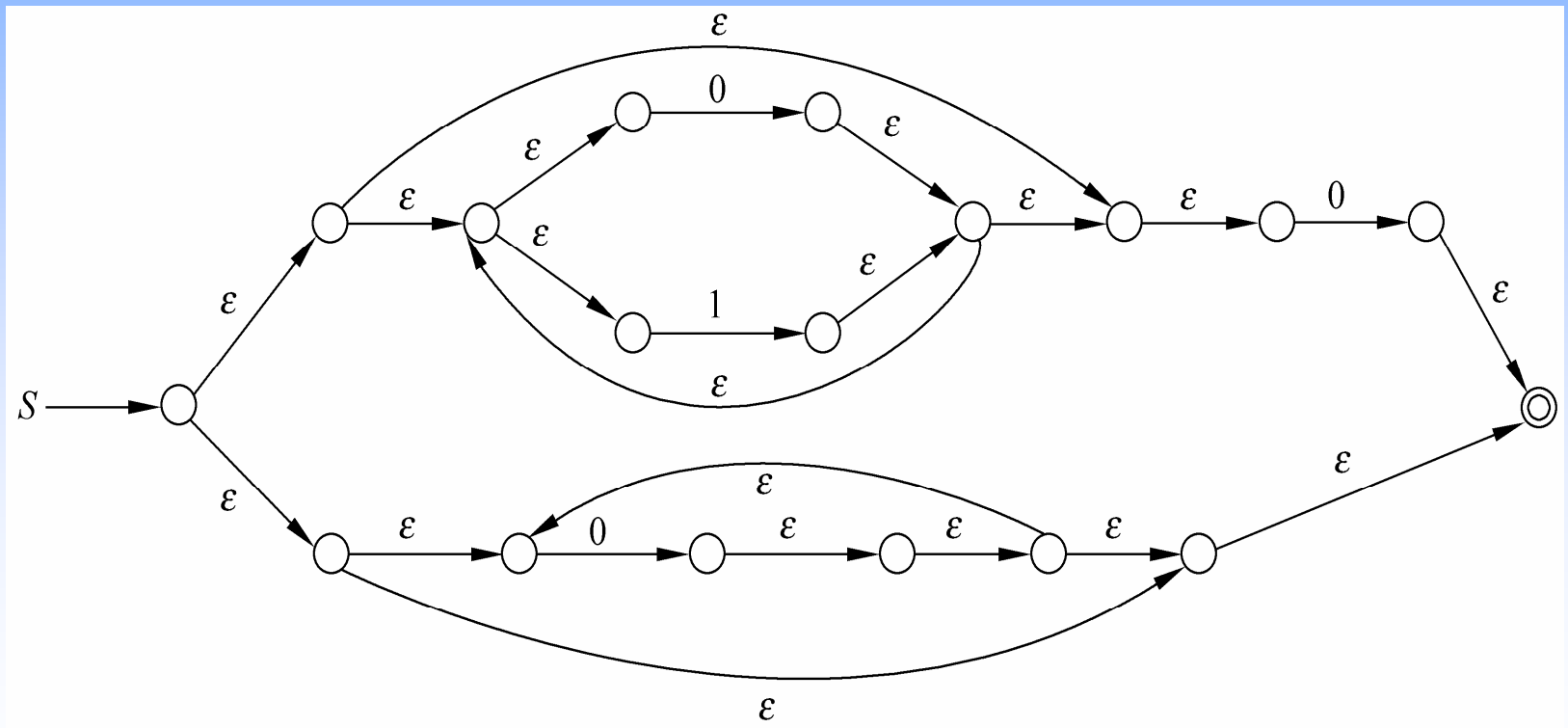
$$r=r_1^*$$

4.3.1 RE到FA的等价变换

- 按照定理4-1证明给出的方法构造一个给定RE的等价FA时，该FA有可能含有许多的空移动。
- 可以按照自己对给定RE的“理解”以及对FA的“理解”“直接地”构造出一个比较“简单”的FA。
- 定理证明中所给的方法是机械的。由于“直接地”构造出的FA的正确性依赖于构造者的“理解”，所以它的正确性缺乏有力的保证。

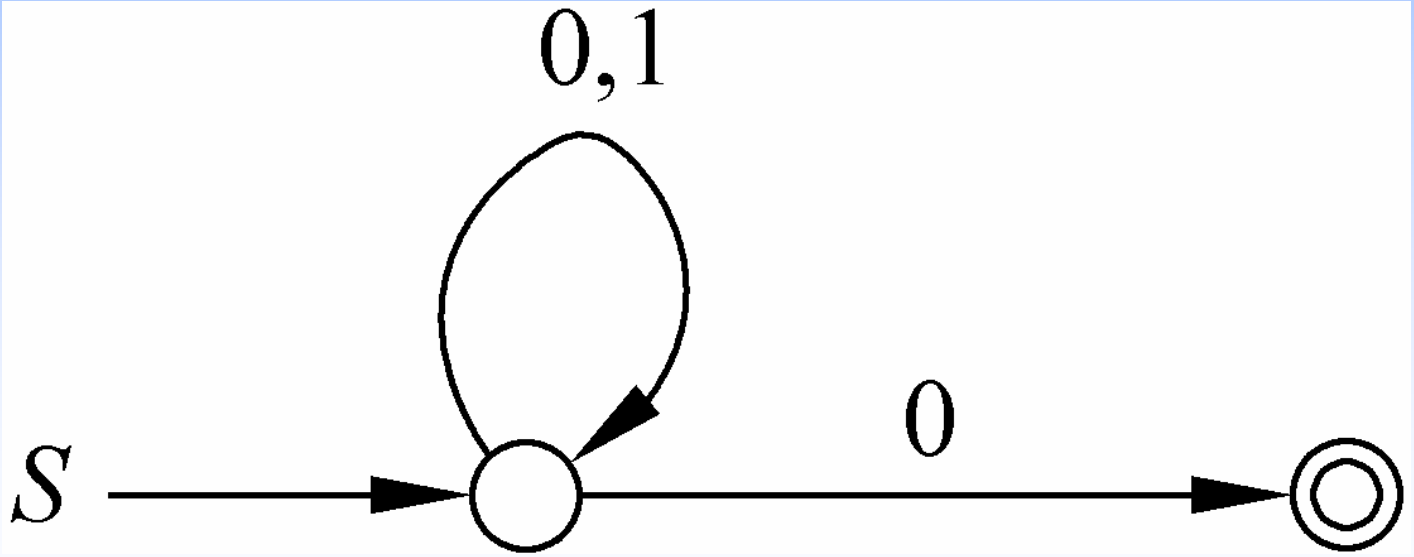
4.3.1 RE到FA的等价变换

- 例 4-3 构造与 $(0+1)^*0+(00)^*$等价的FA。



4.3.1 RE到FA的等价变换

- 按照对$(0+1)^*0+(00)^*$的“理解”“直接地”构造出的FA。



4.3.2 RL可以用RE表示

- 计算**DFA**的每个状态对应的集合——字母表的克林闭包的等价分类，是具有启发意义的。
- 这个计算过程难以“机械”地进行。
- 计算q_1到q_2的一类串的集合：R^k_{ij}。
- 图上作业法。

4.3.2 RL可以用RE表示

定理4-2 RL可以用RE表示。

设DFA

$$M=(\{q_1, q_2, \ldots, q_n\}, \Sigma, \delta, q_1, F)$$

$R^k_{ij}=\{x|\ \delta(q_i, x)=q_j$ 而且对于x的任意前缀 $y(y\neq x, y\neq \varepsilon)$，如果 $\delta(q_i, y)=q_l$，则 $l\leqslant k\}$。

4.3.2 RL可以用RE表示

$$R^0_{ij}=\begin{cases}\{a|\ \delta(q_i,\ a)=q_j\} & \text{如果}i\neq j\\ \{a|\ \delta(q_i,\ a)=q_j\}\cup\{\varepsilon\} & \text{如果}i=j\end{cases}$$

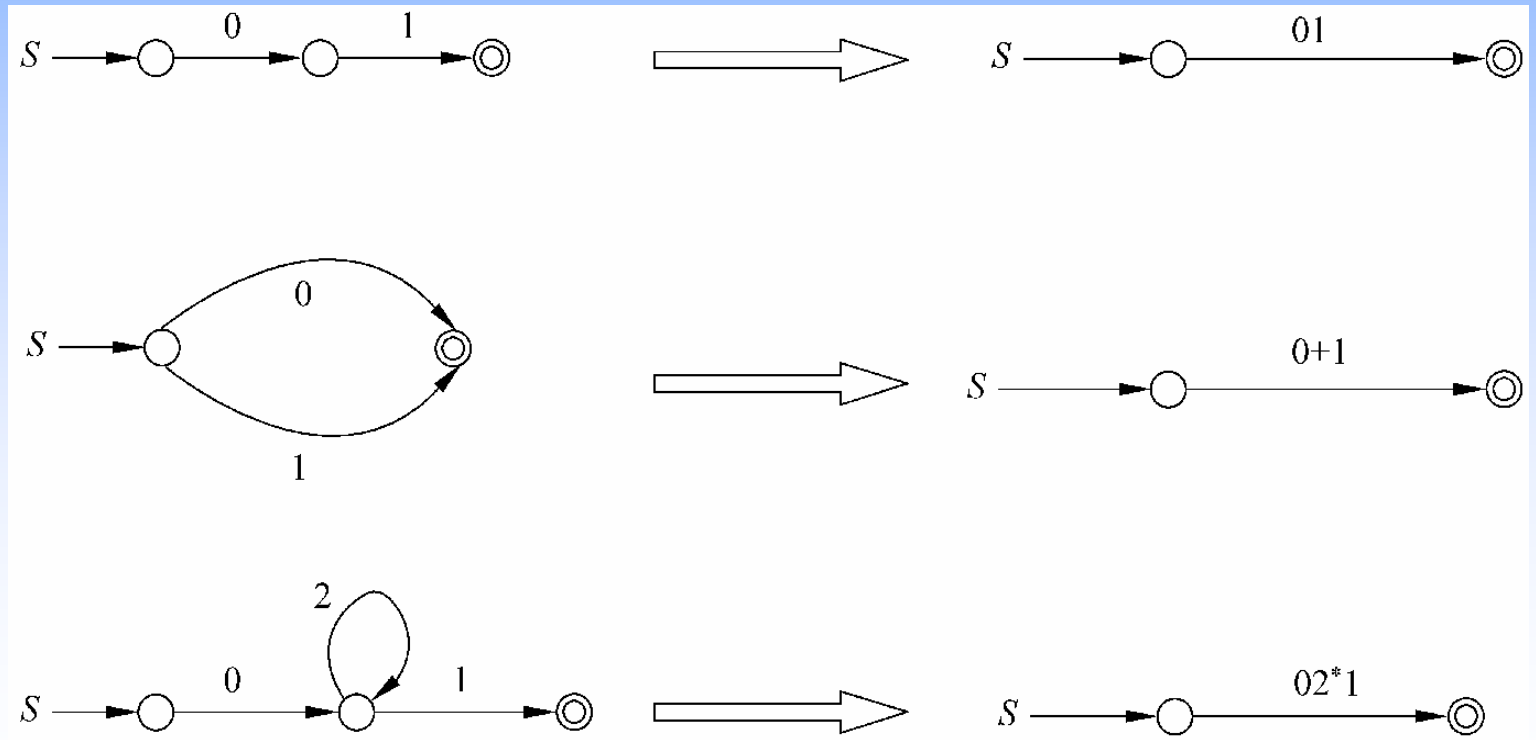
$$R^k_{ij}=R^{k-1}_{ik}(R^{k-1}_{kk})^*R^{k-1}_{kj}R^{k-1}_{ij}$$

$$L(M)=\bigcup_{q_f\in F}R^n_{1f}$$

4.3.2 RL可以用RE表示

- 图上作业法

  启示



4.3.2 RL可以用RE表示

- 图上作业法操作步骤

⑴ 预处理：

① 用标记为X和Y的状态将M“括起来”：

在状态转移图中增加标记为X和Y的状态，从标记为X的状态到标记为q_0的状态引一条标记为 ε 的弧；从标记为q(q∈F)的状态到标记为Y的状态分别引一条标记为 ε 的弧。

② 去掉所有的不可达状态。

4.3.2 RL可以用RE表示

⑵ 对通过步骤(1)处理所得到的状态转移图重复如下操作，直到该图中不再包含除了标记为X和Y外的其他状态，并且这两个状态之间最多只有一条弧。

- 并弧
  - 将从q到p的标记为r_1，r_2，…，r_g并行弧用从q到p的、标记为$r_1+r_2+\ldots+r_g$的弧取代这g个并行弧。

4.3.2 RL可以用RE表示

- 去状态1
  - 如果从q到p有一条标记为r_1的弧，从p到t有一条标记为r_2的弧，不存在从状态p到状态p的弧，将状态p和与之关联的这两条弧去掉，用一条从q到t的标记为r_1r_2的弧代替。
- 去状态2
  - 如果从q到p有一条标记为r_1的弧，从p到t有一条标记为r_2的弧，从状态p到状态p标记为r_3的弧，将状态p和与之关联的这三条弧去掉，用一条从q到t的标记为$r_1r_3^*r_2$的弧代替。

4.3.2 RL可以用RE表示

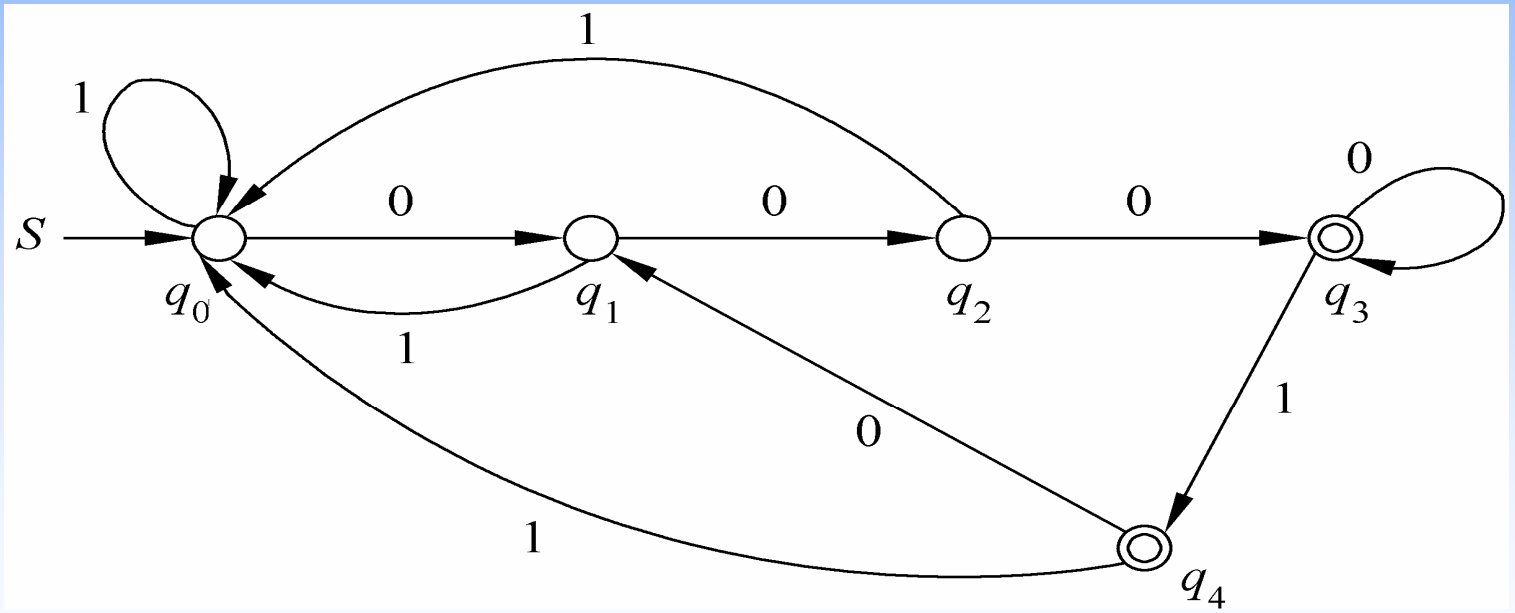
- 去状态3
  - 如果图中只有三个状态，而且不存在从标记为X的状态到达标记为Y的状态的路，则将除标记为X的状态和标记为Y的状态之外的第3个状态及其相关的弧全部删除。

4.3.2 RL可以用RE表示

- (3) 从标记为X的状态到标记为Y的状态的弧的标记为所求的正则表达式。如果此弧不存在，则所求的正则表达式为 Φ。

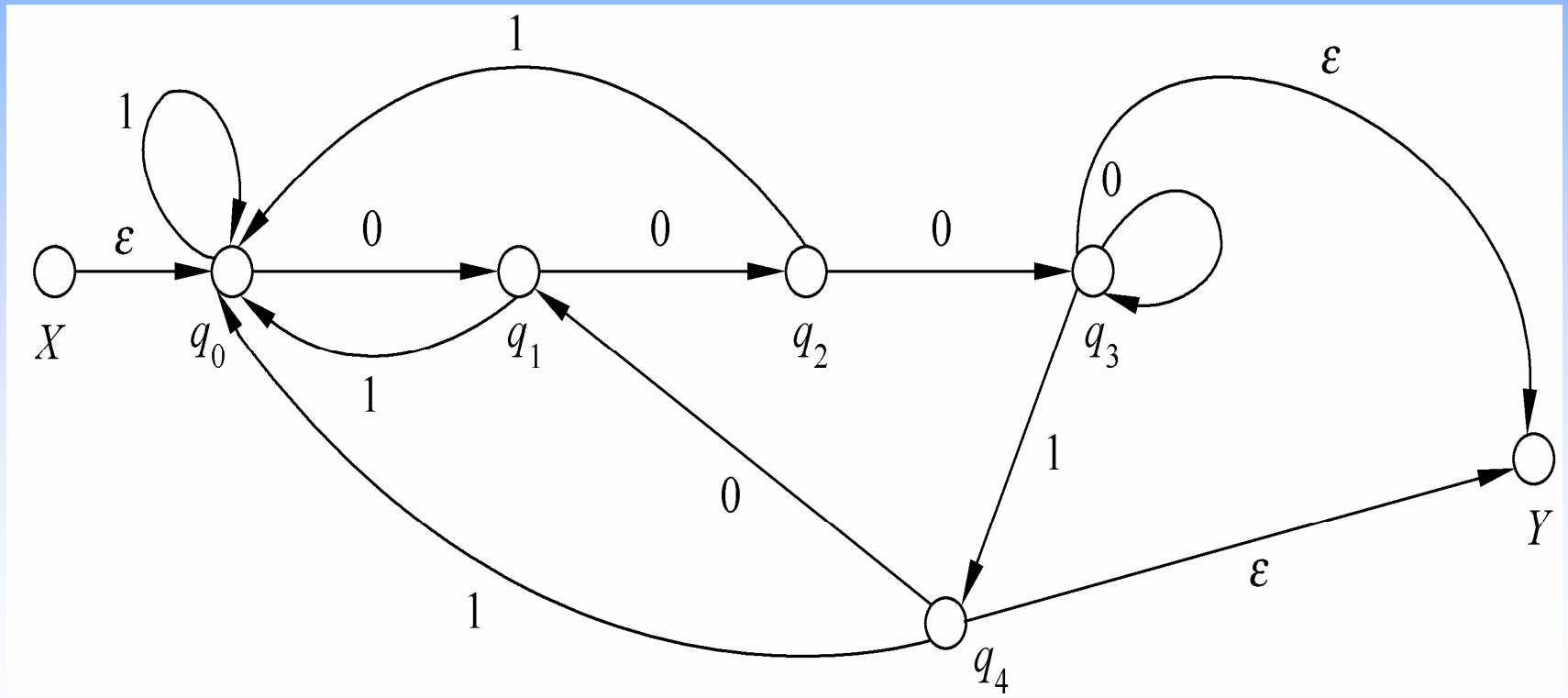
4.3.2 RL可以用RE表示

- 例 4-4 求图4-14所示的DFA等价的RE 。



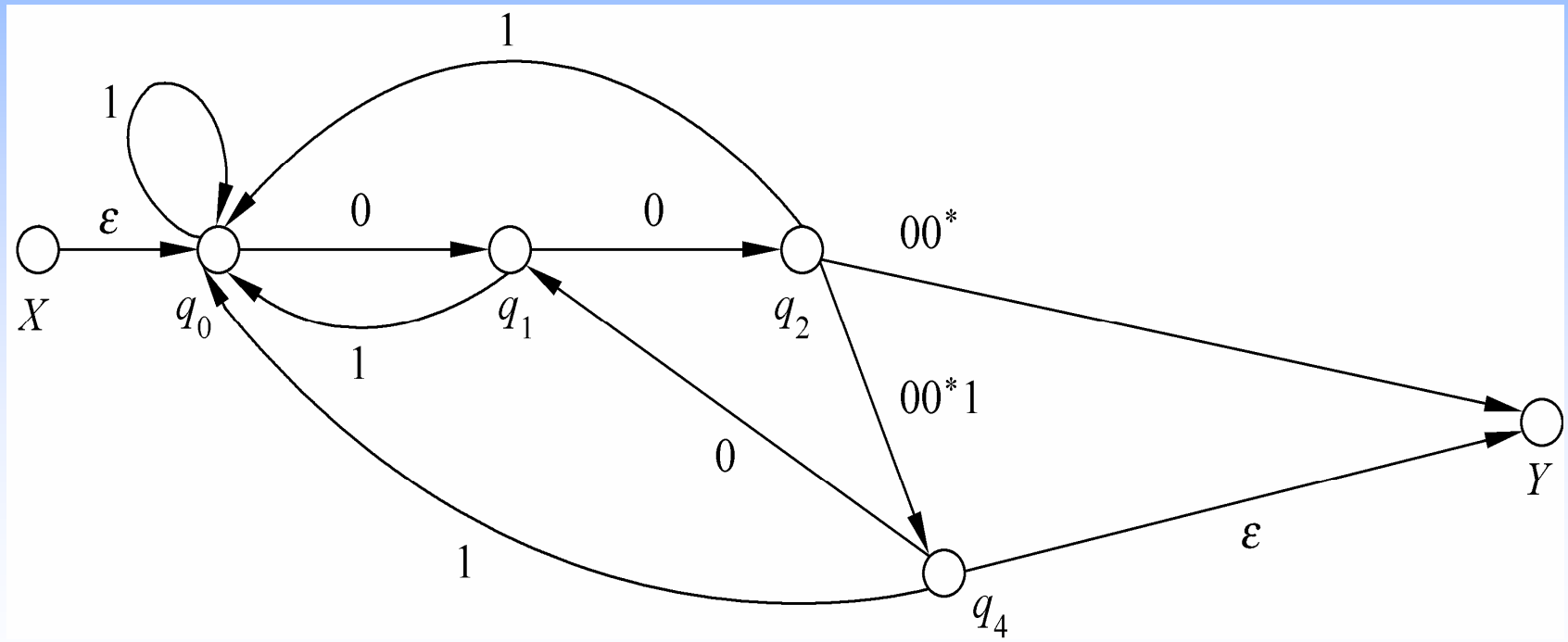
4.3.2 RL可以用RE表示

- 预处理。



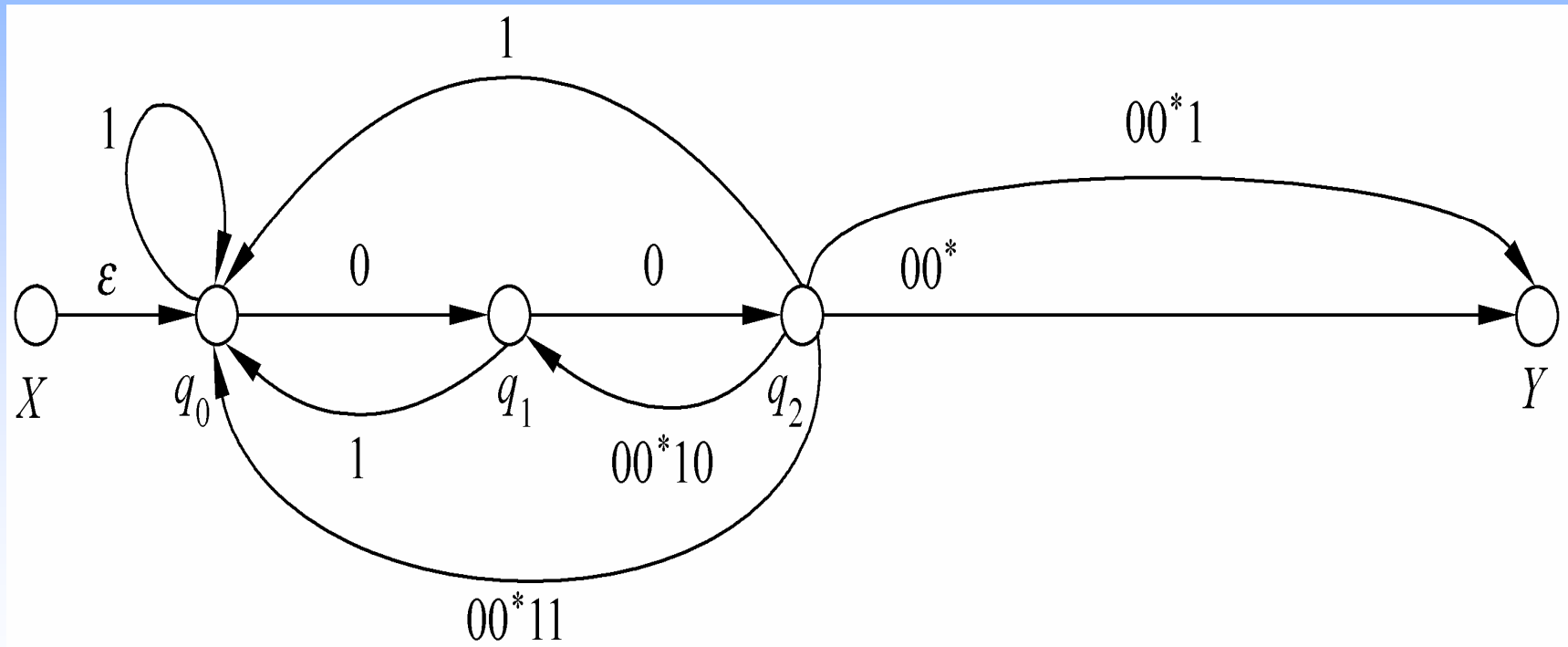
4.3.2 RL可以用RE表示

- 去掉状态q_3。



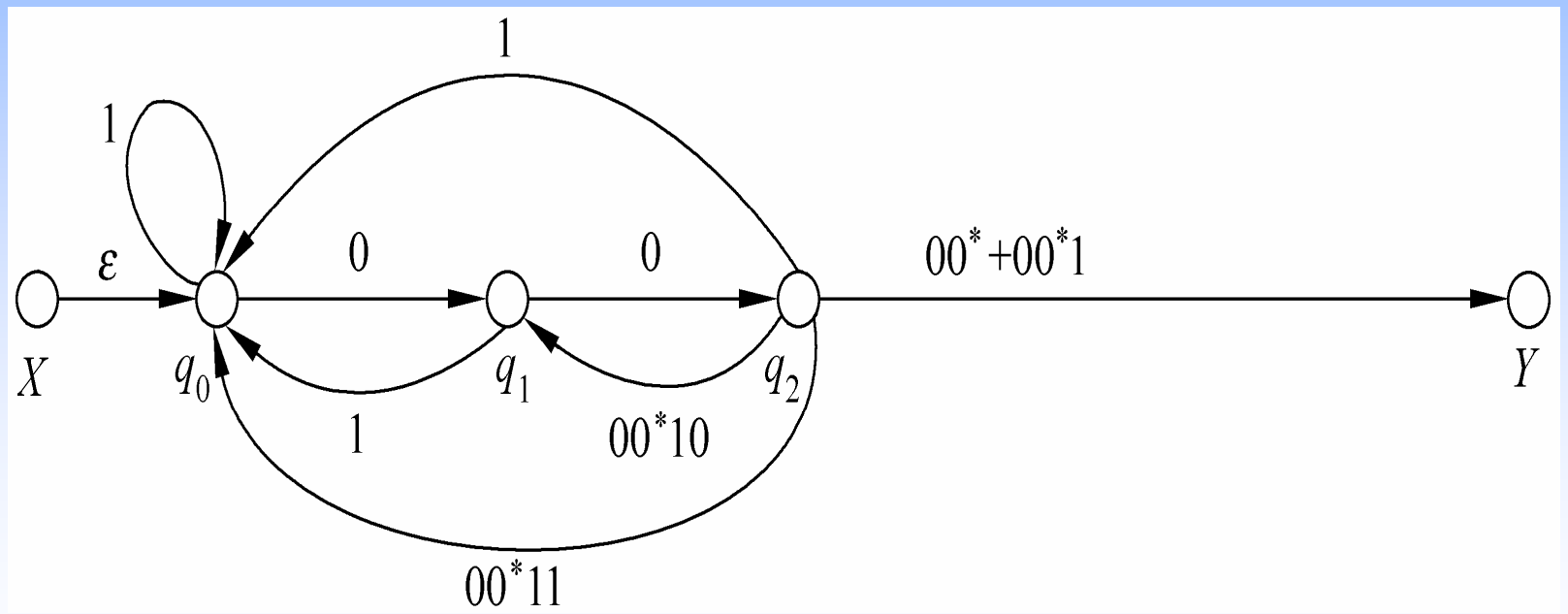
4.3.2 RL可以用RE表示

- 去掉状态q_4。



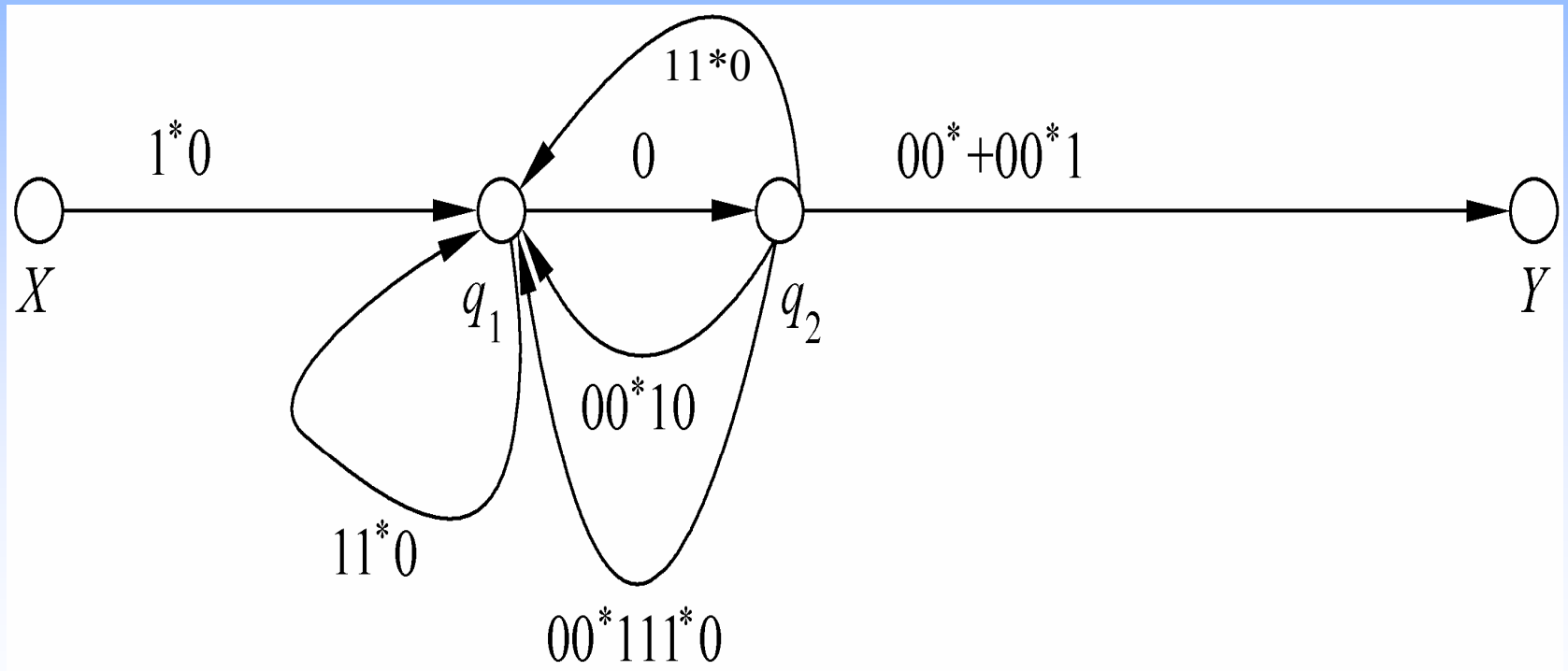
4.3.2 RL可以用RE表示

- 合并从标记为q_2的状态到标记为Y的状态的两条并行弧。



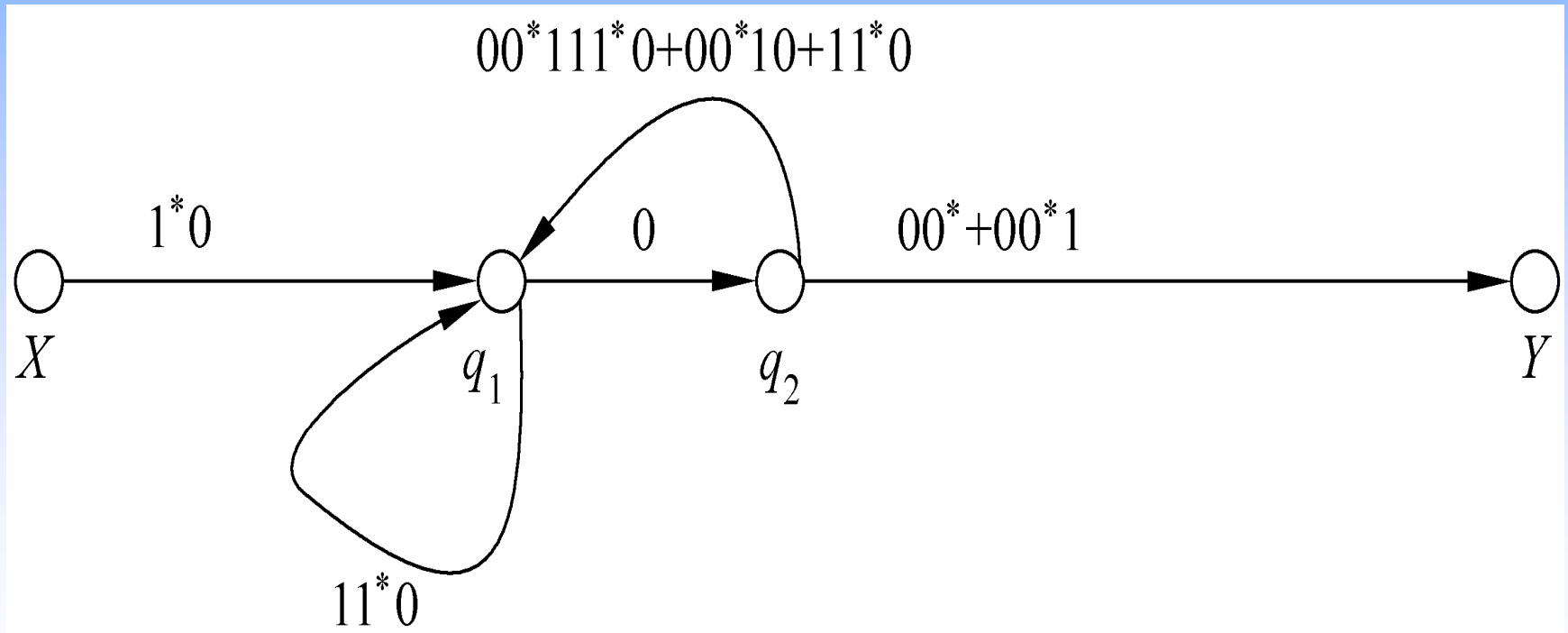
4.3.2 RL可以用RE表示

- 去掉状态q_0。



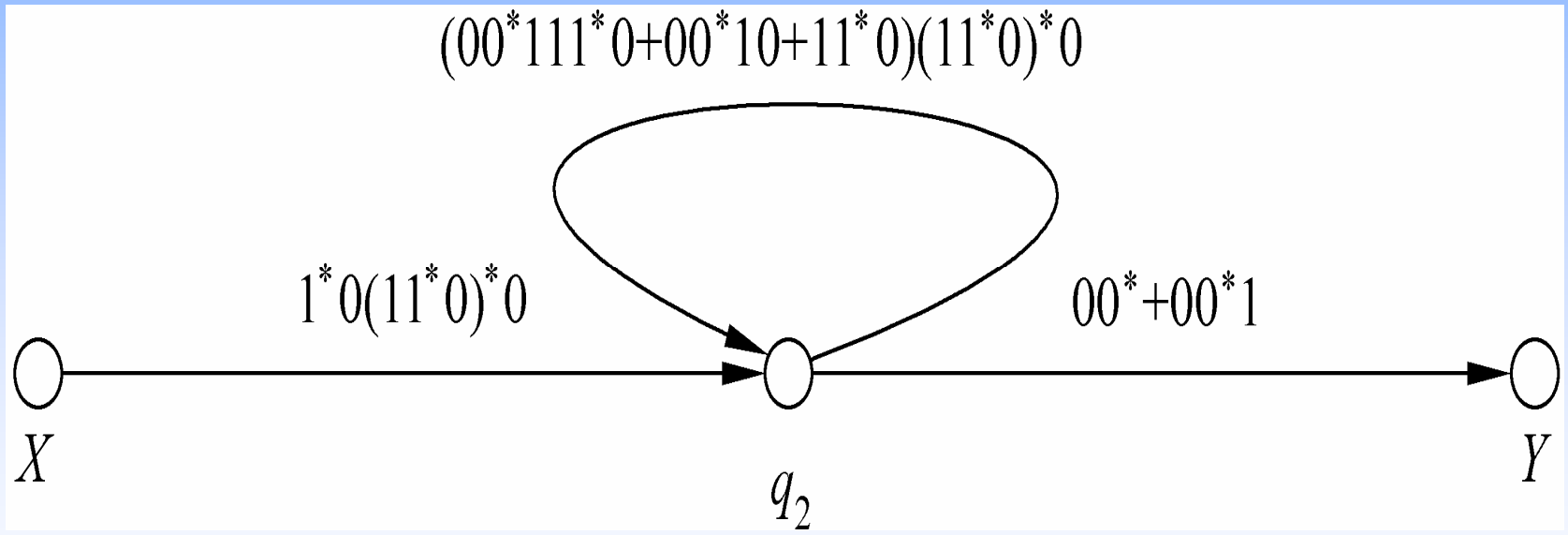
4.3.2 RL可以用RE表示

- 并弧。



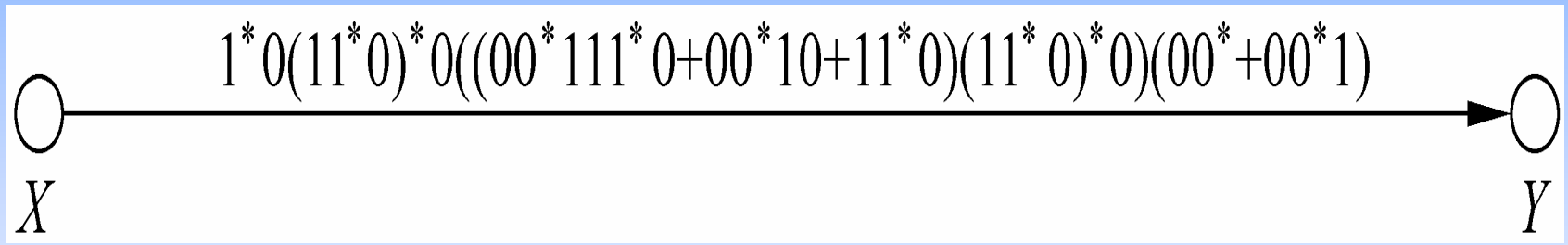
4.3.2 RL可以用RE表示

- 去掉状态q_1。



4.3.2 RL可以用RE表示

- 去掉状态q_2。



1*0(11*0)*0[(00*111*0+00*10+11*0)(11*0)*0]*(00*+00*1)就是所求。

4.3.2 RL可以用RE表示

- 几点值得注意的问题

  (1) 如果去状态的顺序不一样，则得到的RE可能在形式是不一样，但它们都是等价的。

  (2) 当DFA的终止状态都是不可达的时候，状态转移图中必不存在从开始状态到终止状态的路。此时，相应的RE为 Φ。

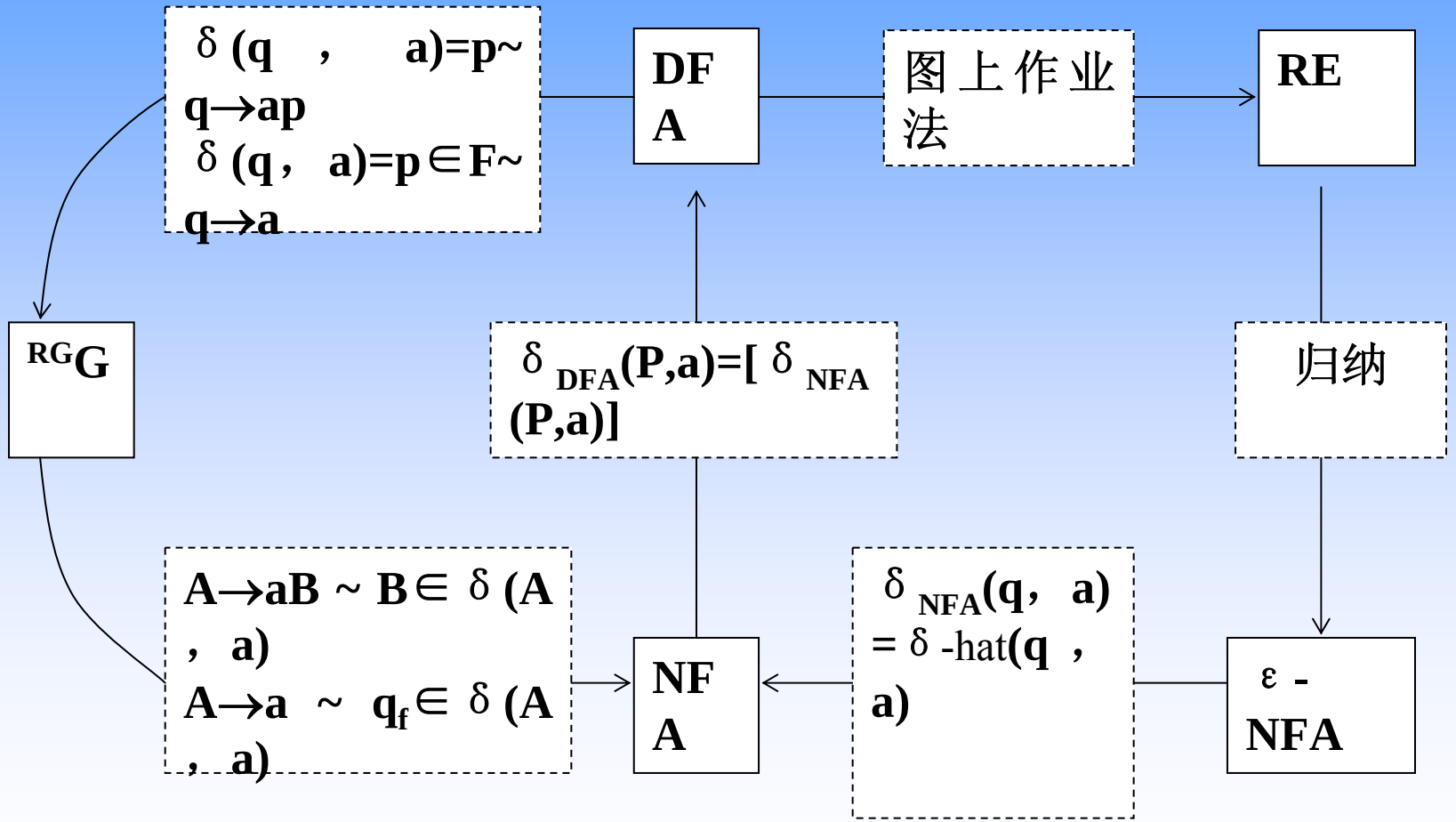
  (3) 不计算自身到自身的弧，如果状态q的入度为n，出度为m，则将状态q及其相关的弧去掉之后，需要添加n*m条新弧。

4.3.2 RL可以用RE表示

⑷ 对操作的步数施归纳，可以证明它的正确性。

推论4-1 正则表达式与**FA**、正则文法等价，是正则语言的表示模型。

4.4 正则语言等价模型的总结



4.5 小结

本章讨论了RL及其与FA的等价性。

(1) 字母表Σ上的RE用来表示Σ上的RL。Φ、ε、a（a∈Σ），是Σ上的最基本的RE，它们分别表示语言Φ、{ε}、{a}，以此为基础，如果r和s分别是Σ上的表示语言R和S的RE，则r+s、rs 、r*分别是Σ上的表示语言R∪S、RS、 R*的RE。如果L(r)=L(s)，则称r与s等价。

4.5 小结

(2) RE对乘、加满足结合律；乘对加满足左、右分配律；加满足交换率和幂等率；Φ是加运算的零元素； ε 是乘运算的单位元； Φ是乘运算的零元素。

(3) RE是RL的一种描述。容易根据RE构造出与它等价的**FA**。反过来，可以用图上作业法构造出与给定的**DFA**等价的RE。

(4) RL的5种等价描述模型转换图。

第5章 RL的性质

- **RL性质**
  - 泵引理及其应用
  - 并、乘积、闭包、补、交
  - 正则代换、同态、逆同态的封闭性
- 从RL固有特征寻求表示的一致性
  - Myhill-Nerode定理
  - FA的极小化
- RL的几个判定问题
  - 空否、有穷否、两个DFA等价否、成员关系

5.1 RL的泵引理

- **泵引理(pumping lemma)**

设L为一个 RL ，则存在仅依赖于L的正整数N，对于$\forall z \in L$，如果$|z| \geqslant N$，则存在u、v、w，满足

⑴ $z=uvw$;

⑵ $|uv| \leqslant N$;

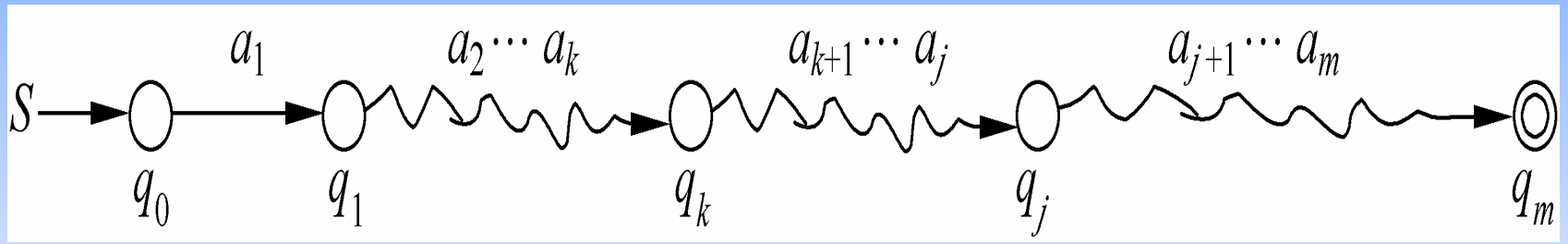
⑶ $|v| \geqslant 1$;

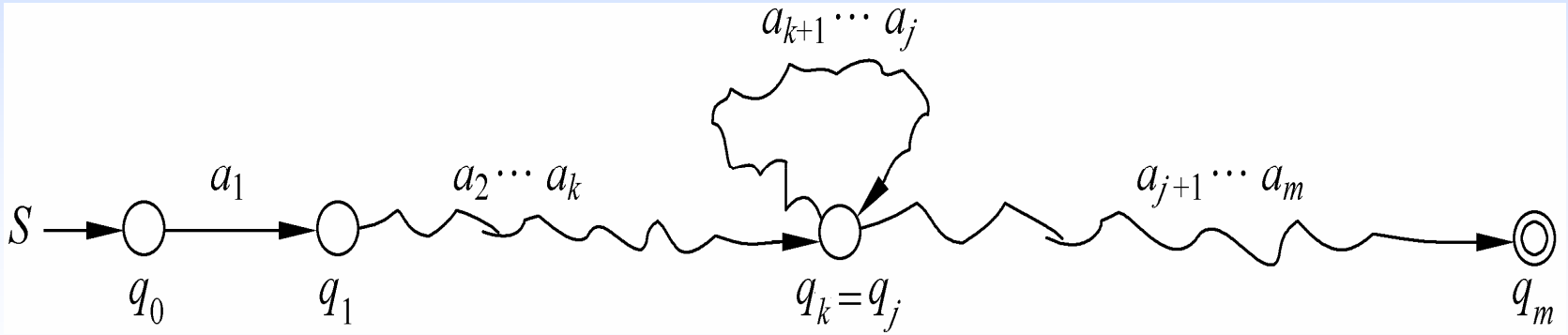
⑷ 对于任意的整数$i \geqslant 0$，$uv^iw \in L$;

⑸ N不大于接受L的最小DFA M的状态数。

5.1 RL的泵引理

- 证明思想





5.1 RL的泵引理

证明：

DFA在处理一个足够长的句子的过程中，必定会重复地经过某一个状态。换句话说，在**DFA**的状态转移图中，必定存在一条含有回路的从启动状态到某个终止状态的路。由于是回路，所以，**DFA**可以根据实际需要沿着这个回路循环运行，相当于这个回路中弧上的标记构成的非空子串可以重复任意多次。

5.1 RL的泵引理

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

$|Q|=N$

$z= a_1a_2\ldots a_m \quad m \geqslant N$

$\delta(q_0, a_1a_2\ldots a_h)=q_h$

状态序列$q_0, q_1, \ldots, q_N$中，至少有两个状态是相同：$q_k=q_j$

5.1 RL的泵引理

$\delta(q_0, a_1a_2\ldots a_k)=q_k$

$\delta(q_k, a_{k+1}\ldots a_j)=q_j=q_k$

$\delta(q_j, a_{j+1}\ldots a_m)=q_m$

对于任意的整数$i \geq 0$

$\delta(q_k, (a_{k+1}\ldots a_j)^i)$

$= \delta(q_k, (a_{k+1}\ldots a_j)^{i-1})$

…

$= \delta(q_k, a_{k+1}\ldots a_j)=q_k$

5.1 RL的泵引理

故，

$\delta(q_0, a_1a_2\ldots a_k(a_{k+1}\ldots a_j)^i a_{j+1}\ldots a_m)=q_m$

也就是说，

$a_1a_2\ldots a_k(a_{k+1}\ldots a_j)^i a_{j+1}\ldots a_m \in L(M)$

$u= a_1a_2\ldots a_k$，

$v=a_{k+1}\ldots a_j$，

$w= a_{j+1}\ldots a_m$

$uv^iw \in L$

5.1 RL的泵引理

- **例 5-1** 证明$\{0^n1^n|n\geqslant 1\}$不是 RL。

  证明：

  假设$L=\{0^n1^n|n\geqslant 1\}$ 是 RL

  $z=0^N1^N$

  按照泵引理所述

  $v=0^k \quad k\geqslant 1$

  此时有，

  $u=0^{N-k-j}$

  $w=0^j1^N$

5.1 RL的泵引理

从而有，

$uv^iw=0^{N-k-j}(0^k)^i0^j1^N=0^{N+(i-1)k}1^N$

当i=2时，我们有：

$uv^2w=0^{N+(2-1)k}1^N = 0^{N+k}1^N$

注意到$k \geqslant 1$，所以，

$N+k>N$。

这就是说，

$0^{N+k}1^N \notin L$

这与泵引理矛盾。所以，L不是 RL。

5.1 RL的泵引理

•**例 5-2** 证明$\{0^n|n$为素数$\}$不是 **RL**。

证明：假设$L=\{0^n|n$为素数$\}$ 是 **RL**。

取 $z=0^{N+p} \in L$，

不妨设$v=0^k$， $k \geqslant 1$

从而有，

$$uv^iw=0^{N+p-k-j}(0^k)^i0^j$$

$$=0^{N+p+(i-1)k}$$

5.1 RL的泵引理

当i=N+p+1时，

$N+p+(i-1)k=N+p+(N+p+1-1)k$

$= N+p+(N+p)k$

$= (N+p)(1+k)$

注意到$k \geq 1$，所以

$N+p+(N+p+1-1)k=(N+p)(1+k)$

不是素数。故当i=N+p+1时，

$uv^{N+p+1}w=0^{(N+p)(1+k)} \notin L$

这与泵引理矛盾。所以，L不是 RL。

5.1 RL的泵引理

- 例 5-3 证明$\{0^n1^m2^{n+m}|m，n\geqslant 1\}$不是 RL。

证明：假设$L=\{0^n1^m2^{n+m}|m，n\geqslant 1\}$ 是RL。

取$z=0^N1^N2^{2N}$

设$v=0^k$　　$k\geqslant 1$

从而有，

$$uv^iw=0^{N-k-j}(0^k)^i0^j1^N2^{2N}$$

$$=0^{N+(i-1)k}1^N2^{2N}$$

5.1 RL的泵引理

$uv^0w=0^{N+(0-1)k}1^N2^{2N}$

$= 0^{N-k}1^N2^{2N}$

注意到$k \geqslant 1$，

$N-k+N=2N-k<2N$

$0^{N-k}1^N2^{2N} \notin L$

这个结论与泵引理矛盾。所以，L不是 RL。

5.1 RL的泵引理

- 用来证明一个语言不是 RL
- 不能用泵引理去证明一个语言是 RL。

⑴ 由于泵引理给出的是 RL 的必要条件，所以，在用它证明一个语言不是 RL 时，我们使用反证法。

⑵ 泵引理说的是对 RL 都成立的条件，而我们是要用它证明给定语言不是 RL，这就是说，相应语言的“仅仅依赖于L的正整数N”实际上是不存在的。所以，我们一定是无法给出一个具体的数的。因此，人们往往就用符号N来表示这个“假定存在”、而实际并不存在的数。

5.1 RL的泵引理

⑶ 由于泵引理指出，如果**L**是 **RL** ，则对任意的$z \in L$，只要$|z| \geq N$，一定会存在**u**、**v**、**w**，使$uv^iw \in L$对所有的**i**成立。因此，我们在选择**z**时，就需要注意到论证时的简洁和方便。

⑷ 当一个特意被选来用作“发现矛盾”的**z**确定以后，就必须依照$|uv| \leq N$和$|v| \geq 1$的要求，说明**v不存在(**对“存在**u**、**v**、**w**”的否定**)** 。

5.1 RL的泵引理

(5) 与选z时类似，在寻找i时，我们也仅需要找到一个表明矛盾的“具体”值就可以了(对“所有i”的否定)。

(6) 一般地，在证明一个语言不是 RL 的时候，我们并不使用泵引理的第(5)条。

(7) 事实上，引理所要求的|uv|≤N并不是必须的。这个限制为我们简化相应的证明提供了良好支撑——扩充了的泵引理 。

5.2 RL的封闭性

- **封闭性(closure property)**

　　如果任意的、属于同一语言类的语言在某一特定运算下所得的结果仍然是该类语言，则称该语言类对此运算是**封闭**的

- **有效封闭性(valid closure property)**

　　给定一个语言类的若干个语言的描述，如果存在一个算法，它可以构造出这些语言在给定运算下所获得的运算结果的相应形式的语言描述，则称此语言类对相应的运算是**有效封闭**的。

5.2 RL的封闭性

定理 5-1 **RL** 在并、乘积、闭包运算下是封闭的。

- 根据**RE**的定义，立即可以得到此定理。

5.2 RL的封闭性

定理 5-2 **RL** 在补运算下是封闭的。

证明：

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

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

显然，对于任意的$x \in \Sigma^*$，

$\delta(q_0, x)=f \in F \Leftrightarrow \delta(q_0, x)=f \notin Q-F$

即： $x \in L(M) \Leftrightarrow x \notin L(M')$，

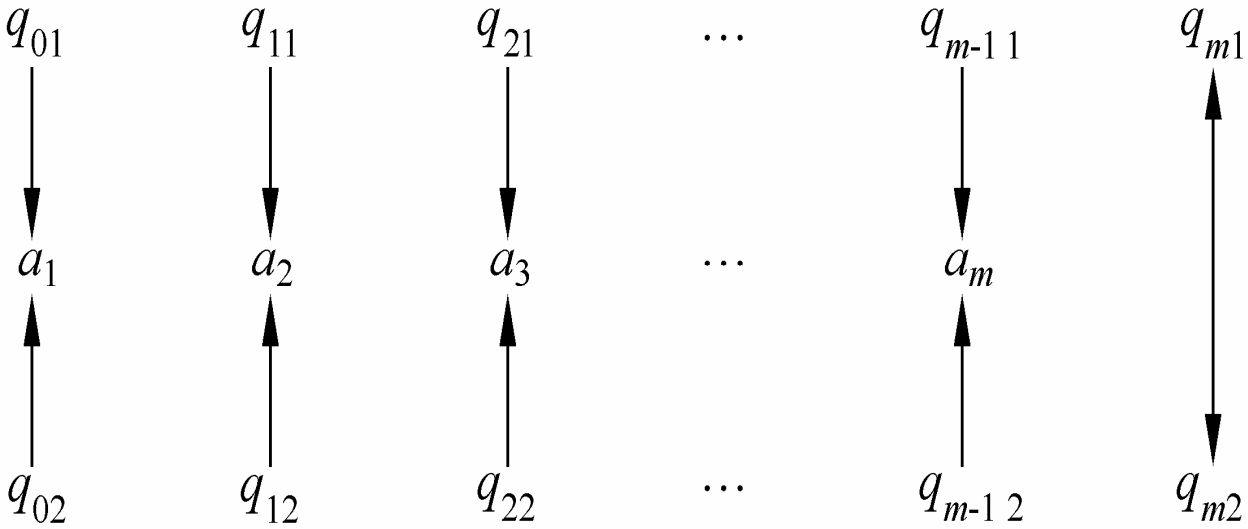
$L(M')= \Sigma^*-L(M)$。

所以， **RL** 在补运算下是封闭的。定理得到证明。

5.2 RL的封闭性

定理 5-3 RL 在交运算下封闭。

证明思路



5.2 RL的封闭性

- **正则代换(regular substitution)**

设Σ、Δ是两个字母表，映射

$$f:\Sigma \to 2^{\Delta^*}$$

被称为是从Σ到Δ的**代换**。如果对于∀**a**∈Σ，**f(a)**是Δ上的 RL ，则称f为**正则代换**。

5.2 RL的封闭性

- 先将f的定义域扩展到Σ^*上：

$$f:\Sigma^* \to 2^{\Delta^*}$$

(1) f(ε)={ε}；

(2) f(xa)=f(x)f(a)。

5.2 RL的封闭性

- 再将f的定义域扩展到 2^{Σ^*}

$$f : 2^{\Sigma^*} \rightarrow 2^{\Delta^*}$$

对于$\forall L \subseteq \Sigma^*$

$$f(L) = \bigcup_{x \in L} f(x)$$

5.2 RL的封闭性

- 例 5-4 设$\Sigma=\{0，1\}$，$\Delta=\{a，b\}$，$f(0)=a$，$f(1)=b^*$，则

$f(010)=f(0)f(1)f(0)=ab^*a$

$f(\{11，00\})=f(11)\cup f(00)$

$=f(1)f(1)\cup f(0)f(0)=b^*b^*+aa=b^*+aa$

$f(L(0^*(0+1)1^*))=L(a^*(a+b^*)(b^*)^*)$

$=L(a^*(a+b^*)b^*)=L(a^*ab^*+a^*b^*b^*)$

$=L(a^*b^*)$

5.2 RL的封闭性

- f是正则代换, 则
  ⑴ **f(Φ)=** Φ ;
  ⑵ **f(ε)=** ε ;
  ⑶ 对于$\forall$**a**$\in\Sigma$，**f(a)**是Δ上的RE;
  ⑷ 如果**r**，**s**是Σ上的RE，则
  f(r+s)=f(r)+f(s)
  f(rs)=f(r)f(s)
  f(r^*)=f(r)*
  是Δ上的RE。

5.2 RL的封闭性

定理 5-4 设L是Σ上的一个 **RL**

$$f:\Sigma \to 2^{\Delta^*}$$

是正则代换，则**f(L)**也是 **RL**。

证明：

描述工具：**RE**。

对**r**中运算符的个数**n**施以归纳，证明**f(r)**是表示**f(L)**的**RE**。

5.2 RL的封闭性

- 当**n=0**时，结论成立。
- 当**n≤k**时定理成立, 即当r中运算符的个数不大于k时：f(L(r)) = L(f®。
- 当**n=k+1**时，

5.2 RL的封闭性

(1) $r=r_1+r_2$。

$f(L)=f(L(r))$

$=f(L(r_1+r_2))$

$=f(L(r_1)\cup L(r_2))$ RE的定义

$=f(L(r_1))\cup f(L(r_2))$ 正则代换的定义

$=L(f(r_1))\cup L\ (f\ (r_2))$ 归纳假设

$=L(f(r_1)+f\ (r_2))$ RE的定义

$=L(f(r_1+r_2))$ RE的正则代换的定义

$=L(f(r))$

5.2 RL的封闭性

(2) $r=r_1r_2$。

$f(L)=f(L(r))$

$=f(L(r_1r_2))$

$=f(L(r_1)\ L(r_2))$ RE的定义

$=f(L(r_1))\ f(L(r_2))$ 正则代换的定义

$=L(f(r_1))\ L\ (f\ (r_2))$ 归纳假设

$=L(f(r_1)\ f\ (r_2))$ RE的定义

$=L(f(r_1r_2))$ RE的正则代换的定义

$=L(f(r))$

5.2 RL的封闭性

(3) $r=r_1^*$。

$f(L)=f(L(r))$

$=f(L(r_1^*))$

$=f(L(r_1)^*)$ RE的定义

$=(f(L(r_1)))^*$ 正则代换的定义

$=(L(f(r_1)))^*$ 归纳假设

$=L(f(r_1)^*)$ RE的定义

$=L(f(r_1^*))$ RE的正则代换的定义

$=L(f(r))$

5.2 RL的封闭性

- **例5-5** 设$\Sigma=\{0，1，2\}$，$\Delta=\{a，b\}$，正则代换f定义为：

  $f(0)=ab$；

  $f(1)=b^*a^*$；

  $f(2)=a^*(a+b)$

则：

⑴ $f(00)=abab$；

⑵ $f(010)=abb^*a^*ab=ab^+a^+b$；

5.2 RL的封闭性

⑶ $f((0+1+2)^*)=(ab+b^*a^*+a^*(a+b))^*$

$=(b^*a^*+a^*(a+b))^*=(a+b)^*$;

⑷ $f(0(0+1+2)^*)=ab(ab+b^*a^*+a^*(a+b))^*$

$=ab(a+b)^*$;

⑸ $f(012)=abb^*a^*\ a^*\ (a+b)=ab^+a^*(a+b)$;

⑹ $f((0+1)^*)=(ab+b^*a^*)^*$

$=(ab+b+a+b^*a^*)^*=(a+b)^*$ 。

5.2 RL的封闭性

- **同态映射(homomorphism)**

  设Σ、Δ是两个字母表，

$$f:\Sigma \rightarrow \Delta^{*}$$

  f为映射，如果对于∀x，y∈Σ*，

  f(xy)=f(x)f(y)，

  则称**f**为从Σ到Δ*的**同态映射**。

5.2 RL的封闭性

- 对于$\forall L \subseteq \Sigma^*$，L的同态像

$$f(L) = \bigcup_{x \in L} f(x)$$

对于$\forall w \subseteq \Delta^*$，w的同态原像是一个集合

$$f^{-1}(w) = \{x \mid f(x) = w \ \& \ x \in \Sigma^*\}$$

对于$\forall L \subseteq \Delta^*$，L的同态原像是一个集合

$$f^{-1}(L) = \{x \mid f(x) \in L\}$$

5.2 RL的封闭性

- **例5-6** 设Σ={0，1}，Δ={a，b}，同态映射f定义为

  f(0)=aa

  f(1)=aba

则：

⑴ **f(01)=aaaba**；

⑵ **f((01)*)=(aaaba)***；

⑶ **f $^{-1}$(aab)= Φ**；

5.2 RL的封闭性

⑷ $f^{-1}(aa)=\{0\}$；

⑸ $f^{-1}(\{aaa，aba，abaaaaa，abaaaaaa\})$

$=\{1，100\}$；

⑹ $f^{-1}((ab+ba)^*a)=\{1\}$；

⑺ $f(f^{-1}((ab+ba)^*a))=f(\{1\})=\{aba\}$。

令$L=(ab+ba)^*a$，上述(7)表明，$f(f^{-1}(L)) \neq L$

$f(f^{-1}(L)) \subseteq L$

5.2 RL的封闭性

推论5-1 RL 的同态像是 RL。

•证明：

注意到同态映射是正则代换的特例，可以直接的到此结论。

该定理表明， **RL** 在同态映射下是封闭的。

5.2 RL的封闭性

定理 5-5 RL 的同态原像是 RL 。

证明：

使用DFA作为描述工具。

(1) 接受RL的同态原像的FA的构造思想。

让新构造出的FA M′用一个移动去模拟M处理f(a)所用的一系列移动。

对于Σ中的任意字符a，如果M从状态q开始处理f(a)，并且当它处理完f(a)时到达状态p，则让M′在状态q读入a时，将状态变成p。

5.2 RL的封闭性

M′ 具有与M相同的状态，并且，在M′ 对应的状态转移图中，从状态q到状态p有一条标记为a的弧当且仅当在M的状态转移图中，有一条从状态q到状态p的标记为f(a)的路。

(2) 接受RL的同态原像的FA的形式描述。

设DFA M=(Q，Δ，δ，q_0，F)，L(M)=L，

取DFA M′ =(Q，Σ，δ′，q_0，F)

δ′ (q，a)= δ(q，f(a))

5.2 RL的封闭性

(3) 等价证明。

- 施归纳于|x|，证明对于$\forall x \in \Sigma^*$，

$$\delta'(q_0, x) = \delta(q_0, f(x))$$

当|x|=0时，结论显然成立。

设当|x|=k是结论成立，往证当|x|=k+1时结论成立。

不妨设x=ya，其中|y|=k

5.2 RL的封闭性

$\delta'(q_0, x) = \delta'(q_0, ya)$

$= \delta'(\delta'(q_0, y), a)$

$= \delta'(\delta(q_0, f(y)), a)$ 归纳假设

$= \delta(\delta(q_0, f(y)), f(a))$ δ' 的定义

$= \delta(q_0, f(y)f(a))$ δ 的意义

$= \delta(q_0, f(ya))$ 同态映射性质

$= \delta(q_0, f(x))$

5.2 RL的封闭性

这表明，结论对$|x|=k+1$成立。由归纳法原理，结论对$\forall x\in\Sigma^*$成立。

$\forall \mathbf{x}\in\Sigma^*$，$\delta'(\mathbf{q_0},\mathbf{x})\in\mathbf{F}\Leftrightarrow\delta(\mathbf{q_0},\mathbf{f(x)})\in\mathbf{F}$。

由于对$\forall x\in\Sigma^*$，$\delta'(q_0,x)=\delta(q_0,f(x))$，

所以，

$\delta'(q_0,x)\in F\Leftrightarrow\delta(q_0,f(x))\in F$。

故

$L(M')=f^{-1}(L(M))$

定理得证。

5.2 RL的封闭性

- **商(quotient)**
- 设L_1、$L_2 \subseteq \Sigma^*$，L_2除以L_1的商定义为：
  $L_1/L_2=\{x|\exists y \in L_2$使得$xy \in L_1\}$。
- 计算语言的商主要是考虑语言句子的后缀。只有当L_1的句子的后缀在L_2 中时，其相应的前缀才属于L_1/L_2。所以，当 $\varepsilon \in L_2$ 时， $L_1 \subseteq L_1/L_2$。

5.2 RL的封闭性

- 注意以下有意思的情况：

取L_1={000}，L_2={ ε }，L_3={ ε ， 0}

L_4={ ε ， 0，00}，L_5={ ε ， 0，00，000}

L_1/L_2={000}= L_1

L_1/L_3={000，00}

L_1/L_4={000，00，0}

L_1/L_5={000，00，0， ε }

5.2 RL的封闭性

定理5-6 L_1、$L_2 \subseteq \Sigma^*$，如果L_1是 RL ，则 L_1/L_2也是 RL 。

证明：设$L_1 \subseteq \Sigma^*$，是 RL ，

DFA $M=(Q, \Sigma, \delta, q_0, F)$，$L(M)=L_1$

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

$F'=\{q \mid \exists y \in L_2, \delta(q, y) \in F\}$

显然，

$L(M')= L_1/L_2$。

定理得证。

5.3 Myhill-Nerode 定理与DFA的极小化

- 对给定 **RL L**，**DFA M**接受**L**，**M**不同，由R_M确定的Σ^*上的等价类也可能不同。
- 如果**M**是最小**DFA**，则**M**所给出的等价类的个数应该是最少的。
- 最小**DFA**是不是惟一的？如果是，如何构造？
- 最小**DFA**的状态对应的集合与其他**DFA**的状态对应的集合有什么样的关系，用这种关系是否能从一般的**DFA**出发，求出最小**DFA**？

5.3.1 Myhill-Nerode 定理

- **DFA M确定的等价关系。**

$M=(Q, \Sigma, \delta, q_0, F)$，对于$\forall x, y \in \Sigma^*$

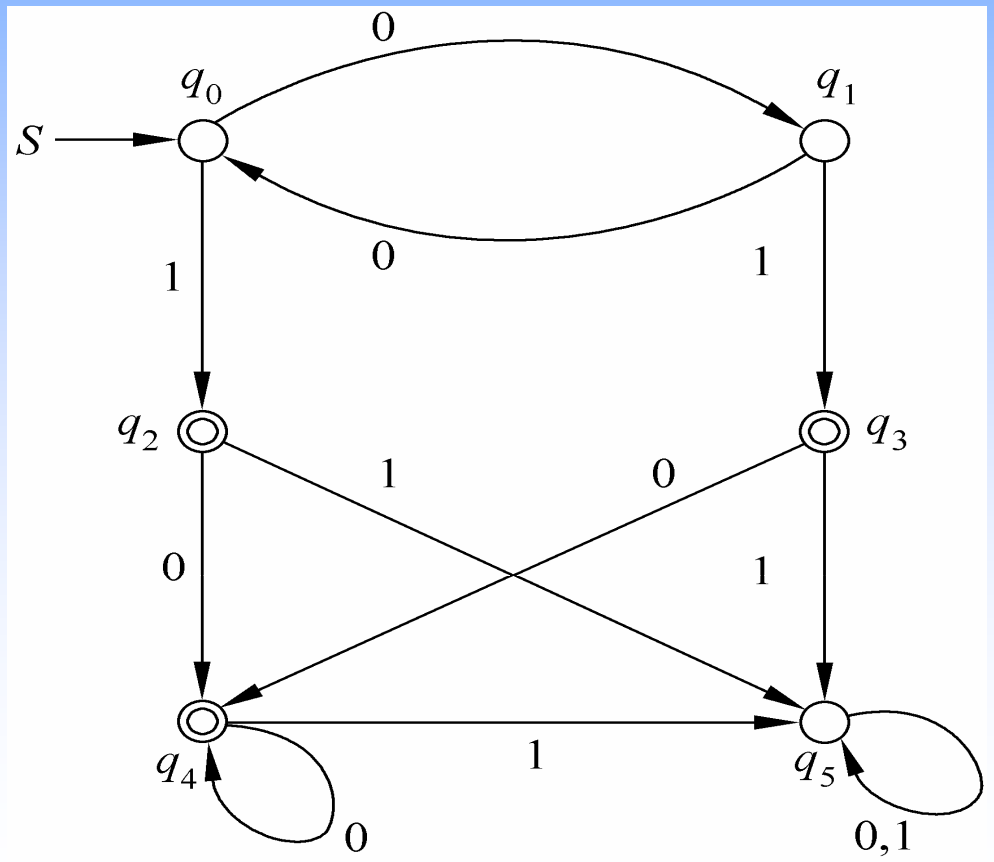
$x R_M y \Leftrightarrow \delta(q_0, x)= \delta(q_0, y)$。

显然，

$x R_M y \Leftrightarrow \exists q \in Q, x, y \in set(q)$

5.3.1 Myhill-Nerode 定理

- 例 5-8 设 L=0*10*，它对应的DFA M如下图。



5.3.1 Myhill-Nerode 定理

对应于q_0：$(00)^n R_M (00)^m$　　n，m≥0；

对应于q_1：$0(00)^n R_M 0(00)^m$　　n，m≥0；

对应于q_2：$(00)^n1 R_M (00)^m1$　　n，m≥0；

对应于q_3：$0(00)^n 1R_M 0(00)^m1$　　n，m≥0；

对应于q_4：$0(00)^n 10^k R_M 0(00)^m10^h$　　n，m≥0, k, h≥1；

$(00)^n10^k R_M (00)^m10^h$　　n, m≥0，k, h≥1；

$0(00)^n 10^k R_M (00)^m10^h$　　n, m≥0，k, h≥1；

也就是：　$0^n 10^k R_M 0^m10^h$　　n, m≥0，k, h≥1；

对应于q_5：$x R_M y$——x，y为至少含两个1的串。

5.3.1 Myhill-Nerode 定理

- L确定的Σ^*上的关系R_L。

  对于$\forall x, y \in \Sigma^*$，

  $x\ R_L\ y \Leftrightarrow$ (对$\forall z \in \Sigma^*$，$xz \in L \Leftrightarrow zy \in L$)

  对于$\forall x, y \in \Sigma^*$，如果$x\ R_L\ y$，则在x和y后无论接$\Sigma^*$中的任何串z，xz和yz要么都是L的句子，要么都不是L的句子。

5.3.1 Myhill-Nerode 定理

任意x，y∈set(q)，$\delta(q_0, x)=\delta(q_0, y)=q$

对于$\forall z\in\Sigma^*$，

$\delta(q_0, xz)=\delta(\delta(q_0, x), z))$

$=\delta(q, z)$

$=\delta(\delta(q_0, y), z)$

$=\delta(q_0, yz)$

这就是说，

$\delta(q_0, xz)\in F \Leftrightarrow \delta(q_0, yz)\in F$

5.3.1 Myhill-Nerode 定理

即，对于$\forall z \in \Sigma^*$，

$xz \in L \Leftrightarrow yz \in L$。

表明，

$x \, R_L \, y$，

也就是

$x \, R_{L(M)} y$。

5.3.1 Myhill-Nerode 定理

- **右不变的(right Invariant)**等价关系

  设R是Σ^*上的等价关系，对于$\forall x, y \in \Sigma^*$，如果$x \, R_L \, y$，则必有$xz \, R_L \, yz$对于$\forall z \in \Sigma^*$成立，则称$R$是**右不变的**等价关系。

5.3.1 Myhill-Nerode 定理

命题 5-1 对于任意DFA $M=(Q, \Sigma, \delta, q_0, F)$，M所确定的$\Sigma^*$上的关系$R_M$为右不变的等价关系。

证明：

(1) R_M是等价关系。

自反性显然 。

对称性：$\forall x, y \in \Sigma^*$,

$x\ R_M\ y \Leftrightarrow \delta(q_0, x) = \delta(q_0, y)$　　根据R_M的定义；

$\Leftrightarrow \delta(q_0, y) = \delta(q_0, z)$　　“=”的对称性；

$\Leftrightarrow y\ R_M\ x$　　根据R_M的定义。

5.3.1 Myhill-Nerode 定理

传递性：设$x\ R_M\ y$，$y\ R_M\ z$。

由于$x\ R_M\ y$，$\delta(q_0, x)=\delta(q_0, y)$

由于$y\ R_M\ z$，$\delta(q_0, y)=\delta(q_0, z)$

由“=”的传递性知，

$\delta(q_0, x)=\delta(q_0, z)$

再由R_M的定义得：

$x\ R_M\ z$

即R_M是等价关系。

5.3.1 Myhill-Nerode 定理

(2) R_M 是右不变的

设$x\ R_M\ y$。则 $\delta(q_0, x)=\delta(q_0, y)=q$

所以，对于$\forall z \in \Sigma^*$，

$\delta(q_0, xz)=\delta(\delta(q_0, x), z))$

$=\delta(q, z)$

$=\delta(\delta(q_0, y), z)$

$=\delta(q_0, yz)$

这就是说，$\delta(q_0, xz)=\delta(q_0, yz)$，再由$R_M$的定义，

$xz\ R_M\ yz$

所以，R_M 是右不变的等价关系。

5.3.1 Myhill-Nerode 定理

命题 5-2 对于任意$L \subseteq \Sigma^*$，L所确定的Σ^*上的关系R_L为右不变的等价关系 。

证明：

⑴ R_L是等价关系。

自反性显然。

对称性：不难看出：$x\, R_L\, y \Leftrightarrow (\text{对} \forall z \in \Sigma^*, xz \in L \Leftrightarrow yz \in L) \Leftrightarrow y\, R_L\, x$

5.3.1 Myhill-Nerode 定理

传递性：设$x\ R_L\ y$，$y\ R_L\ z$。

$$x\ R_L\ y \Leftrightarrow (\text{对}\forall w\in\Sigma^*,\ xw\in L \Leftrightarrow yw\in L)$$

$$y\ R_L\ z \Leftrightarrow (\text{对}\forall w\in\Sigma^*,\ yw\in L \Leftrightarrow zw\in L)$$

所以，

$$(\forall w\in\Sigma^*, xw\in L \Leftrightarrow yw\in L \ \text{且} yw\in L \Leftrightarrow zw\in L)$$

即：

$$(\text{对}\forall w\in\Sigma^*,\ xw\in L \Leftrightarrow zw\in L)$$

故：

$$x\ R_L\ z$$

即R_L是等价关系。

5.3.1 Myhill-Nerode 定理

⑵ R_L 是右不变的。

设$x\ R_L\ y$。由R_L的定义，对$\forall w$，$v \in \Sigma^*$，$xwv \in L \Leftrightarrow zwv \in L$，注意到v的任意性，知，

$xw\ R_L\ yw$。

所以，R_L是右不变的等价关系。

5.3.1 Myhill-Nerode 定理

- **指数(index)**
- 设**R**是Σ^*上的等价关系，则称$|\Sigma^*/R|$是**R**关于Σ^*的**指数**。简称为**R**的指数。简称Σ^*的关于**R**的一个等价类，也就是Σ^*/R的任意一个元素，为**R**的一个等价类

5.3.1 Myhill-Nerode 定理

- 例 5-9 图5-4 所给DFA M所确定的R_M的指数为6。R_M将Σ^*分成6个等价类：

  $set(q_0)=\{(00)^n \mid n \geqslant 0\}$；

  $set(q_1)=\{0(00)^n \mid n \geqslant 0\}$；

  $set(q_2)=\{(00)^n 1 \mid n，\ m \geqslant 0\}$；

  $set(q_3)=\{0(00)^n 1 \mid n \geqslant 0\}$；

  $set(q_4)=\{0^n 10^k \mid n \geqslant 0，\ k \geqslant 1\}$；

  $set(q_5)=\{x \mid x$为至少含两个1的串$\}$。

5.3.1 Myhill-Nerode 定理

- R_M是$R_{L(M)}$的“加细”(refinement)
  - $\forall x, y \in \Sigma^*$，如果$x\ R_M\ y$，必有$x\ R_{L(M)}\ y$成立。即对于任意DFA $M=(Q, \Sigma, \delta, q_0, F)$。

    $$|\Sigma^*/R_{L(M)}| \leqslant |\Sigma^*/R_M| \leqslant |Q|$$
  - 按照R_M中被分在同一等价类的串，在按照$R_{L(M)}$分类时，一定会被分在同一个等价类。
  - R_M对Σ^*的划分比$R_{L(M)}$对Σ^*的划分更“细”。称R_M是$R_{L(M)}$的“加细”(refinement)。

5.3.1 Myhill-Nerode 定理

- $\Sigma^*/R_M = \{set(q_0)，set(q_1)，set(q_2)，set(q_3)，set(q_4)，set(q_5)\}$

  （1）取$00 \in set(q_0)$，$000 \in set(q_1)$。

  对于任意的$x \in \Sigma^*$，当x含且只含一个1时，$00x \in L(M)$，$000x \in L(M)$；当x不含1或者含多个1时，$00x \notin L(M)$，$000x \notin L(M)$。这就是说，对于任意的$x \in \Sigma^*$，$00x \in L(M) \Leftrightarrow 000x \in L(M)$。即按照$R_{L(M)}$，00与000被分在同一个等价类中。从而$set(q_0)$和$set(q_1)$ 被包含在$R_{L(M)}$的同一个等价类中。

5.3.1 Myhill-Nerode 定理

⑵ 取$00 \in set(q_0)$，$001 \in set(q_2)$。

取特殊的字符串$1 \in \Sigma^*$，$001 \in L(M)$，但$0011 \notin L(M)$。所以，根据$R_{L(M)}$，$set(q_0)$和$set(q_2)$不能被“合并”到一个等价类中。

类似地，根据$R_{L(M)}$，$set(q_3)$、$set(q_4)$、$set(q_5)$也都不能被“合并”到$set(q_0)$的句子所在的等价类中。

5.3.1 Myhill-Nerode 定理

⑶ 取001∈set(q_2)，01∈set(q_3)。

对于任意的x∈Σ^*，x要么不含1，要么含有1。当x不含1时，001x∈L(M)，01x∈L(M)；当x含有1时，001x∉L(M)，01x∉L(M)。这就是说，对于任意的x∈Σ^*，001x∈L(M) ⇔ 01x∈L(M)。即按照$R_{L(M)}$，001与01属于$R_{L(M)}$的同一个等价类中。从而set(q_2)和set(q_3) 被包含在$R_{L(M)}$的同一个等价类中。

5.3.1 Myhill-Nerode 定理

⑷ 取$1 \in set(q_2)$， $10 \in set(q_4)$。

对于任意的$x \in \Sigma^*$，x要么不含1，要么含有1。当x不含1时，$1x \in L(M)$，$10x \in L(M)$；当x含有1时，$1x \notin L(M)$，$10x \notin L(M)$。这就是说，对于任意的$x \in \Sigma^*$，$1x \in L(M) \Leftrightarrow 10x \in L(M)$。即按照$R_{L(M)}$，1与10被分在$R_{L(M)}$的同一个等价类中。从而在$set(q_2)$和$set(q_4)$被包含在$R_{L(M)}$的同一个等价类中。

5.3.1 Myhill-Nerode 定理

⑸ 取$1 \in set(q_2)$， $11 \in set(q_5)$。

注意到$1\varepsilon = 1$，$11\varepsilon = 11$；而$1 \in L(M)$，$11 \notin L(M)$。即1和11不满足关系$R_{L(M)}$，所以，$set(q_2)$和$set(q_5)$不能被“合并”到$R_{L(M)}$的同一个等价类中。在这里，$\varepsilon \in \Sigma^*$是一个特殊的字符串。

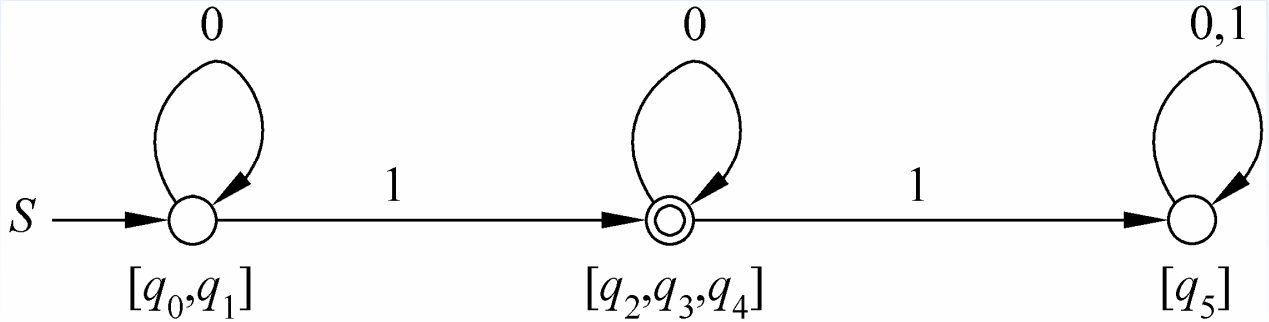
5.3.1 Myhill-Nerode 定理

$\Sigma^*/R_{L(M)}=\{\ set(q_0)\cup set(q_1),\ set(q_2)\cup set(q_3)\cup set(q_4),\ set(q_5)\}$

不含1：$[0]= set(q_0)\cup set(q_1)=0^*$；

含一个1：$[1]= set(q_2)\cup set(q_3)\cup set(q_4)=0^*10^*$；

含多个1：$[11]= set(q_5)=0^*10^*1(0+1)^*$ 。



5.3.1 Myhill-Nerode 定理

定理5-1 (Myhill-Nerode定理)如下三个命题等价：

⑴ $L \subseteq \Sigma^*$是 **RL**；

⑵ **L**是Σ^*上的某一个具有有穷指数的右不变等价关系**R**的某些等价类的并；

⑶ R_L具有有穷指数。

5.3.1 Myhill-Nerode 定理

证明：

- 由(1)可以推出(2)

设$L\subseteq\Sigma^*$是 RL，所以，存在DFA $M=(Q, \Sigma, \delta, q_0, F)$，使得$L(M)=L$。由命题5-3-1，$R_M$是$\Sigma^*$上的右不变等价关系，而且$|\Sigma^*/R_M|\leqslant|Q|$，所以，$R_M$具有有穷指数。而

$$L=\bigcup_{q\in F} set(q)$$

L是Σ^*上的具有有穷指数的右不变等价关系R_M的、对应于M的终止状态的等价类的并。

5.3.1 Myhill-Nerode 定理

•由(2)可以推出(3)。

设$x\ R\ y$，由R的右不变性可知，对于任意$z \in \Sigma^*$，

$xz\ R\ yz$

而L是R的某些等价类的并，所以，

$xz \in L \Leftrightarrow yz \in L$

根据R_L的定义，

$x\ R_L\ y$

故R是R_L的加细。由于R具有有穷指数，所以，R_L具有有穷指。

5.3.1 Myhill-Nerode 定理

- 由(3)可以推出(1)。

令$M'=(\Sigma^*/R_L, \Sigma, \delta', [\varepsilon], \{[x]|x\in L\})$

[ε]表示ε所在的等价类对应的状态；

[x] 表示x所在的等价类对应的状态。

对于$\forall([x], a)\in(\Sigma^*/R_L)\times\Sigma$，$\delta'([x], a)=[xa]$

- δ' 定义的相容性
- $L(M')=L$

5.3.1 Myhill-Nerode 定理

- **例5-10** 用定理5-7证明$\{0^n1^n|n\geqslant 0\}$不是 RL
  - 根据L的句子的特征来寻找R_L的等价类。
  - L的句子的主要特点有两个：

  (1) 句子中所含的字符0的个数与所含的字符1的个数相同。

  (2) 所有的0都在所有的1的前面
  - 可以得到如下一些等价类。

5.3.1 Myhill-Nerode 定理

$[10]=\{x \mid x=0^n1^m(m \geqslant n+1)$或者x中含子串10$\}$

[ε]—— ε 所在的等价类；

[1]——0所在的等价类；

[2]——00所在的等价类；

[3]——000所在的等价类；

…

[n]——0^n所在的等价类；

…

所以，R_L的指数是无穷的。因此，**L**不是 **RL**。

5.3.1 Myhill-Nerode 定理

推论 5-1 对于任意的 **RL L**，如果**DFA M=(Q，Σ，δ，q_0，F)**满足**L(M)=L**，则 $|\Sigma^*/R_L| \leqslant |Q|$。

- 表明，对于任意**DFA M=(Q，Σ，δ，q_0，F)**，$|Q| \geqslant |\Sigma^*/R_{L(M)}|$。
- 也表明，对任意一个 **RL L**，按照定理中所给的方法构造出来的**DFA M′**是一个接受**L**的最小**DFA**。这个**DFA**是惟一的么？

5.3.1 Myhill-Nerode 定理

推论5-2 对于任意的 **RL L**，在同构意义下，接受**L**的最小**DFA**是惟一的。

证明：

- 接受L的最小DFA $M=(Q, \Sigma, \delta, q_0, F)$的状态数与$R_L$的指数相同，也就是说，这个最小DFA的状态数与Myhill-Nerode定理证明中构造的$M'=(\Sigma^*/R_L, \Sigma, \delta', [\varepsilon], \{[x] \mid x \in L\})$的状态数是相同的。

5.3.1 Myhill-Nerode 定理

- **DFA同构是指这两个DFA的状态之间有一个一一对应，而且这个一一对应还保持状态转移也是相应一一对应的。也就是说，如果q与[w]对应，p与[z]对应，当δ(q，a)=p时，必定有δ([w]，a)=[z]。**
- **这两个DFA是同构。定义映射f**

f(q)=f(δ(q$_0$，x))=δ′([ε]，x)=[x]

⇔δ(q$_0$，x)=q

5.3.1 Myhill-Nerode 定理

- **f为Q与Σ^*/R_L之间的一一对应**
  - 如果$\delta(q_0, x)=\delta(q_0, y)$，则$x\ R_M\ y$
  - 由于R_M是R_L的加细，所以，$x\ R_L\ y$
  - 故，$[x]=[y]$，即，$\delta'([\varepsilon], x)=\delta'([\varepsilon], y)$。
  - 如果，$\delta(q_0, x)\neq\delta(q_0, y)$
  - 则，$\delta'([\varepsilon], x)\neq\delta'([\varepsilon], y)$
  - 即，$[x]\neq[y]$
  - 否则，$|\Sigma^*/R_M|>|\Sigma^*/R_L|$。

5.3.1 Myhill-Nerode 定理

- 如果δ(q，a)=p，f(q)= [x]，必有f(p)=[xa]
  - ∀ q∈Q，如果，f(q)=f(δ(q_0，x))=[x]
  - 所以，∀ a∈Σ，如果，
  - p=δ(q，a)=δ(δ(q_0，x)，a)=δ(q_0，xa)
  - 则 f(p)=f(δ(q ， a))= f(δ(δ(q_0 ， x) ， a))=f(δ(q_0，xa))=[xa]
  - 即，如果M在状态q读入字符a时进入状态p，则M在q对应的状态f(δ(q_0，x))=[x]读入字符a时，进入p对应的状态f(δ(q_0，xa))=[xa]。所以，f是M和M′之间的同构映射。

5.3.2 DFA的极小化

- **可以区分的(distinguishable)** 状态
- 设**DFA M=(Q，Σ，δ，q_0，F)**，如果∃**x**∈Σ*，对**Q**中的两个状态**q**和**p**，使得**δ(q，x)∈F** 和 **δ(p，x)∈F**中，有且仅有一个成立，则称**p**和**q**是**可以区分的**。否则，称**q**和**p**等价。并记作**q≡p** 。

5.3.2 DFA的极小化

算法5-1 DFA的极小化算法

- 算法思想：扫描所有的状态对，找出所有的可区分的状态对，不可取分的状态对一定是等价的。
- 输入：给定的**DFA**。
- 输出：可区分状态表。
- 主要数据结构：状态对的关联链表；可区分状态表。

5.3.2 DFA的极小化

- 主要步骤

⑴ **for** $\forall(q, p)\in F\times(Q-F)$ **do**

标记可区分状态表中的表项**(q，p)**；

⑵ **for**$\forall(q, p)\in F\times F\cup(Q-F)\times(Q-F)\&q\neq p$ **do**

⑶ **if** $\exists a\in\Sigma$，可区分状态表中的表项(δ**(q，a)**，δ**(p，a)**)已被标记 **then**

begin

⑷ 标记可区分状态表中的表项**(q，p)**；

⑸ 递归地标记本次被标记的状态对的关联链表上的各个状态对在可区分状态表中的对应表项

end

5.3.2 DFA的极小化

(6) **else for** $\forall a \in \Sigma$，**do**

(7) **if** $\delta(q, a) \neq \delta(p, a)$ **&**(q, p) 与 $(\delta(q, a), \delta(p, a))$不是同一个状态对 **then**

将(q, p)放在$(\delta(q, a), \delta(p, a))$的关联链表上。

5.3.2 DFA的极小化

定理5-8 对于任意DFA M=(Q，Σ，δ，q_0，F)，Q中的两个状态q和p是可区分的充要条件是(q，p)在DFA的极小化算法中被标记。

证明：

先证必要性。

设q和p是可区分的，x是区分q和p的最短字符串。现施归纳于x的长度，证明(q，p)一定被算法标记。

5.3.2 DFA的极小化

- 当|x|=0时

ε 区分q和p，表明q和p有且仅有一个为M的终止状态，所以，

(q，p)∈F×(Q-F)

因此，它在算法的第(1)行被标记。

- 设当|x|=n时结论成立

x是区分q和p的长度为n的字符串，则(q，p)被算法标记。

5.3.2 DFA的极小化

- 当|x|=n+1时

设x=ay，其中|y|=n。由于x是区分q和p的最短的字符串，所以，$\delta(q, x)\in F$ 和 $\delta(p, x)\in F$中，有且仅有一个成立。不妨假设：

$\delta(q, x)\notin F$ ， $\delta(p, x)\in F$

即 $\delta(\delta(q, a), y)\notin F$ ， $\delta(\delta(p, a), y)\in F$

设 $\delta(q, a)=u$， $\delta(p, a)=v$

y是区分u和v的长度为n的字符串。

5.3.2 DFA的极小化

由归纳假设，(u，v)可以被算法标记。

- 如果在考察(q，p)时，(u，v)已经被标记，则(q，p)在算法的第(4)行被标记；
- 如果在考察(q，p)时，(u，v)还没有被标记，则(q，p)在算法的第(7)行被放入到(u，v)的关联链表中，而当(u，v)被标记时，在算法的第(5)行在“递归”过程中(q，p)被标记。
- 结论对|x|=n+1成立。

5.3.2 DFA的极小化

- 充分性。
- 设(q，p)在算法中被标记。对它被标记的顺序n施归纳，证明q和p是可区分的。
- 令|F×(Q-F)|=m，显然，当1≤n≤m时，(q，p)是在算法的第(1)行被标记的，此时，ε是区分q和p的字符串：

$\delta(q, \varepsilon) \in F$ 和 $\delta(p, \varepsilon) \in F$

有且仅有一个成立。

5.3.2 DFA的极小化

- 设n≤k(k≥m)时结论成立。即，如果 (q，p)是被算法在第k个或者第k个之前标记的，则存在字符串x，x区分q和p。即：δ(q，x)∈F 和 δ(p，x)∈F有且仅有一个成立。
- 当n=k+1时，如果(q，p)是在算法的第(4)行被标记的，此时，(δ(q，a)，δ(p，a))一定是在第k个之前被标记的。设δ(q，a)=u，δ(p，a)=v，由归纳假设，存在字符串x，x区分u和v:
- δ(u，x)∈F 和 δ(v，x)∈F
- 有且仅有一个成立，从而，
- δ(q，ax)∈F 和 δ(p，ax)∈F
- 有且仅有一个成立。即，ax是区分q和p的字符串。

5.3.2 DFA的极小化

- 如果(q，p)是在算法的第(5)行被标记的，则它必在某个状态对(u，v)的关联链表中，而(u，v)必在(q，p)之前被标记。由归纳假设，存在x区分(u，v)；
- 存在a∈∑，δ(q，a)=u，δ(p，a)=v使得(q，p)被放在(u，v)的关联链表中；ax是区分q和p的字符串。
- 所以，结论对n=k+1成立。由归纳法原理，结论对所有的n成立。

5.3.2 DFA的极小化

定理5-9 由算法5-1构造的DFA在去掉不可达状态是最小DFA 。

证明：

设$M=(Q,\Sigma,\delta,q_0,F)$为算法5-1的输入DFA，$M'=(Q/\equiv,\Sigma,\delta',[q_0],F')$是相应的输出DFA。$F'=\{[q]|q\in F\}$。

对于$\forall [q]\in Q/\equiv$，$\forall a\in\Sigma$，定义

$\delta'([q],a)=[\delta(q,a)]$

5.3.2 DFA的极小化

- δ′ 的相容性。
  - 设[q]=[p]，也就是说，q和p等价：q≡p。即根据算法5-1，状态q和p是不可区分的(未被算法标记)。此时，对于∀ a∈Σ，必须有[δ(q，a)]≡[δ(p，a)]。否则，状态对(δ(q，a)，δ(p，a))必定被算法标记，从而最终导致(q，p)被算法标记。此与q≡p矛盾。所以，状态[δ(q，a)]和状态[δ(p，a)] 等价：δ(q，a)= δ(p，a)。所以，δ′ 的定义是相容的。

5.3.2 DFA的极小化

- **L(M′)=L(M)**。
  - 对∀**x**∈Σ*，现施归纳于**|x|**，证明**δ′([q_0]，x)=[δ(q_0，x)]**
  - **|x|=0**
    - **δ′([q_0]，ε)=[q_0]=[δ(q_0，ε)]**
  - ∀ x∈Σ*并且**|x|=n**，
    - **δ′([q_0]，xa)=δ′(δ′([q_0]，x)，a)**
    - **=δ′([δ(q_0，x)]，a)**
    - **=[δ([δ(q_0，x)]，a)]**
    - **=[δ(q_0，xa)]**
  - 由归纳法原理，结论对∀**x**∈Σ*成立。

5.3.2 DFA的极小化

- 再由F′ 的定义，
- $\delta'([q_0], x)=[\delta(q_0, x)]\in F' \Leftrightarrow \delta(q_0, x)\in F$。
- 所以，
- $x\in L(M') \Leftrightarrow x\in L(M)$。
- 即：
- $L(M')=L(M)$。

5.3.2 DFA的极小化

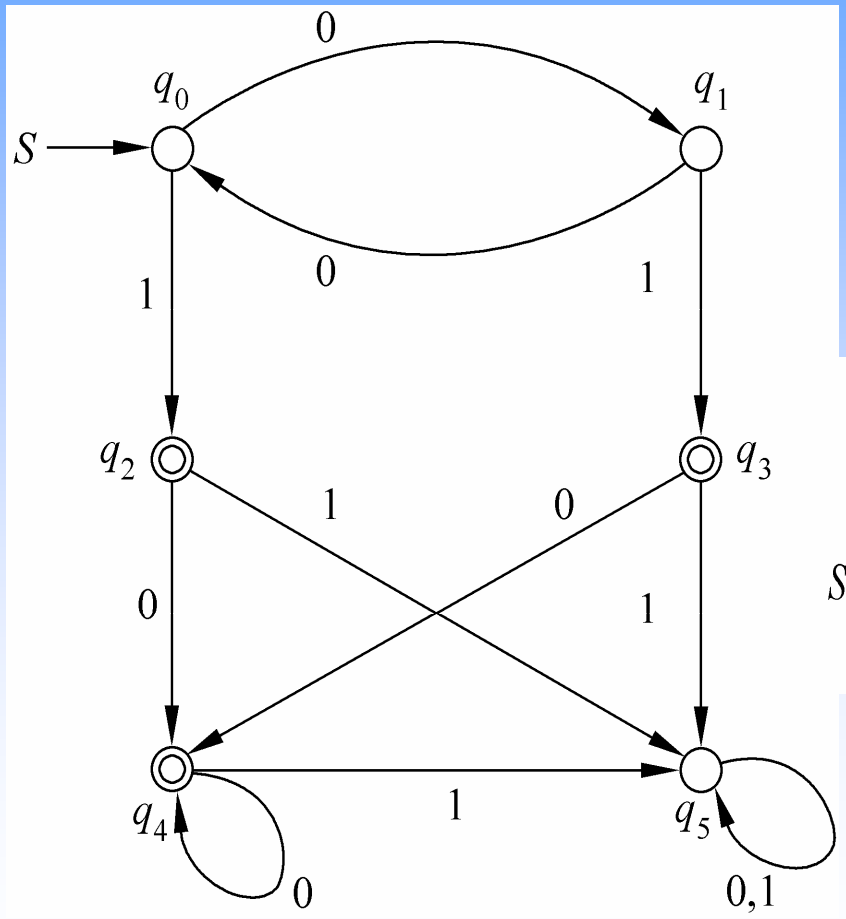
- 证明所构造的M′去掉不可达状态后是最小DFA。
  - 如果[q]≠[p]，则对于∀x∈set([q])，∀y∈set([p])，x R_L y不成立。事实上，如果[q]≠[p]，则存在z∈Σ*，z区分q和p，有δ(q，z)=q′ 和δ(p，z)= p′有且仅有一个是终止状态，这就是说，xz和yz有且仅有一个是L的句子。所以，x R_L y是不成立的。

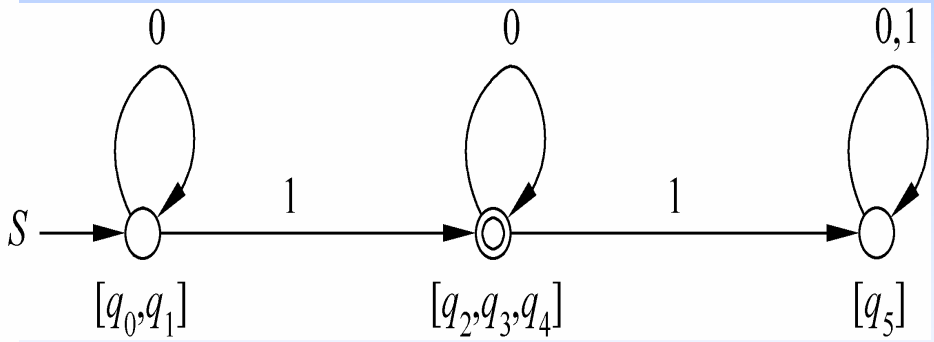
5.3.2 DFA的极小化

- **例5-11** 用算法5-1对图5-4所给的DFA进行极小化。

q_1					
q_2	×	×			
q_3	×	×			
q_4	×	×			
q_5	×	×	×	×	×
	q_0	q_1	q_2	q_3	q_4

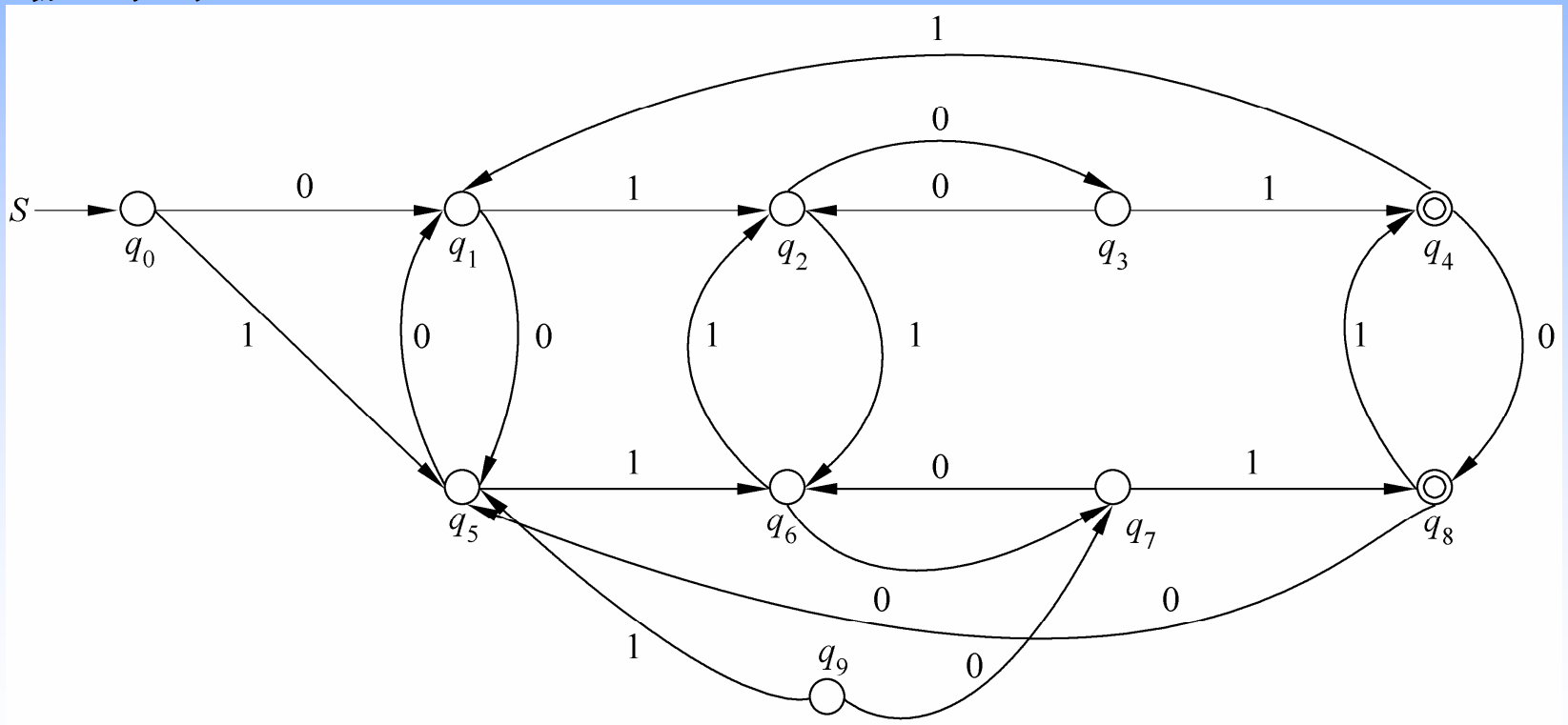
5.3.2 DFA的极小化





5.3.2 DFA的极小化

- **例5-11** 用算法5-1对图5-7所给的DFA进行极小化。



5.3.2 DFA的极小化

q_1	×							
q_2	×	×						
q_3	×	×	×					
q_4	×	×	×	×				
q_5	×	×	×	×	×			
q_6	×	×	×	×	×	×		
q_7	×	×	×	×	×	×	×	
q_8	×	×	×	×	×	×	×	×
	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7

5.4 关于正则语言的判定算法

定理 5-10 设**DFA M=(Q，∑，δ，q_0，F)，L=L(M)**非空的充分必要条件是：存在 $x\in\Sigma^*$，$|x|<|Q|$，$\delta(q_0, x)\in F$。

- 证明：充分性显然。
- 必要性：**M**的状态转移图中必存在一条从q_0到某一个终止状态q_f且无重复状态的路，此路中的状态数$n\leqslant|Q|$。此路的标记**x**满足$|x|\leqslant n-1$。而$\delta(q_0, x)\in F$。即**x**是**L=L(M)**的长度小于**|Q|**的句子。

5.4 关于正则语言的判定算法

定理5-11设DFA $M=(Q, \Sigma, \delta, q_0, F)$，$L=L(M)$为无穷的充分必要条件是：存在$x \in \Sigma^*$，$|Q| \leqslant |x| < 2|Q|$，$\delta(q_0, x) \in F$。

- 算法通过判定是否存在$x \in \Sigma^*$，$|Q| \leqslant |x| < 2|Q|$，$\delta(q_0, x) \in F$即可。

5.4 关于正则语言的判定算法

定理 5-12 设DFA $M_1=(Q_1, \Sigma, \delta_1, q_{01}, F_1)$，DFA $M_2=(Q_2, \Sigma, \delta_2, q_{02}, F_2)$，则存在判定$M_1$与 M_2是否等价的算法。

- 通过判定两个**DFA**的极小**DFA**是否同构就可以判定它们是否等价。

5.4 关于正则语言的判定算法

定理 5-13 设L是字母表Σ上的 RL，对任意$x \in \Sigma^*$，存在判定x是不是L的句子的算法。

- 从一定的意义上讲，接受L的DFA M就是判定x是否L的一个桔子的“算法”。

5.5 小结

本章讨论了RL的性质。包括： RL 的泵引理，RL 关于并、乘积、闭包、补、交、正则代换、同态、逆同态等运算的封闭性。Myhill-Nerode定理与FA的极小化。

(1) 泵引理。泵引理是用RL的必要条件来用来证明一个语言不是 RL 的。它不能用来证明一个语言是 RL，而且是采用反证法。

5.5 小结

⑵ **RL** 对有关运算的封闭性。**RL** 在并、乘、闭包、补、交、正则代换、同态映射运算下是有效封闭的。**RL** 的同态原像是 **RL** 。

⑶ 设 L_1、$L_2 \subseteq \Sigma^*$，如果L_1是 **RL** ，则L_1/L_2也是 **RL** 。

5.5 小结

(4) 如果L是RL，则根据R_L确定的Σ^*的等价类可以构造出接受L的最小DFA。更方便的方法是通过确定给定DFA状态的可区分性构造出等价的最小DFA。

(5) 存在判定L(M)是非空、M_1与 M_2是否等价、L(M)是否无穷、x是不是RL L的句子的算法。

第6章 上下文无关语言

- G_{bra}：$S \rightarrow S(S) \mid \varepsilon$
- $L(G_{bra})$不是RL,是CFL

$$0^{n_1}1^{n_1}0^{n_2}1^{n_2}\ldots\ldots0^{n_h}1^{n_h}$$

•高级程序设计语言的绝大多数语法结构都可以用上下文无关文法(CFG)描述。。

•BNF(巴科斯范式：Backus normal form，又叫Backus-naur form)。

第6章 上下文无关语言

- 主要内容
  - 关于**CFL**的分析
    - 派生和归约、派生树
  - **CFG**的化简
    - 无用符、单一产生式、空产生式
  - **CFG**的范式
    - **CNF**
    - **GNF**
  - **CFG**的自嵌套特性

第6章 上下文无关语言

- 重点
  - **CFG的化简。**
  - **CFG到GNF的转换。**
- 难点
  - **CFG到GNF的转换，特别是其中的用右递归替换左递归的问题。**

6.1 上下文无关语言

- 文法**G=(V，T，P，S)**被称为是上下文无关的。 如果除了形如**A**→ ε 的产生式之外，对于∀ α → β ∈**P**，均有| β |≥| α |，并且 α ∈**V**成立。
- 关键：对于∀**A**∈**V**，如果**A**→ β ∈**P**，则无论**A**出现在句型的任何位置，我们都可以将**A**替换成 β，而不考虑**A**的上下文。

6.1.1 上下文无关文法的派生树

- 算术表达式的文法

G_{exp1}：E→E+T|E-T|T

T→T*F|T/F|F

F→F↑P|P

P→(E)|N(L)|id

N→sin|cos|exp|abs|log|int

L→L,E|E

6.1.1 上下文无关文法的派生树

- 算术表达式x+x/y↑2的不同派生

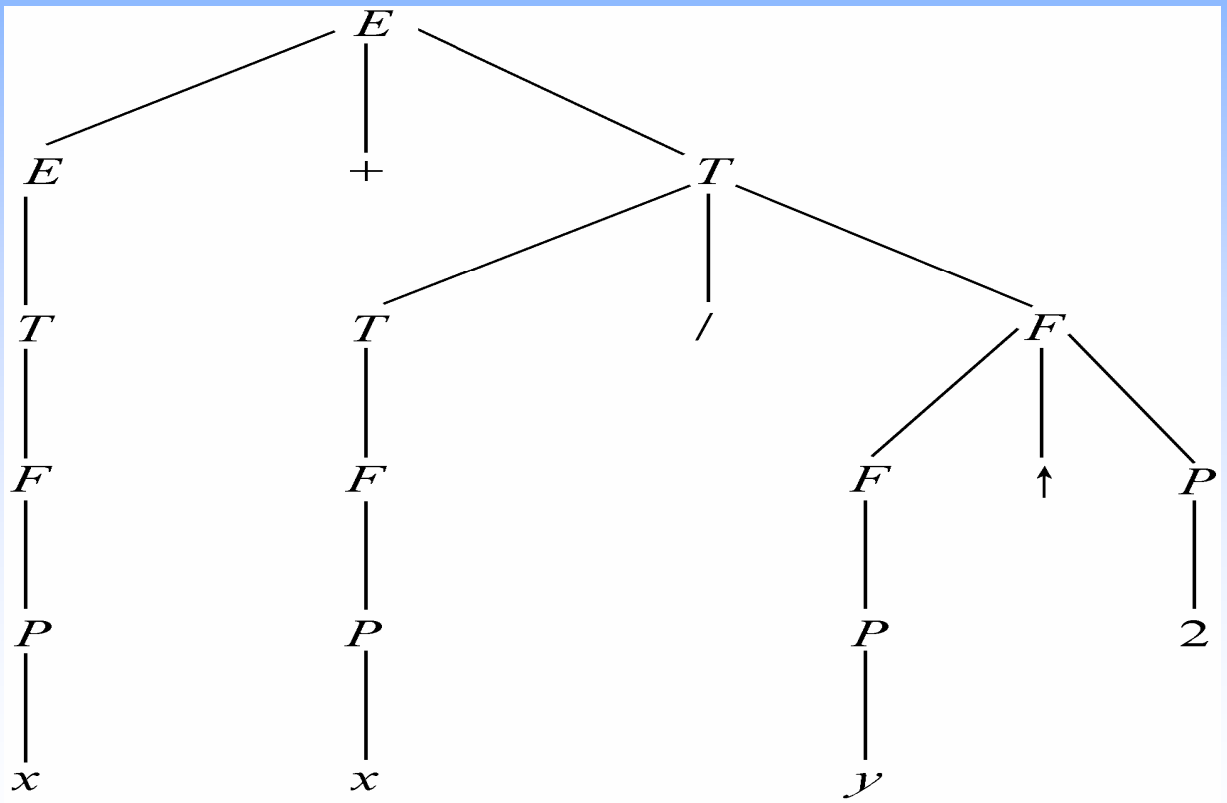
E⇒E+T⇒T+T⇒F+T⇒P+T⇒x+T⇒x+T/F⇒x+F/F
⇒x+P/F⇒x+x/F⇒x+x/F↑P⇒x+x/P↑P⇒x+x/y↑P
⇒x+x/y↑2

E⇒E+T⇒E+T/F⇒E+T/F↑P⇒E+T/F↑2⇒E+T/P↑2
⇒E+T/y↑2⇒ E+F/y↑2⇒ E+P/y↑2⇒ E+x/y↑2
⇒ T+x/y↑2⇒ F+x/y↑2⇒ P+x/y↑2⇒x+x/y↑2

E⇒E+T⇒T+T⇒T+T/F⇒F+T/F⇒F+T/F↑P
⇒P+T/F↑P⇒x+T/F↑P⇒x+F/F↑P⇒x+F/F↑2
⇒x+F/P↑2⇒x+P/P↑2⇒x+P/y↑2⇒x+x/y↑2

6.1.1 上下文无关文法的派生树

- 文法G_{exp1}句子x+x/y↑2的结构。



6.1.1 上下文无关文法的派生树

- 派生树(derivation tree)
  - 一棵(有序)树(ordered tree)
  - 树的每个顶点有一个标记X，且$X \in V \cup T \cup \{\varepsilon\}$
  - 树根的标记为S；
  - 如果非叶子顶点v标记为A，v的儿子从左到右依次为v_1，v_2，…，v_n，并且它们分别标记为X_1，X_2，…，X_n，则$A \to X_1X_2\ldots X_n \in P$；
  - 如果X是一个非叶子顶点的标记，则$X \in V$；
  - 如果顶点v标记为 ε，则v是该树的叶子，并且v是其父顶点的惟一儿子。

6.1.1 上下文无关文法的派生树

- 别称
  - **生成树**
  - **分析树(parse tree)**
  - **语法树(syntax tree)**
- 顺序
  - v_1，v_2是派生树T的两个不同顶点，如果存在顶点v，v至少有两个儿子，使得v_1是v的较左儿子的后代，v_2是v的较右儿子的后代，则称顶点v_1在顶点v_2的左边，顶点v_2在顶点v_1的右边。

6.1.1 上下文无关文法的派生树

- **结果(yield)**
- 派生树T的所有叶子顶点从左到右依次标记为X_1，X_2，…，X_n，则称符号串$X_1X_2…X_n$是T的**结果**。
- 一个文法可以有多棵派生树，它们可以有不同的结果。
- **句型α的派生树**：“G的结果为α的派生树”。

6.1.1 上下文无关文法的派生树

- **派生子树(subtree)**
- 满足派生树定义中除了对跟结点的标记的要求以外各条的树叫**派生子树(subtree)**。
- 如果这个子树的根标记为**A**，则称之为**A**子树。
- 惟一差别是根结点可以标记非开始符号。

6.1.1 上下文无关文法的派生树

定理6-1 设**CFG G=(V，T，P，S)，$S\Rightarrow^{*}\alpha$** 的充分必要条件为**G**有一棵结果为 α 的派生树。

证明：

- 证一个更为一般的结论：对于任意**$A\in V$**，**$A\Rightarrow^{*}\alpha$** 的充分必要条件为**G**有一棵结果为 α 的**A-**子树。
- 充分性：设**G**有一棵结果为 α 的**A-**子树，非叶子顶点的个数**n**施归纳，证明**$A\Rightarrow^{*}\alpha$** 成立。

6.1.1 上下文无关文法的派生树

- 设A-子树有k+1个非叶子顶点，根顶点A的儿子从左到右依次为v_1，v_2，…，v_m，并且它们分别标记为X_1，X_2，…，X_m。
- $A \rightarrow X_1X_2\ldots X_m \in P$。
- 以X_1，X_2，…，X_m为根的子树的结果依次为α_1，α_2，…，α_m。
- X_1，X_2，…，X_m为根的子树的非叶子顶点的个数均不大于k。

6.1.1 上下文无关文法的派生树

$X_1 \Rightarrow^* \alpha_1$

$X_2 \Rightarrow^* \alpha_2$

…

$X_m \Rightarrow^* \alpha_m$

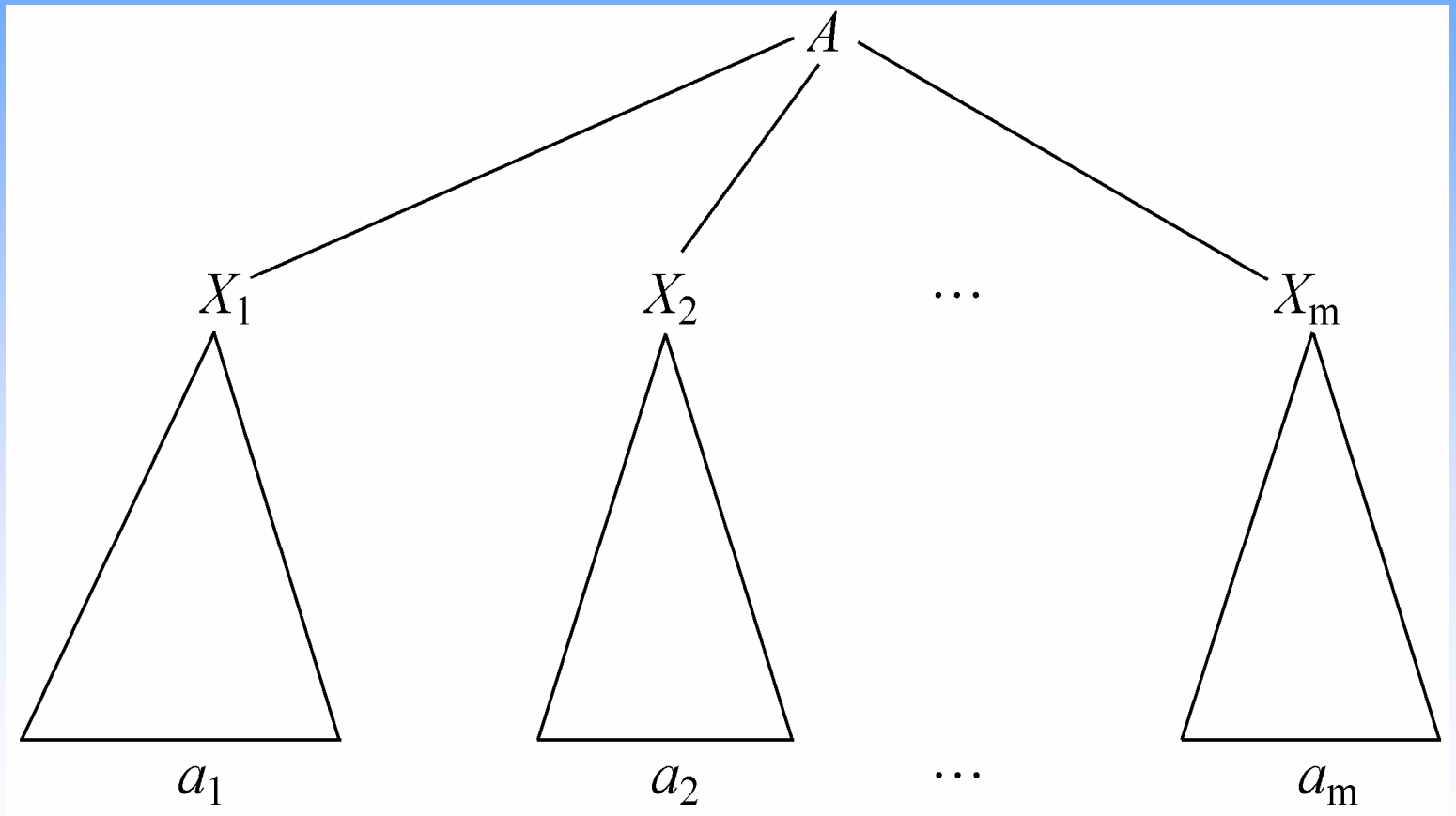
而且

$\alpha = \alpha_1 \alpha_2 \ldots \alpha_m$

6.1.1 上下文无关文法的派生树

$$A \Rightarrow X_1X_2\cdots X_m$$
$$\Rightarrow^* \alpha_1 X_2\cdots X_m$$
$$\Rightarrow^* \alpha_1 \alpha_2\cdots X_m$$
$$\cdots$$
$$\Rightarrow^* \alpha_1 \alpha_2\cdots \alpha_m$$

6.1.1 上下文无关文法的派生树



6.1.1 上下文无关文法的派生树

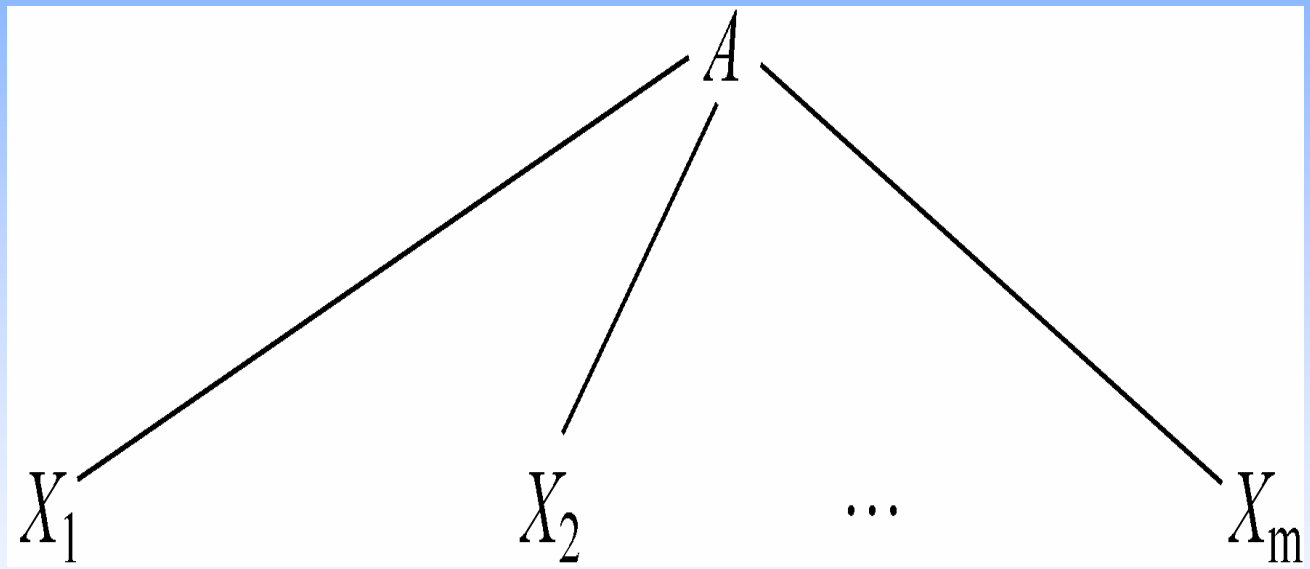
- 必要性
  - 设$A \Rightarrow^{n} \alpha$，现施归纳于派生步数n，证明存在结果为α的A-子树。

设$n \leqslant k (k \geqslant 1)$时结论成立，往证当n=k+1时结论也成立：

令$A \Rightarrow^{k+1} \alpha$，则有：

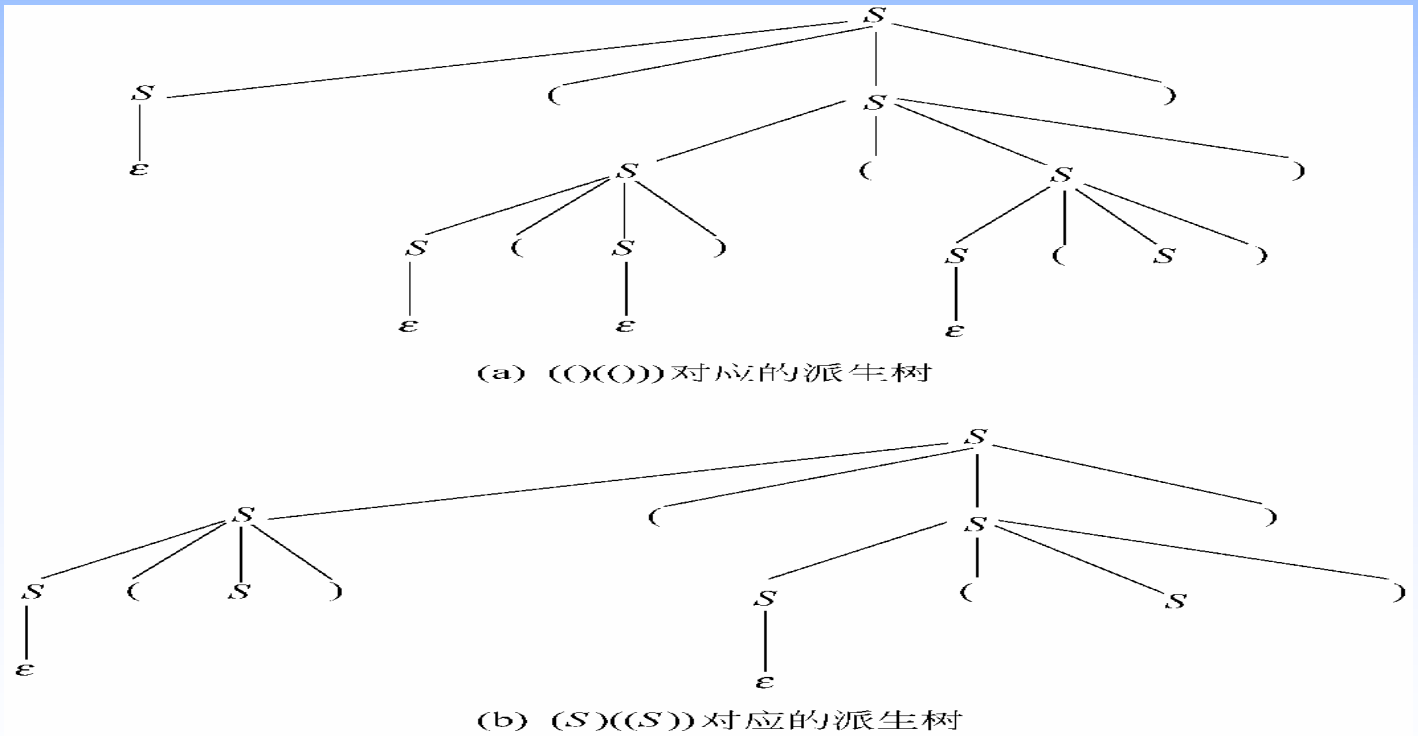
$$A \Rightarrow X_1X_2\ldots X_m$$
$$\Rightarrow^{*} \alpha_1 X_2\ldots X_m$$
$$\Rightarrow^{*} \alpha_1 \alpha_2\ldots X_m$$
$$\ldots$$
$$\Rightarrow^{*} \alpha_1 \alpha_2\ldots \alpha_m$$

6.1.1 上下文无关文法的派生树



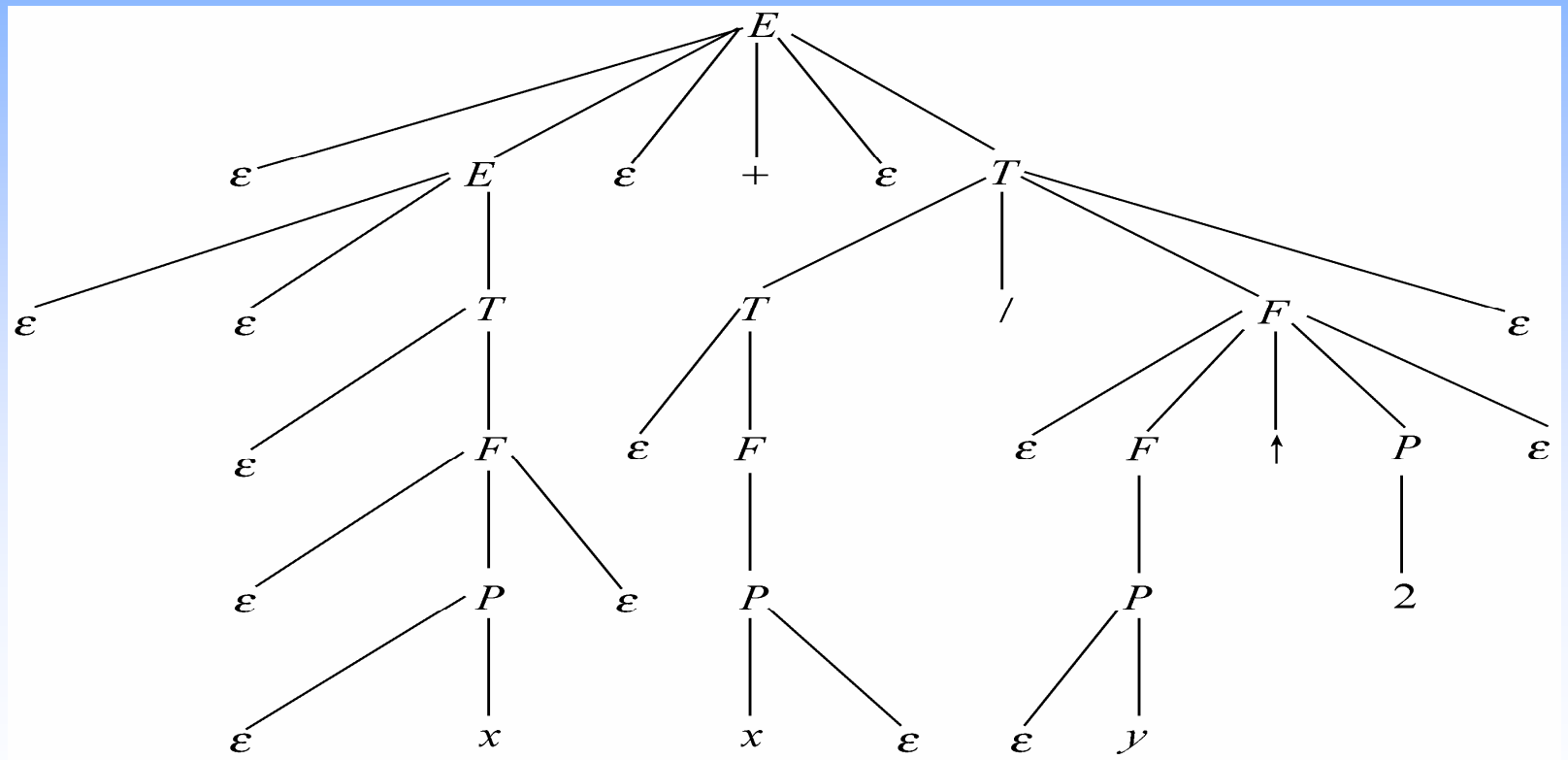
6.1.1 上下文无关文法的派生树

- 例6-1设G_{bra}：S→S(S)| ε，(()(()))和(S)((S))的派生树。



6.1.1 上下文无关文法的派生树

- 关于标记 ε 的结点



6.1.1 上下文无关文法的派生树

- **最左派生(leftmost derivation)**
  - α 的派生过程中，每一步都是对当前句型的最左变量进行替换。
- **左句型(left sentencial form)**
  - 最左派生得到的句型可叫做左句型。
- **最右归约**(rightmost reduction)
  - 与最左派生对相的归约叫做最有归约。

6.1.1 上下文无关文法的派生树

- **最右派生(rightmost derivation)**
  - α 的派生过程中，每一步都是对当前句型的最右变量进行替换。
- **右句型(right sentencial form)**
  - 最右派生得到的句型可叫做右句型。
- **最左归约（leftmost reduction）**
  - 与最左派生对相的归约叫做最右归约。

6.1.1 上下文无关文法的派生树

- **规范派生(normal derivation)**
  - 最右派生。
- **规范句型(normal sentencial form)**
  - 规范派生产生的句型。
- **规范归约(normal reduction)**
  - 最左归约。

6.1.1 上下文无关文法的派生树

定理6-2 如果α是**CFG G**的一个句型，则**G**中存在α的最左派生和最右派生。

证明：

基本思路：对派生的步数**n**施归纳，证明对于任意$A \in V$，如果$A \Rightarrow^{n} \alpha$，在**G**中，存在对应的从**A**到α的最左派生：$A \Rightarrow^{n左} \alpha$。

6.1.1 上下文无关文法的派生树

$A \Rightarrow X_1X_2\cdots X_m$

$\Rightarrow^{*} \alpha_1 X_2\cdots X_m$

$\Rightarrow^{*} \alpha_1 \alpha_2\cdots X_m$

$\cdots$

$\Rightarrow^{*} \alpha_1 \alpha_2\cdots \alpha_m$

$A \Rightarrow^{左} X_1X_2\cdots X_m$

$\Rightarrow^{*左} \alpha_1 X_2\cdots X_m$

$\Rightarrow^{*左} \alpha_1 \alpha_2\cdots X_m$

$\cdots$

$\Rightarrow^{*左} \alpha_1 \alpha_2\cdots \alpha_m$

同理可证，句型 α 有最右派生。

6.1.1 上下文无关文法的派生树

定理6-3 如果α是**CFG G**的一个句型，α的派生树与最左派生和最右派生是一一对应的，但是，这棵派生树可以对应多个不同的派生。

6.1.2 二义性

- 简单算术表达式的二义性文法

G_{exp2}：

E→E+E|E-E|E/E|E*E

E→ E ↑ E|(E)|N(L)|id

N→sin|cos|exp|abs|log|int

L→L,E|E

6.1.2 二义性

句子x+x/y ↑ 2在文法中的三个不同的最左派生

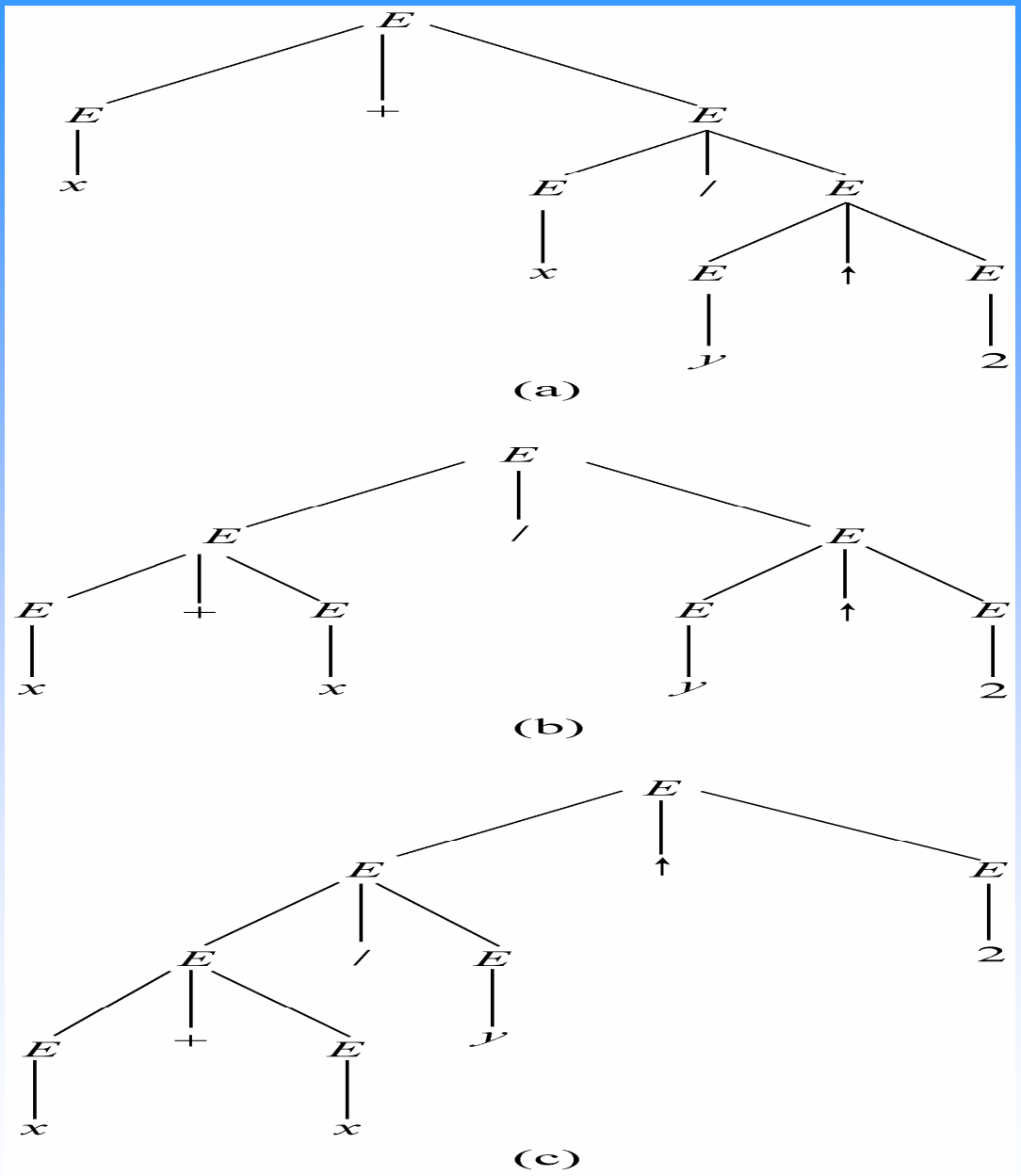
$E \Rightarrow E+E$
$\Rightarrow x+E$
$\Rightarrow x+E/E$
$\Rightarrow x+x/E$
$\Rightarrow x+x/E \uparrow E$
$\Rightarrow x+x/y \uparrow E$
$\Rightarrow x+x/y \uparrow 2$

$E \Rightarrow E/E$
$\Rightarrow E+E/E$
$\Rightarrow x+E/E$
$\Rightarrow x+x/E$
$\Rightarrow x+x/E \uparrow E$
$\Rightarrow x+x/y \uparrow E$
$\Rightarrow x+x/y \uparrow 2$

$E \Rightarrow E \uparrow E$
$\Rightarrow E/E \uparrow E$
$\Rightarrow E+E/E \uparrow E$
$\Rightarrow x+E/E \uparrow E$
$\Rightarrow x+x/E \uparrow E$
$\Rightarrow x+x/y \uparrow E$
$\Rightarrow x+x/y \uparrow 2$

6.1.2 二义性

对应3个不同的语法树



6.1.2 二义性

- **二义性(ambiguity)**
- CFG G=(V，T，P，S)，如果存在w∈L(G)，w至少有两棵不同的派生树，则称G是**二义性的**。否则，G为非二义性的。
- 二义性的问题是**不可解的**(unsolvable)**问题**。

6.1.2 二义性

- **例6-2** 用其他方法消除二义性。

G_{ifa}： **S→if E then S else S | if E then S**

G_{ifm}： **S→U|M**

U→if E then S

U→if E then M else U

M→if E then M else M|S

G_{ifh}： **S→TS|CS**

C→if E then

T →CS else

6.1.2 二义性

- **例 6-3** 设

$L_{ambiguity}=\{0^n1^n2^m3^m|n,m\geqslant 1\}\cup\{0^n1^m2^m3^n|n,m\geqslant 1\}$

可以用如下文法产生语言$L_{ambiguity}$：

G：S→AB|0C3

A→01|0A1

B→23|2B3

C→0C3|12|1D2

D→12|1D2

语言$L_{ambiguity}$不存在非二义性的文法。

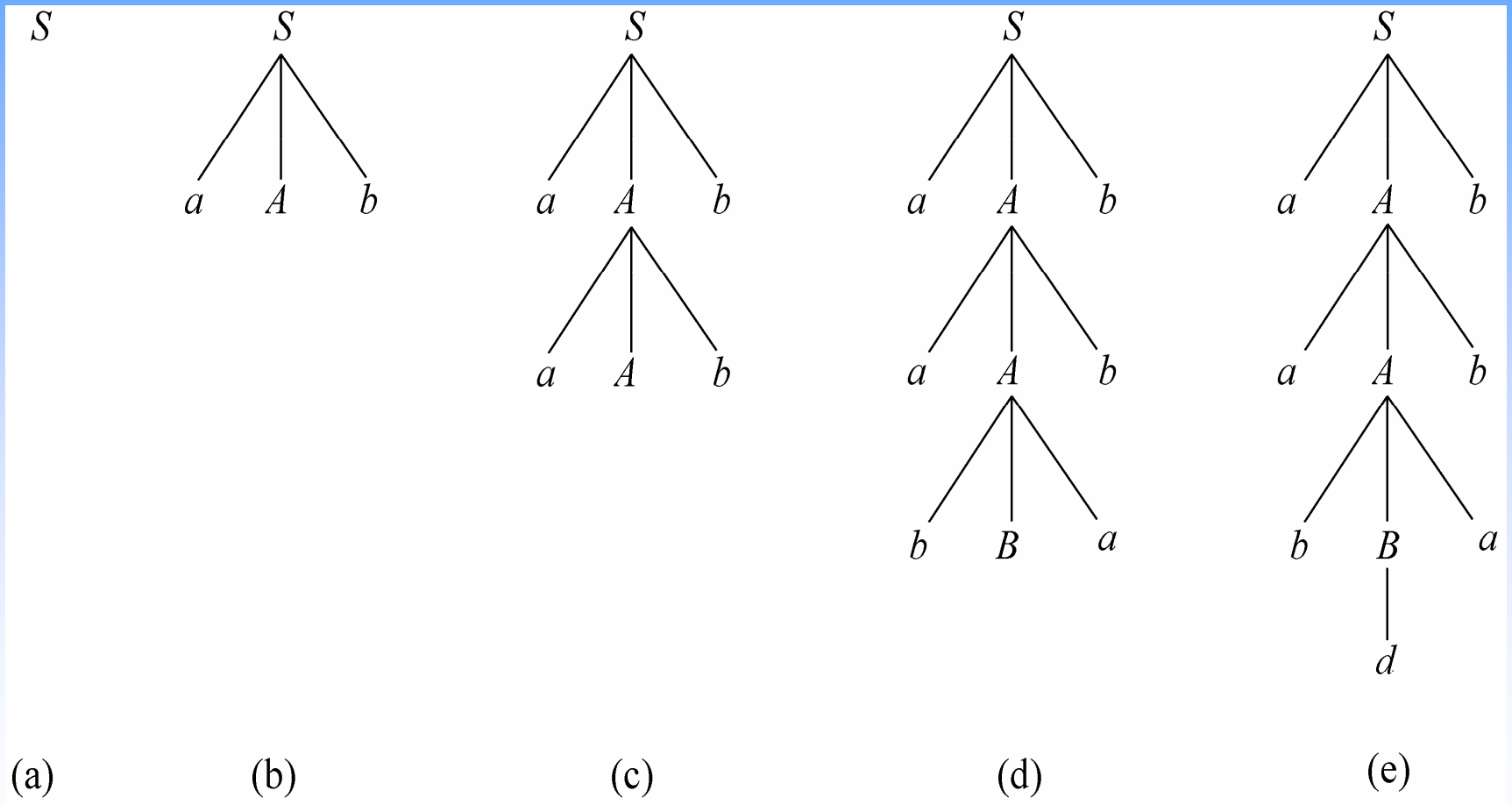
6.1.2 二义性

- **固有二义性的(inherent ambiguity)**
- 如果语言L不存在非二义性文法，则称L是**固有二义性的**，又称L是**先天二义性的**。
- 文法可以是二义性的。
- 语言可以是固有二义性的。

6.1.3 自顶向下的分析和自底向上的分析

- 自顶向下的分析方法
  - 通过考察是否可以从给定文法的开始符号派生出一个符号串，可以判定一个符号串是否为该文法的句子。
- 例
  - **S→aAb|bBa**
  - **A→aAb|bBa**
  - **B→d**

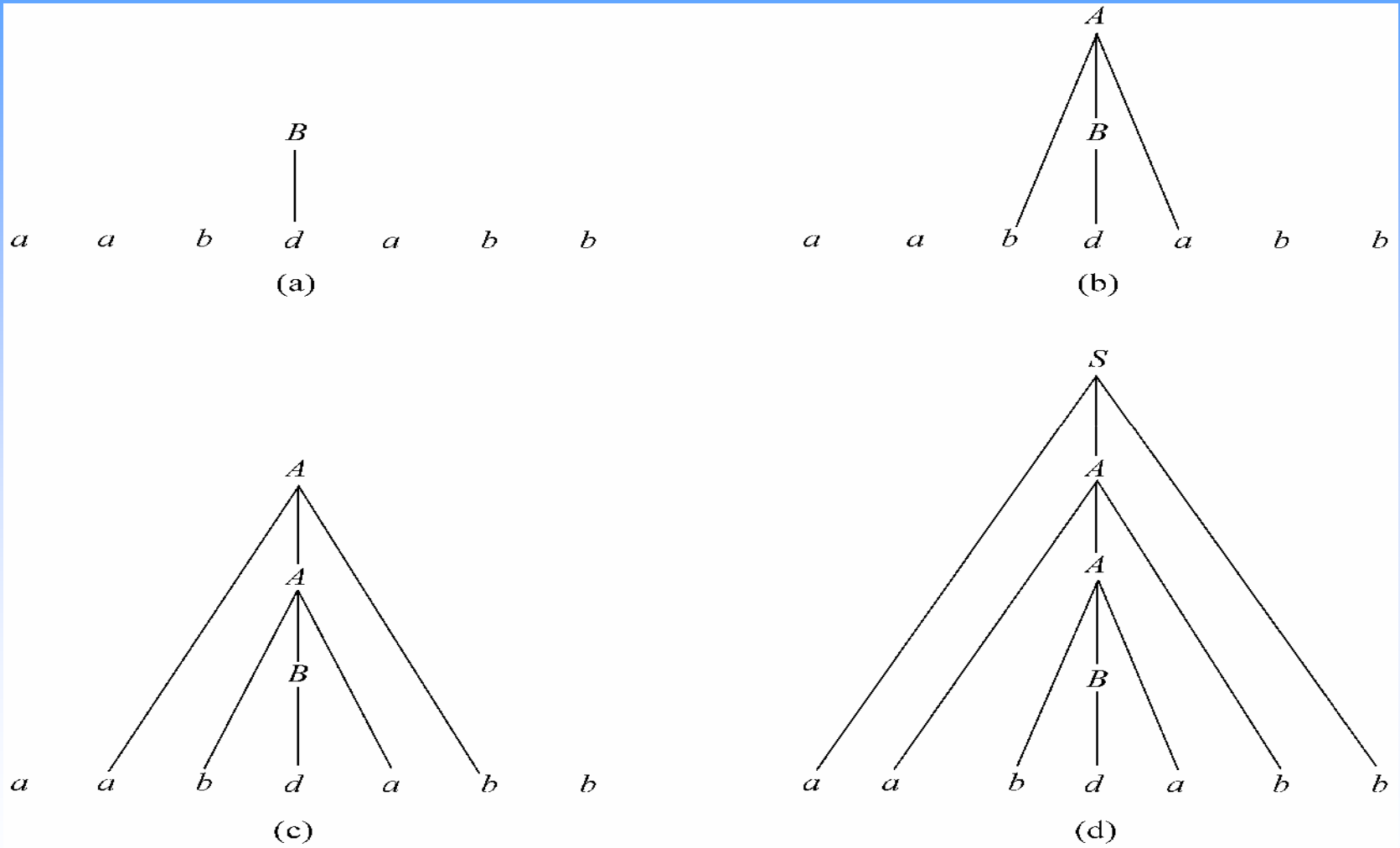
aabdabb的派生树的自顶向下的“生长”过程



6.1.3 自顶向下的分析和自底向上的分析

- **自底向上**的分析方法
  - 通过考察是否可以将一个符号串归约为给定文法的开始符号，完成判定一个符号串是否为该文法的句子的任务。
- 和归约与派生是互逆过程相对应，自顶向下的分析与自底向上的分析互逆的分析过程。

aabdabb的派生树的自底向上的“生长”过程



6.2 上下文无关文法的化简

- 如下文法含有无用的“东西”

G_1：S→0|0A|E

A→ ε |0A|1A|B

B→_C

C→0|1|0C|1C

D→1|1D|2D

E→0E2|E02

- 去掉无用“东西”后的文法

G_2：S→0|0A

A→ ε |0A|1A|B

B→_C

C→0|1|0C|1C

6.2 上下文无关文法的化简

- 去掉产生式**A→ ε** 后的文法

G_3：S→0|0A

A→0|1|0A|1A|B

B→_C

C→0|1|0C|1C

- 去掉产生式**A→B**后的文法

G_4：S→0|0A

A→0|1|0A|1A| _C

C→0|1|0C|1C

- 可以去掉文法中的无用符号、 ε 产生式和单一产生式。

6.2.1 去无用符号

- **无用符号(useless symbol)**
  - 对于任意$X \in V \cup T$，如果存在$w \in L(G)$，X出现在w的派生过程中，即存在α，$\beta \in (V \cup T)^*$，使得$S \Rightarrow^* \alpha X \beta \Rightarrow^* w$，则称$X$是有用的，否则，称$X$是**无用符号**。
- 对**CFG G=(V，T，P，S)**

  ⑴ **G**中的符号X既可能是有用的，也可能是无用的。当X是无用的时候，它既可能是终极符号，也可能是语法变量。

6.2.1 去无用符号

(2) 对于任意$X \in V \cup T$，如果X是有用的，它必须同时满足如下两个条件：

- ① 存在$w \in T^*$，使得$X \Rightarrow^* w$；
- ② α，$\beta \in (V \cup T)^*$，使得$S \Rightarrow^* \alpha X \beta$。

(3) 注意到文法是语言的有穷描述，所以，集合V，T，P都是有穷的。从而我们有可能构造出有效的算法，来完成消除文法的无用符号的工作。

6.2.1 去无用符号

算法 6-1 删除派生不出终极符号行的变量。

- 输入：**CFG G=(V，T，P，S)**。
- 输出：**CFG G′=(V′，T，P′，S)，V′**中不含派生不出终极符号行的变量，并且**L(G′)=L(G)**。
- 主要步骤：

6.2.1 去无用符号

(1) OLDV= Φ;

(2) NEWV={A|A→w∈P且w∈T*};

(3) while OLDV≠NEWV do

begin

(4) OLDV=NEWV;

(5) NEWV=OLDV∪{A|A→α∈P且α∈(T∪OLDV)*}

end

(6) V′=NEWV;

(7) P′={A→α| A→α∈P且A∈V′且α∈(T∪V′)*}

6.2.1 去无用符号

- 第(3)条语句控制对NEWV进行迭代更新。第一次循环将那些恰经过两步可以派生出终极符号行的变量放入NEWV；第二次循环将那些恰经过三步和某些至少经过三步可以派生出终极符号行的变量放入NEWV；……，第n次循环将那些恰经过n步和某些至少经过n+1步可以派生出终极符号行的变量放入NEWV。这个循环一直进行下去，直到所给文法G中的所有可以派生出终极符号行的变量都被放入NEWV中。

6.2.1去无用符号

定理 6-4 算法6-1是正确的。

证明要点：首先证明对于任意$A \in V$，A被放入V'中的充要条件是存在$w \in T$，$A \Rightarrow^{n} w$。再证所构造出的文法是等价的。

(1)对A被放入NEWV的循环次数n施归纳，证明必存在$w \in T$，满足$A \Rightarrow^{+} w$。

6.2.1去无用符号

(2)施归纳于派生步数n，证明如果$A \Rightarrow^n w$，则A被算法放入到NEWV中。实际上，对原教材所给的证明进行分析，同时考虑算法6-1的实际运行，可以证明，A是在第n次循环前被放入到NEWV中的。

(3) 证明 $L(G') = L(G)$ 。显然有 $L(G') \subseteq L(G)$，所以只需证明$L(G)$。

6.2.1 去无用符号

算法 6-2 删除不出现在任何句型中的语法符号。

- 输入：CFG G=(V，T，P，S)。
- 输出：CFG G′=(V′，T′，P′，S)，V′∪T′中的符号必在G的某个句型中出现，并且有L(G′)=L(G)。
- 主要步骤：

6.2.1 去无用符号

- 主要步骤：

(1) OLDV= Φ;

(2) OLDT= Φ;

(3) NEWV={S}∪{A|S→αAβ∈P};

(4) NEWT={a|S→αaβ∈P};

6.2.1去无用符号

(5) while OLDV≠NEWV 或者OLDT≠NEWT do

begin

(6) OLDV=NEWV;

(7) OLDT=NEWT;

(8) NEWV=OLDV∪{B| A∈OLDV且 $A\rightarrow \alpha B \beta \in P$ 且B∈V};

(9) NEWT=OLDT∪{a| A∈OLDV且 $A\rightarrow \alpha a \beta \in P$ 且a∈T};

end

6.2.1去无用符号

(10) $V' = NEWV;$

(11) $T' = NEWT;$

(12) $P' = \{ A \to \alpha \mid A \to \alpha \in P$且 $A \in V'$ 且 $\alpha \in (T' \cup V')^* \}$。

6.2.1去无用符号

定理 6-5 算法**6-2**是正确的。

证明要点：

(1) 施归纳于派生步数**n**，证明如果$S\Rightarrow^n \alpha X\beta$，则当$X\in V$时，**X**在算法中被语句**(3)**或者语句**(8)**放入**NEWV**；当$X\in T$时，它在算法中被语句**(4)**或者语句**(9)**放入**NEWT**。

(2) 对循环次数**n**施归纳，证明如果**X**被放入**NEWT**或者**NEWV**中，则必定存在α，$\beta\in(NEWV\cup NEWT)^*$，使得$S\Rightarrow^n \alpha X\beta$。

(3) 证明$L(G')=L(G)$。

6.2.1去无用符号

定理6-6 对于任意CFL L，$L \neq \Phi$，则存在不含无用符号的CFG G，使得L(G)=L。

- 证明要点：
- 依次用算法6-1和算法6-2对文法进行处理，可以得到等价的不含无用符号的文法。
- 不可先用算法6-2后用算法6-1。

6.2.1去无用符号

- **例 6-2-1** 设有如下文法

  S→AB|a|BB，A→a，C→b|ABa

- 先用算法**6-2**，文法被化简成：

  S→AB|a|BB，A→a

- 再用算法**6-1**，可得到文法：

  S→ a，A→a

- 显然，该文法中的变量**A**是新的无用变量。

6.2.2 去ε-产生式

- ε-产生式（ε-production）
  - 形如$A \rightarrow \varepsilon$的产生式叫做ε-产生式。
  - ε-产生式又称为空产生式（null production。
- 可空(nullable)变量
  - 对于文法G=(V，T，P，S)中的任意变量A，如果$A \Rightarrow^{+} \varepsilon$，则称A为可空变量。

6.2.2 去ε-产生式

- 对形如$A\rightarrow X_1X_2\ldots X_m$的产生式进行考察，找出文法的可空变量集U，然后对于$\forall H\subseteq U$，从产生式$A\rightarrow X_1X_2\ldots X_m$中删除H中的变量。对于不同的H，得到不同的A产生式，用这组A产生式替代产生式$A\rightarrow X_1X_2\ldots X_m$。
- 必须避免在这个过程中产生新的ε-产生式：当$\{X_1, X_2, \ldots, X_m\}\subseteq U$时，不可将$X_1, X_2, \ldots, X_m$同时从产生式$A\rightarrow X_1X_2\ldots X_m$中删除。

6.2.2 去ε-产生式

算法6-3 求CFG G的可空变量集U。

- 输入：CFG G=(V，T，P，S)。
- 输出：G的可空变量集U。
- 主要步骤：

(1) OLDU= Φ;

(2) NEWU={A| A→ ε ∈P};

6.2.2 去 ε -产生式

(3) **while NEWU≠OLDU do**

begin

(4) **OLDU = NEWU;**

(5) **NEWU= OLDU ∪ {A|A→α ∈P并且 α ∈OLDU*}**

end

(6) **U=NEWU**

6.2.2 去ε-产生式

定理 6-7 对于任意**CFG G**，存在不含ε-产生式的**CFG G′**使得**L(G′)=L(G)-{ε}**。

证明：

(1) 构造

- 设**CFG G=(V，T，P，S)**，
- 用算法**6-3**求出**G**的可空变量集**U**，
- 构造**P′**。

6.2.2 去 ε -产生式

- 对于 $\forall A \to X_1 X_2 \ldots X_m \in P$
- 将 $A \to \alpha_1 \alpha_2 \ldots \alpha_m$ 放入 P' ，其中
- if $X_i \in U$ then $\alpha_i = X_i$ 或者 $\alpha_i = \varepsilon$;
- if $X_i \notin U$ then $\alpha_i = X_i$
- 要求：在同一产生式中， α_1， α_2， …， α_m 不能同时为 ε 。

6.2.2 去ε-产生式

- 证明对于任意$w \in T^+$，$A \Rightarrow^n_G w$的充分必要条件是$A \Rightarrow_{G'} w$。
- 必要性：设$A \Rightarrow^n_G w$，施归纳于n，证明$A \Rightarrow^m_{G'} w$成立。
- 当$n=1$时，由$A \Rightarrow_G w$知，$A \to w \in P$，按照定理所给的构造G'的方法，必定有$A \to w \in P'$。所以，$A \Rightarrow_{G'} w$成立。

6.2.2 去 ε -产生式

- 设n≤k时结论成立(k≥1)，当n=k+1时

$A \Rightarrow X_1X_2\cdots X_m$

$\Rightarrow^*_G w_1X_2\cdots X_m$

$\Rightarrow^*_G w_1w_2\cdots X_m$

…

$\Rightarrow^*_G w_1w_2\cdots w_m$

其中$w_1w_2\cdots w_m=w$，且w_1，w_2，……，$w_m \in T^*$。

6.2.2 去 ε -产生式

注意到$w \neq \varepsilon$，必存在$1 \leqslant i \leqslant m$，$w_i \neq \varepsilon$，设$i$，$j$，…，$k$是$w_1$，$w_2$，…，$w_m$中所有非空串的下标，并且$1 \leqslant i \leqslant j \leqslant \ldots \leqslant k \leqslant m$，即：

$$w = w_i w_j \ldots w_k$$

按照G'的构造方法，$A \to X_i X_j \ldots X_k \in P'$

再由归纳假设，

$X_i \Rightarrow^*_{G'} w_i$，$X_j \Rightarrow^*_{G'} w_j$，…，$X_k \Rightarrow^*_{G'} w_k$。

6.2.2 去 ε -产生式

$A \Rightarrow^*_{G'} X_iX_j\cdots X_k$

$\Rightarrow^*_{G'} w_iX_j\cdots X_k$

$\Rightarrow^*_{G'} w_iw_j\cdots X_k$

…

$\Rightarrow^*_{G'} w_iw_j\cdots w_k$

所以，结论对**n=k+1**成立。由归纳法原理，结论对所有的**n**成立。

6.2.2 去ε-产生式

充分性：设$A \Rightarrow^{m}_{G'} w$，施归纳于m，证明$A \Rightarrow^{n}_{G} w$成立。

当m=1时，由$A \Rightarrow_{G'} w$知，$A \to w \in P'$，按照定理所给的构造G'的方法，必定有$A \to \alpha \in P$。$A \to w$是通过删除产生式$A \to \alpha$右部中的可空变量而构造出来的，所以，

$A \Rightarrow_{G} \alpha \Rightarrow^{*}_{G} w$ 成立。

6.2.2 去 ε -产生式

设$n \leqslant k$时结论成立$(k \geqslant 1)$，当$m=k+1$时

$A \Rightarrow^*_{G'} X_iX_j\cdots X_k$

$\Rightarrow^*_{G'} w_iX_j\cdots X_k$

$\Rightarrow^*_{G'} w_iw_j\cdots X_k$

…

$\Rightarrow^*_{G'} w_iw_j\cdots w_k=w$

其中$X_i \Rightarrow^*_{G'} w_i$，$X_j \Rightarrow^*_{G'} w_j$，…，$X_k \Rightarrow^*_{G'} w_k$。

6.2.2 去 ε -产生式

表明$A\to X_iX_j\ldots X_k\in P'$ 。按照G' 的构造方法，必定存在$A\to X_1X_2\ldots X_m\in P'$ ，而且

$\{X_i,\ X_j,\ \ldots,\ X_k\}\subseteq\{X_1,\ X_2,\ \ldots,\ X_m\}$

$\{X_1,\ X_2,\ \ldots,\ X_m\}-\{X_i,\ X_j,\ \ldots,\ X_k\}\subseteq U$

从而，

$$A\Rightarrow_G X_1X_2\ldots X_m$$
$$\Rightarrow^*_G X_iX_j\ldots X_k$$

6.2.2 去 ε -产生式

再根据$X_i \Rightarrow^*_{G'} w_i$，$X_j \Rightarrow^*_{G'} w_j$，…，$X_k \Rightarrow^*_{G'} w_k$和归纳假设，有

$X_i \Rightarrow^*_G w_i$，$X_j \Rightarrow^*_G w_j$，…，$X_k \Rightarrow^*_G w_k$。

这表明，如下派生成立：

$$
\begin{aligned}
A &\Rightarrow_G X_1X_2\cdots X_m \\
&\Rightarrow^*_G X_iX_j\cdots X_k \\
&\Rightarrow^*_G w_iX_j\cdots X_k \\
&\Rightarrow^*_G w_iw_j\cdots X_k \\
&\cdots \\
&\Rightarrow^*_G w_iw_j\cdots w_k=w
\end{aligned}
$$

6.2.2 去ε-产生式

表明结论对**m=k+1**成立。由归纳法原理，结论对任意**m**成立。

注意到**A** 的任意性，当**S=A**时结论成立。即：

$S\Rightarrow^{*}_{G} w$ 的充分必要条件是**$S\Rightarrow^{*}_{G'} w$**

亦即：**$L(G')=L(G)-\{\varepsilon\}$**。

定理得证。

6.2.3 去单一产生式

文法G_{exp1}：E→E+T|E-T|T

T→T*F|T/F|F

F→F↑P|P

P→(E)|N(L)|id

N→sin|cos|exp|abs|log|int

L→L,E|E

存在派生：

E⇒T⇒F⇒P⇒id

6.2.3 去单一产生式

G_{exp3}： E→E+T|E-T|T*F|T/F|F ↑ P|(E)|N(L)|id

T→T*F|T/F| F ↑ P| (E)|N(L)|id

F→F ↑ P| (E)|N(L)|id

P→(E)|N(L)|id

N→sin|cos|exp|abs|log|int

L→L,E|E

- 该文法中不存在类似的派生。

6.2.3 去单一产生式

- **单一产生式(unit production)**
- 形如$A \to B$的产生式称为**单一产生式**。
- **定理 6-8**对于任意CFG G，$\varepsilon \notin L(G)$，存在等价的CFG G_1，G_1不含无用符号、ε-产生式和单一产生式。
- 满足本定理的CFG为化简过的文法。
- 已有去无用符号和去ε-产生式的结论，所以只讨论去单一产生式的问题。

6.2.3 去单一产生式

- 证明要点：

（1）构造G_2，满足$L(G_2)=L(G)$，并且G_2中不含单一产生式。

用非单一产生式$A_1\rightarrow\alpha$取代$A_1\Rightarrow^*_G A_n\Rightarrow\alpha$用到的产生式系列$A_1\rightarrow A_2$，$A_2\rightarrow A_3$，…，$A_{n-1}\rightarrow A_n$，$A_n\rightarrow\alpha$。其中，$A_1\rightarrow A_2$，$A_2\rightarrow A_3$，…，$A_{n-1}\rightarrow A_n$都是单一产生式。（$n\geqslant 1$）

6.2.3 去单一产生式

（2） 证明$L(G_2)=L(G)$。

用$A_1\to\alpha$ 所完成的派生$A_1\Rightarrow\alpha$ 与产生式系列 $A_1\to A_2$，$A_2\to A_3$，……，$A_{n-1}\to A_n$，$A_n\to\alpha$ 所完成的派生$A_1\Rightarrow^*_G A_n\Rightarrow\alpha$ 相对应。

在原文法中可能会出现一个变量在派生过程中循环出现的情况，在$w\in L(G)$ 证明$w\in L(G_2)$ 的过程中，要取w在G中的一个最短的最左派生。

$$S=\alpha_0\Rightarrow_G\alpha_1\Rightarrow_G\alpha_2\Rightarrow_G\cdots\Rightarrow_G\alpha_n=w$$

6.2.3 去单一产生式

（3）删除G_2中的无用符号。

由于在删除单一产生式后，文法中可能出现新的无用符号，因此，我们还需要再次删除新出现的无用符号。

此外，在去ε-产生式后可能会产生新的单一产生式，也可能会引进新的无用符号。这是值得注意的。

6.3 乔姆斯基范式

- **乔姆斯基范式文法(Chomsky normal form，CNF)简称为Chomsky文法，或Chomsky范式。**

CFG G=(V，T，P，S)中的产生式形式：

$A \rightarrow BC$，$A \rightarrow a$

其中，**A、B、$C \in V$，$a \in T$。**

- **CNF**中，不允许有ε-产生式、单一产生式。

6.3 乔姆斯基范式

- **例 6-3-1** 试将文法G_{exp4}转换成等价的 **GNF**。

G_{exp4}： $E \to E+T \mid T*F \mid F \uparrow P \mid (E) \mid id$

$T \to T*F \mid F \uparrow P \mid (E) \mid id$

$F \to F \uparrow P \mid (E) \mid id$

$P \to (E) \mid id$

- 可以分两步走
  - 变成$A \to a$和$A \to A_1A_2 \ldots A_n$的形式。
  - 变成**CNF**。

第一步

$E \rightarrow EA_{+}T \mid T\ A_{*}F \mid F\ A_{\uparrow}P \mid A_{(}EA_{5} \mid id$

$T \rightarrow TA_{*}F \mid FA_{\uparrow}P \mid A_{(}EA_{)} \mid id$

$F \rightarrow FA_{\uparrow}P \mid A_{(}EA_{)} \mid id$

$P \rightarrow A_{(}EA_{)} \mid id$

$A_{+} \rightarrow +$

$A_{*} \rightarrow *$

$A_{\uparrow} \rightarrow \uparrow$

$A_{(} \rightarrow ($

$A_{)} \rightarrow)$

第二步

G_{expCNF} : $E \rightarrow EA_1 | TA_2$

$E \rightarrow FA_3 | A_(A_4 | id$

$T \rightarrow TA_2 | FA_3 | A_(A_4 | id$

$F \rightarrow FA_3 | A_(A_4 | id$

$P \rightarrow A_(A_4 | id$

$A_+ \rightarrow +$

$A_* \rightarrow *$

$A_\uparrow \rightarrow \uparrow$

$A_(\rightarrow ($

$A_) \rightarrow)$

$A_1 \rightarrow A_+ T$

$A_2 \rightarrow A_* F$

$A_3 \rightarrow A_\uparrow P$

$A_4 \rightarrow EA_)$

6.3 乔姆斯基范式

定理6-9 对于任意CFG G，$\varepsilon \notin L(G)$，存在等价的 CNF G_2。

证明要点：

1. 构造**CNF**

按照上述例子所描述的转换方法，在构造给定**CFG**的**CNF**时，可以分两步走。

- 假设**G**为化简过的文法
- 构造$G_1=(V_1, T, P_1, S)$ G_1中的产生式都是形如$A\to B_1B_2\ldots B_m$和$A\to a$的产生式，其中，A，B_1，B_2，…，$B_m\in V_1$，$a\in T$，$m\geqslant 2$

6.3 乔姆斯基范式

- 构造CNF $G_2=(V_2，T，P_2，S)$。
- $m \geq 3$时，引入新变量：B_1、B_2、…、B_{m-2}，将G_1的形如$A \rightarrow A_1A_2 \ldots A_m$的产生式替换成

$A \rightarrow A_1B_1$

$B_1 \rightarrow A_2B_2$

…

$B_{m-2} \rightarrow A_{m-1}A_m$

6.3 乔姆斯基范式

2. 构造的正确性证明。

- 按照上述构造，证明被替换的产生式是等价的。
- 例如：在第二步中{$A\rightarrow A_1B_1$，$B_1\rightarrow A_2B_2$，…，$B_{m-2}\rightarrow A_{m-1}A_m$}与$A\rightarrow A_1A_2\ldots A_m$等价。

6.3 乔姆斯基范式

- **例 6-6** 试将下列文法转换成等价的 **CNF**。

  S→bA | aB

  A→bAA | aS | a

  B→aBB | bS | b

6.3 乔姆斯基范式

- 先引入变量 B_a，B_b 和产生式 $B_a \to a$，$B_b \to b$，完成第一步变换。

$S \to B_bA \mid B_aB$

$A \to B_bAA \mid B_aS \mid a$

$B \to B_aBB \mid B_bS \mid b$

$B_a \to a$

$B_b \to b$

6.3 乔姆斯基范式

- 引入新变量B_1、B_2

  $S \rightarrow B_bA \mid B_aB$

  $A \rightarrow B_bB_1 \mid B_aS \mid a$

  $B \rightarrow B_aB_2 \mid B_bS \mid b$

  $B_a \rightarrow a$

  $B_b \rightarrow b$

  $B_1 \rightarrow AA$

  $B_2 \rightarrow BB$

6.3 乔姆斯基范式

- 不能因为原来有产生式$A\rightarrow a$和$B\rightarrow b$而放弃引进变量B_a、B_b和产生式$B_a\rightarrow a$、$B_b\rightarrow b$。
- $L(A)=\{x \mid x\in\{a，b\}^+ \ \&\ x$中a的个数比b的个数恰多1个$\}$。
- $L(B)=\{x \mid x\in\{a，b\}^+ \ \&\ x$中b的个数比a的个数恰多1个$\}$。
- $L(B_a)=\{ a \}$。
- $L(B_b)=\{ b \}$。

6.4 格雷巴赫范式

- **格雷巴赫范式文法(Greibach normal form ,GNF)简称为Greibach文法，或Greibach范式。**

  $A \rightarrow a\alpha$

其中，$A \in V$，$a \in T$，$\alpha \in V^*$。

- 在GNF中，有如下两种形式的产生式
  - $A \rightarrow a$
  - $A \rightarrow aA_1A_2 \ldots A_m \qquad (m \geqslant 1)$

6.4 格雷巴赫范式

- 右线性文法是一种特殊的**GNF**。
- 由于**GNF**中不存 ε -产生式，所以对任意的 **GNF G，** ε ∉**L(G)**。
- 当 ε ∉**L(G′)**时，能够找到一个**GNF G**，使得 **L(G)=L(G′)** 。
- 经过化简的**CFG**，都有一个等价的**GNF**。

6.4 格雷巴赫范式

引理 6-1 对于任意的**CFG G=(V，T，P，S)**，

$A\to \alpha B \beta \in P$，且G中所有的B产生式为

$$B\to \gamma_1 | \gamma_2 | \ldots | \gamma_n$$

取$G_1=(V，T，P_1，S)$

$P_1=(P-\{A\to \alpha B \beta\})\cup\{A\to \alpha \gamma_1 \beta$ ，

$A\to \alpha \gamma_2 \beta$ ，…，$A\to \alpha \gamma_n \beta\}$

则，$L(G_1)=L(G)$。

6.4 格雷巴赫范式

- 证明

  以下两组产生式等价

  – $A \rightarrow \alpha B \beta$ ； $B \rightarrow \gamma_1 | \gamma_2 | \ldots | \gamma_n$

  – $\{ A \rightarrow \alpha \gamma_1 \beta ,\ A \rightarrow \alpha \gamma_2 \beta ,\ \ldots ,\ A \rightarrow \alpha \gamma_n \beta \}$

6.4 格雷巴赫范式

- **递归(recursive)**
- 如果G中存在形如$A \Rightarrow^n \alpha A \beta$ 的派生，则称该派生是关于变量**A递归的**，简称为递归派生。
- 当**n=1**时，称该派生关于变量**A直接递归(directly recursive)**，简称为直接递归派生。
- 形如$A \rightarrow \alpha A \beta$ 的产生式是变量A的直接递归的**(directly recursive)**产生式。

6.4 格雷巴赫范式

- 当**n≥2**时，称该派生是关于变量**A**的**间接递归(indirectly recursive)**派生。简称为间接递归派生。
- 当 α = ε 时，称相应的(直接/间接)递归为**(直接/间接)左递归(left-recursive)**；
- 当 β = ε 时，称相应的(直接/间接)递归为**(直接/间接)右递归(right-recursive)**。

6.4 格雷巴赫范式

- **引理 6-2** 对于任意的**CFG G=(V，T，P，S)，G**中所有**A**的产生式

$$\begin{cases} A \to A\alpha_1 | A\alpha_2 | \ldots | A\alpha_n \\ A \to \beta_1 | \beta_2 | \ldots | \beta_m \end{cases}$$

可以被等价地替换为产生式组

$$\begin{cases} A \to \beta_1 | \beta_2 | \ldots | \beta_m \\ A \to \beta_1 B | \beta_2 B | \ldots | \beta_m B \\ B \to \alpha_1 | \alpha_2 | \ldots | \alpha_n \\ B \to \alpha_1 B | \alpha_2 B | \ldots | \alpha_n B \end{cases}$$

6.4 格雷巴赫范式

- 证明要点：

  用直接右递归取代原来的直接左递归。

  两组产生式产生的符号串都是

  $$\beta_k \alpha_{h_q} \alpha_{h_{q-1}} \ldots\ldots \alpha_{h_2} \alpha_{h_1}$$

  前者是先产生

  $$\alpha_{h_q} \alpha_{h_{q-1}} \ldots\ldots \alpha_{h_2} \alpha_{h_1}$$

  然后产生 β_k。

6.4 格雷巴赫范式

- 后者先产生 β_k。
- 然后产生

$$\alpha_{h_q}\alpha_{h_{q-1}}.....\alpha_{h_2}\alpha_{h_1}$$

6.4 格雷巴赫范式

$$A \Rightarrow A\alpha_{h_1}$$
$$\Rightarrow A\alpha_{h_2}\alpha_{h_1}$$
$$\cdots\cdots$$
$$\Rightarrow A\alpha_{h_{q-1}}..\alpha_{h_2}\alpha_{h_1}$$
$$\Rightarrow A\alpha_{h_q}\alpha_{h_{q-1}}..\alpha_{h_2}\alpha_{h_1}$$
$$\Rightarrow \beta_k\alpha_{h_q}\alpha_{h_{q-1}}..\alpha_{h_2}\alpha_{h_1}$$

$$A \Rightarrow \beta_k\alpha_{hq}B$$
$$\Rightarrow \beta_k\alpha_{h_q}\alpha_{h_{q-1}}B$$
$$\cdots\cdots$$
$$\Rightarrow \beta_k\alpha_{h_q}\alpha_{h_{q-1}}..\alpha_{h_2}B$$
$$\Rightarrow \beta_k\alpha_{h_q}\alpha_{h_{q-1}}..\alpha_{h_2}\alpha_{h_1}$$

6.4 格雷巴赫范式

定理6-10 对于任意CFG G，$\varepsilon \notin L(G)$，存在等价的GNF G_1。

- 证明要点：

(1)使用引理6-1将产生式都化成如下形式的产生式

$A \to A_1A_2 \ldots A_m$

$A \to aA_1A_2 \ldots A_{m-1}$

$A \to a$

其中，A，A_1，A_2，…，$A_m \in V_1$，$a \in T$，$m \geqslant 2$。

6.4 格雷巴赫范式

(2)根据引理6-1和6-2，将产生式变成如下形式的产生式

$A_i \to A_j \alpha$　　$i \leqslant j+1$

$A_i \to a \alpha$

$B_i \to \alpha$

其中，$V_2=V_1 \cup \{B_1, B_2, \ldots, B_n\}$，$V_1 \cap \{B_1, B_2, \ldots, B_n\}=\Phi$。

“B类变量”：$\{B_1, B_2, \ldots, B_n\}$是在文法的改造过程中引入的新变量。

V_1中的变量称为“A类变量”。

6.4 格雷巴赫范式

（3）根据引理6-1，从编号较大的变量开始，逐步替换，使所有产生式满足GNF的要求：

1) for k=m-1 to 1 do

2) if $A_k \rightarrow A_{k+1}\beta \in P_2$ then

3) for 所有的A_{k+1}产生式$A_{k+1} \rightarrow \gamma$ do 将产生式$A_k \rightarrow \gamma\beta$放入$P_3$;

4) for k= 1 to n do

5) 根据引理6-1，用P_3中的产生式将所有的B_k产生式替换成满足GNF要求的形式。

6.5 自嵌套文法

- **自嵌套文法(self-embedding grammar)**

CFG G=(V，T，P，S)是化简后的文法，如果**G**中存在有形如$A \Rightarrow^{+} \alpha A \beta$的派生，则称**G**为**自嵌套文法**，其中$\alpha$、$\beta \in (V \cup T)^{+}$。

- 自嵌套的文法描述的语言可以是正则语言
- 例如：

$$S \rightarrow 0S0|1S1|0S1|1S0|0S|1S|0|1$$

6.5 自嵌套文法

定理 6-11 非自嵌套的文法产生的语言是正则语言。

- 证明要点：

（1）将G化成GNF

（2）取RG $G'=(V',T,P',[S])$，其中

$V'=\{[\alpha] \mid \alpha \in V^{+}$并且$|\alpha| \leqslant m(n-2)+1\}$

$P'=\{[A\alpha] \rightarrow a[\beta\ \alpha] \mid A \rightarrow a\beta \in P$并且$\beta \in V^{*}\}$

当 $\alpha=\varepsilon$ 时，$[\alpha]=\varepsilon$ 。

6.6 小结

本章讨论了CFG的派生树，A子树，最左派生与最右派生，派生与派生树的关系，二义性文法与固有二义性语言，句子的自顶向下分析和自底向上分析；无用符号的消去算法，空产生式的消除，单一产生式的消除。CFG的CNF和GNF；CFG的自嵌套特性。

(1)$S\Rightarrow^{*}\alpha$ 的充分必要条件为G有一棵结果为 α 的派生树。

(2)如果 α 是CFG G的一个句型，则G中存在 α 的最左派生和最右派生。

6.6 小结

(3)文法可能是二义性的，但语言只可能是固有二义性的，且这种语言是存在的。

(4)对于任意CFG G，$\varepsilon \notin L(G)$，存在等价的CFG G_1，G_1不含无用符号、ε-产生式和单一产生式。

(5)对于任意CFG G，$\varepsilon \notin L(G)$，存在等价的CNF G_2。

(6)对于任意CFG G，$\varepsilon \notin L(G)$，存在等价的GNF G3。

(7)非自嵌套的文法产生的语言是正则语言。

第7章下推自动机

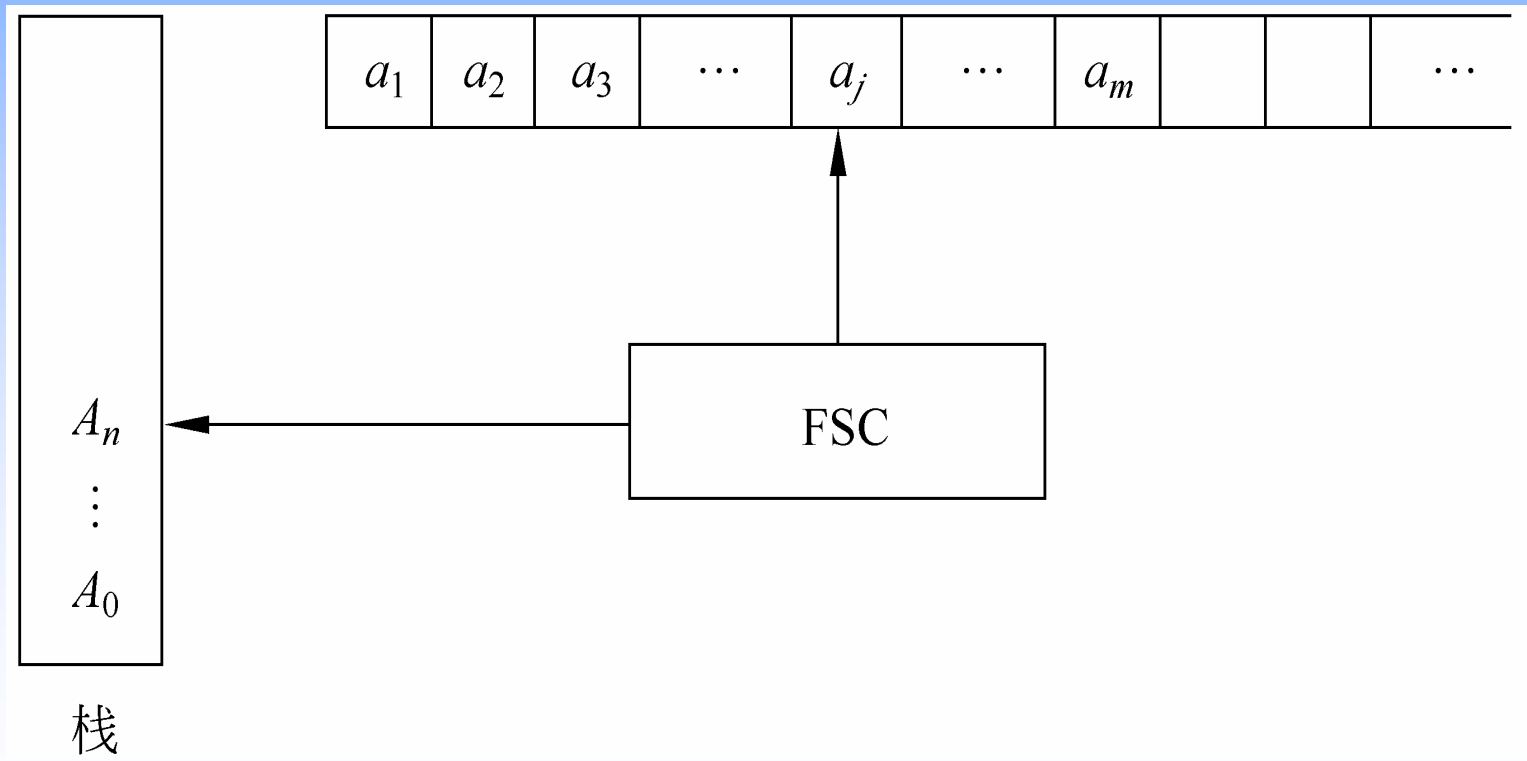
- **PDA描述CFL，所以它应该与CFG等价。**
- **PDA应该包含FA的各个元素，或者包含那些可以取代FA的各个元素的功能的元素。**
- **PDA按照最左派生的派生顺序，处理处于当前句型最左边的变量，因此，需要采用栈作为其存储机构。**
- **按照FA的“习惯”，PDA用终态接受语言。**
- **模拟GNF的派生PDA用空栈接受语言。**

第7章下推自动机

- 主要内容
  - **PDA**的基本概念。
  - **PDA**的构造举例。
  - 用终态接受语言和用空栈接受语言的等价性。
  - **PDA**是**CFL**的接受器。
- 重点
  - **PDA**的基本定义及其构造，**PDA**是CFL的等价描述。
- 难点
  - 根据**PDA**构造CFG。

7.1 基本定义

- PDA的物理模型



7.1 基本定义

- **PDA**应该含有三个基本结构
  - 存放输入符号串的输入带。
  - 存放文法符号的栈。
  - 有穷状态控制器。
- **PDA**的动作
  - 在有穷状态控制器的控制下根据它的当前状态、栈顶符号、以及输入符号作出相应的动作，在有的时候，不需要考虑输入符号。

7.1 基本定义

- **下推自动机(pushdown automaton，PDA)**

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

Q——状态的非空有穷集合。$\forall q \in Q$，**q**称为**M**的一个**状态(state)**;

Σ——输入字母表(input alphabet)。要求**M**的输入字符串都是Σ上的字符串;

Γ——栈符号表(stack alphabet)。$\forall A \in \Gamma$，叫做一个栈符号;

7.1 基本定义

- **Z_0——$Z_0 \in \Gamma$ 叫做开始符号(start symbol)**，是M启动时候栈内惟一的一个符号。所以，习惯地称其为栈底符号；
- **q_0——$q_0 \in Q$，是M的开始状态(initial state)**，也可叫做**初始状态**或者**启动状态**；
- **F——$F \subseteq Q$**，是M的**终止状态(final state)**集合，简称为终态集。$\forall q \in F$，q称为M的终止状态，也可称为**接受状态(accept state)**，简称为终态。

7.1 基本定义

- δ ——**状态转移函数**(transition function)，有时候又叫做**状态转换函数**或者**移动函数**。

$$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$$

7.1 基本定义

$\delta(q, a, Z)=\{(p_1, \gamma_1), (p_2, \gamma_2), \ldots, (p_m, \gamma_m)\}$

- 表示M在状态q，栈顶符号为Z时，读入字符a，对于i=1，2，…，m，可以选择地将状态变成p_i，并将栈顶符号Z弹出，将γ_i中的符号从右到左依次压入栈，然后将读头向右移动一个带方格而指向输入字符串的下一个字符。

7.1 基本定义

$$\delta(q, \varepsilon, Z)=\{(p_1, \gamma_1), (p_2, \gamma_2), \ldots, (p_m, \gamma_m)\}$$

- 表示M进行一次 ε -移动(空移动)，即M在状态q，栈顶符号为Z时，无论输入符号是什么，对于i=1，2，…，m，可以选择地将状态变成p_i，并将栈顶符号Z弹出，将γ_i中的符号从右到左依次压入栈，读头不移动。

7.1 基本定义

- 符号使用约定
- 英文字母表较为前面的小写字母，如**a**，**b**，**c**，…，表示输入符号；
- 英文字母表较为后面的小写字母，如**x**，**y**，**z**，…，表示由输入字符串；
- 英文字母表的大写字母，表示栈符号；
- 希腊字母α，β，γ，…，表示栈符号串。

7.1 基本定义

- **即时描述(instantaneous description，ID)**

  (q，w，γ)∈(Q，Σ*，Γ*)称为M的一个即时描述。它表示M处于状态q，w是当前还未处理的输入字符串，而且M正注视着w的首字符，栈中的符号串为γ，γ的最左符号为栈顶符号，最右符号为栈底的符号，较左的符号在栈的较上面，较右的符号在栈的较下面。

7.1 基本定义

如果$(p，\ \gamma)\in\delta(q，a，Z)$，$a\in\Sigma$，则

$(q，aw，Z\beta)\vdash_M(p，w，\gamma\beta)$

表示M做一次非空移动，从ID(q，aw，Zβ)变成ID(p，w，γβ)。

如果$(p，\ \gamma)\in\delta(q，\varepsilon，Z)$，则

$(q，w，Z\beta)\vdash_M(p，w，\gamma\beta)$

表示M做一次空移动，从ID(q，aw，Zβ)变成ID(p，w，γβ)。

7.1 基本定义

- $\vdash_M^n$是 $\vdash_M$ 的n次幂
  - $(q_1, w_1, \beta_1) \vdash_M^n (q_n, w_n, \beta_n)$
- $\vdash_M^*$是 $\vdash_M$ 的克林闭包
  - $(q, w, \alpha) \vdash_M^* (p, x, \beta)$
- $\vdash_M^+$是 $\vdash_M$ 的正闭包
  - $(q, w, \alpha) \vdash_M^+ (p, x, \beta)$

7.1 基本定义

- M接受的语言
  - M用终态接受的语言
    - $L(M)=\{w \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \beta)$ 且 $p \in F\}$
  - M用空栈接受的语言
    - $N(M) =\{w \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon)\}$

7.1 基本定义

- **例 7-1** 考虑接受语言$L=\{w2w^T \mid w\in\{0,1\}^*\}$的**PDA**的设计。
- 解法**1**：
- 先设计产生**L**的**CFG** G_1：

  G_1：$S\rightarrow 2|0S0|1S1$
- 再将此文法转化成**GNF**：

  G_2：$S\rightarrow 2|0SA|1SB$

  $A\rightarrow 0$

  $B\rightarrow 1$

7.1 基本定义

- 句子**0102010**的最左派生和我们希望相应的**PDA M**的动作。

派生	M应该完成的动作
S⇒0SA	从q_0启动，读入**0**，将**S**弹出栈，将**SA**压入栈，状态不变
⇒01SBA	在状态q_0，读入**1**，将**S**弹出栈，将**SB**压入栈，状态不变
⇒010SABA	在状态**q0**，读入**0**，将**S**弹出栈，将**SA**压入栈，状态不变
⇒0102ABA	在状态q_0，读入**2**，将**S**弹出栈，将 ε 压入栈，状态不变
⇒01020BA	在状态q_0，读入**0**，将**A**弹出栈，将 ε 压入栈，状态不变
⇒010201A	在状态q_0，读入**1**，将**B**弹出栈，将 ε 压入栈，状态不变
⇒0102010	在状态q_0，读入**0**，将**A**弹出栈，将 ε 压入栈，状态不变

7.1 基本定义

$M_1=(\{q_0\}, \{0, 1, 2\}, \{S, A, B\}, \delta_1, q_0, S, \Phi)$。其中：

$\delta_1(q_0, 0, S)=\{(q_0, SA)\}$

$\delta_1(q_0, 1, S)=\{(q_0, SB)\}$

$\delta_1(q_0, 2, S)=\{(q_0, \varepsilon)\}$

$\delta_1(q_0, 0, A)=\{(q_0, \varepsilon)\}$

$\delta_1(q_0, 1, B)=\{(q_0, \varepsilon)\}$

此时有：$N(M_1)=L$。

7.1 基本定义

$M_2=(\{q_0,q_1\},\{0,1,2\},\{S,A,B,Z_0\},\delta_2,q_0,Z_0,\{q_1\})$

$\delta_2(q_0,\ 0,\ Z_0)=\{(q_0,\ SAZ_0)\}$

$\delta_2(q_0,\ 1,\ Z_0)=\{(q_0,\ SBZ_0)\}$

$\delta_2(q_0,\ 2,\ Z_0)=\{(q_1,\ \varepsilon)\}$

$\delta_2(q_0,\ 0,\ S)=\{(q_0,\ SA)\}$

$\delta_2(q_0,\ 1,\ S)=\{(q_0,\ SB)\}$

$\delta_2(q_0,\ 2,\ S)=\{(q_0,\ \varepsilon)\}$

$\delta_2(q_0,\ 0,\ A)=\{(q_0,\ \varepsilon)\}$

$\delta_2(q_0,\ 1,\ B)=\{(q_0,\ \varepsilon)\}$

$\delta_2(q_0,\ \varepsilon,\ Z_0)=\{(q_1,\ \varepsilon)\}$

此时有：$N(M_2)=L(M_2)=L$。

7.1 基本定义

- 解法2：
- 注意到$L=\{w2w^T|w\in\{0,1\}^*\}$，所以PDA M_3的工作可以分成两大阶段。
  - 在读到字符2之前，为“记载”阶段：每读到一个符号就在栈中做一次相应的记载。
  - 在读到2以后，再读到字符时，就应该进入“匹配”阶段：由于栈的“先进后出”特性正好与w^T相对应，所以，用栈顶符号逐一地与输入字符匹配。

7.1 基本定义

- $M_3=(\{q_0, q_1, q_2, q_f, q_t\}, \{0, 1, 2\}, \{A, B, Z_0\}, \delta_3, q_0, Z_0, \{q_f\})$
- q_0为开始状态
- q_1为记录状态
- q_2为匹配状态
- q_f为终止状态
- q_t陷阱状态

7.1 基本定义

$\delta_3(q_0, 0, Z_0)=\{(q_1, AZ_0)\}$

$\delta_3(q_0, 1, Z_0)=\{(q_1, BZ_0)\}$

$\delta_3(q_0, 2, Z_0)=\{(q_f, \varepsilon)\}$

$\delta_3(q_1, 0, A)=\{(q_1, AA)\}$

$\delta_3(q_1, 1, A)=\{(q_1, BA)\}$

$\delta_3(q_1, 0, B)=\{(q_1, AB)\}$

$\delta_3(q_1, 1, B)=\{(q_1, BB)\}$

7.1 基本定义

$\delta_3(q_1, 2, A)=\{(q_2, A)\}$

$\delta_3(q_1, 2, B)=\{(q_2, B)\}$

$\delta_3(q_2, 0, A)=\{(q_2, \varepsilon)\}$

$\delta_3(q_2, 0, B)=\{(q_t, \varepsilon)\}$

$\delta_3(q_2, 1, B)=\{(q_2, \varepsilon)\}$

$\delta_3(q_2, 1, A)=\{(q_t, \varepsilon)\}$

$\delta_3(q_2, \varepsilon, Z_0)=\{(q_f, \varepsilon)\}$

此时有：$N(M_3)=L(M_3)=L$。

7.1 基本定义

- 不追求让PDA同时用终止状态和空栈接受同样的语言，还可以删除状态q_f。这样我们可以得到PDA M_4 。
- $M_4=(\{q_0, q_1, q_2\}, \{0, 1, 2\}, \{A, B, Z_0\}, \delta_4, q_0, Z_0, \Phi)$

$\delta_4(q_0, 0, Z_0)=\{(q_1, AZ_0)\}$

$\delta_4(q_0, 1, Z_0)=\{(q_1, BZ_0)\}$

$\delta_4(q_0, 2, Z_0)=\{(q_2, \varepsilon)\}$

7.1 基本定义

$\delta_4(q_1, 0, A)=\{(q_1, AA)\}$

$\delta_4(q_1, 1, A)=\{(q_1, BA)\}$

$\delta_4(q_1, 0, B)=\{(q_1, AB)\}$

$\delta_4(q_1, 1, B)=\{(q_1, BB)\}$

$\delta_4(q_1, 2, A)=\{(q_2, A)\}$

$\delta_4(q_1, 2, B)=\{(q_2, B)\}$

$\delta_4(q_2, 0, A)=\{(q_2, \varepsilon)\}$

$\delta_4(q_2, 1, B)=\{(q_2, \varepsilon)\}$

7.1 基本定义

- **确定的(deterministic)PDA**

$\forall(q, a, Z) \in Q \times \Sigma \times \Gamma$，

$|\delta(q, a, Z)| + |\delta(q, \varepsilon, Z)| \leqslant 1$

- **PDA**在每一个状态**q**和一个栈顶符号下的动作都是惟一的。
- 关键
  - 对于$\forall(q, Z) \in Q \times \Gamma$，**M**此时如果有非空移动，就不能有空移动。
  - 每一种情况下的移动都是惟一的。

7.1 基本定义

- **例 7-2** 构造接受$L=\{ww^T|w\in\{0,1\}^*\}$的PDA。
- 差异

$\delta(q_0, 0, A)=\{(q_0, AA), (q_1, \varepsilon)\}$

0是w中的0或者是w^T的首字符0；

$\delta(q_0, 1, B)=\{(q_0, BB), (q_1, \varepsilon)\}$

1是w中的1或者是w^T的首字符1。

7.2 PDA与CFG等价

- 对于任意PDA M_1，存在PDA M_2，使得 $L(M_2)=N(M_1)$;
- 对于任意PDA M_1，存在PDA M_2，使得 $N(M_2)= L(M_1)$。
- CFL可以用空栈接受语言的PDA接受。
- PDA接受语言可以用CFG描述。

7.2.1 PDA用空栈接受和用终止状态接受等价

定理 7-1 对于任意**PDA** M_1，存在**PDA** M_2，使得$N(M_2)= L(M_1)$。

证明要点：

（1） 构造。

设**PDA** $M_1 = (Q, \Sigma, \Gamma, \delta_1, q_{01}, Z_{01}, F)$

取**PDA** $M_2 = (Q \cup \{q_{02}, q_e\}, \Sigma, \Gamma \cup \{Z_{02}\}, \delta, q_{02}, Z_{02}, F)$

其中$Q \cap \{q_{02}, q_e\}= \Gamma \cap \{Z_{02}\}= \Phi$。

7.2.1 PDA用空栈接受和用终止状态接受等价

$\delta_2(q_{02}, \varepsilon, Z_{02})=\{(q_{01}, Z_{01}Z_{02})\}$

$\forall(q,a,Z)\in Q\times\Sigma\times\Gamma$,

$\delta_2(q,a,Z)=\delta_1(q,a,Z)$;

$\forall(q,Z)\in(Q-F)\times\Gamma$, $\delta_2(q,\varepsilon,Z)=\delta_1(q,\varepsilon,Z)$;

$\forall(q,Z)\in F\times\Gamma$

$\delta_2(q,\varepsilon,Z)=\delta_1(q,\varepsilon,Z)\cup\{(q_e,\varepsilon)\}$;

$\forall q\in F$, $\delta_2(q,\varepsilon,Z_{02})=\{(q_e,\varepsilon)\}$;

$\forall Z\in\Gamma\cup\{Z_{02}\}$, $\delta_2(q_e,\varepsilon,Z)=\{(q_e,\varepsilon)\}$

7.2.1 PDA用空栈接受和用终止状态接受等价

（2） 证明$N(M_2)= L(M_1)$。

$x \in L(M_1)$

$\Leftrightarrow (q_{01}, x, Z_{01}) \vdash_{M1}^{*} (q, \varepsilon, \gamma)$且$q \in F$

$\Leftrightarrow (q_{01}, x, Z_{01}Z_{02}) \vdash_{M1}^{*} (q, \varepsilon, \gamma Z_{02})$ 且 $q \in F$

$\Leftrightarrow (q_{01}, x, Z_{01}Z_{02}) \vdash_{M2}^{*} (q, \varepsilon, \gamma Z_{02})$ 且 $q \in F$

7.2.1 PDA用空栈接受和用终止状态接受等价

$\Leftrightarrow (q_{01}, x, Z_{01}Z_{02}) \vdash_{M2}^{*} (q, \varepsilon, \gamma Z_{02}) \vdash_{M2}^{*} (q_e, \varepsilon, \varepsilon)$且$q \in F$

$\Leftrightarrow (q_{01}, x, Z_{01}Z_{02}) \vdash_{M2}^{*} (q_e, \varepsilon, \varepsilon)$

$\Leftrightarrow (q_{02}, x, Z_{02}) \vdash_{M2} (q_{01}, x, Z_{01}Z_{02}) \vdash_{M2}^{*} (q_e, \varepsilon, \varepsilon)$

$\Leftrightarrow (q_{02}, x, Z_{02}) \vdash_{M2}^{*} (q_e, \varepsilon, \varepsilon)$

$\Leftrightarrow x \in N(M_2)$

7.2.1 PDA用空栈接受和用终止状态接受等价

定理 7-2 对于任意**PDA** M_1，存在**PDA** M_2，使得$L(M_2)= N(M_1)$。

证明要点：

(1)构造。

设**PDA** $M_1 = (Q, \Sigma, \Gamma, \delta_1, q_{01}, Z_{01}, \Phi)$

7.2.1 PDA用空栈接受和用终止状态接受等价

取PDA $M_2 = (Q \cup \{q_{02}, q_f\}, \Sigma, \Gamma \cup \{Z_{02}\}, \delta, q_{02}, Z_{02}, \{q_f\})$

其中$Q \cap \{q_{02}, q_f\} = \Gamma \cap \{Z_{02}\} = \Phi$。$\delta_2$的定义如下，

$\delta_2(q_{02}, \varepsilon, Z_{02}) = \{(q_{01}, Z_{01}Z_{02})\}$

对于$\forall (q, a, Z) \in Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$，$\delta_2(q, a, Z) = \delta_1(q, a, Z)$。

$\delta_2(q, \varepsilon, Z_{02}) = \{(q_f, \varepsilon)\}$

7.2.1 PDA用空栈接受和用终止状态接受等价

(2) 证明$L(M_2)= N(M_1)$。

$x \in L(M_2)$

$\Leftrightarrow (q_{02}, x, Z_{02}) \vdash_{M2}^{*} (q_f, \varepsilon, \varepsilon)$

$\Leftrightarrow (q_{02}, x, Z_{02}) \vdash_{M2} (q_{01}, x, Z_{01}Z_{02})$

$\Leftrightarrow (q_{02}, x, Z_{02}) \vdash_{M2} (q_{01}, x, Z_{01}Z_{02}) \vdash_{M2}^{*} (q_f, \varepsilon, \varepsilon)$。

$\Leftrightarrow (q_{01}, x, Z_{01}Z_{02}) \vdash_{M2}^{*} (q, \varepsilon, Z_{02})$且$(q, \varepsilon, Z_{02}) \vdash_{M2}^{*} (q_f, \varepsilon, \varepsilon)$

$\Leftrightarrow (q_{01}, x, Z_{01}Z_{02}) \vdash_{M1}^{*} (q, \varepsilon, Z_{02})$。

$\Leftrightarrow (q_{01}, x, Z_{01}) \vdash_{M1}^{*} (q, \varepsilon, \varepsilon)$。

$\Leftrightarrow x \in N(M_1)$。

7.2.2 PDA与CFG等价

定理 7-3 对于任意**CFL L**，存在**PDA M**，使得**N(M)=L**。

证明要点：先考虑识别**L-{ε}**的**PDA**，然后再考虑对 ε 的处理问题。

7.2.2 PDA与CFG等价

(1) 构造PDA。

设GNF G=(V，T，P，S)，使得L(G)=L-{ε}。

取PDA M=({q}，T，V，δ，q，S，Φ)

对于任意的A∈V，a∈T，

$\delta(q, a, A)=\{(q, \gamma) \mid A \rightarrow a\gamma \in P\}$

也就是说，(q，γ) ∈ δ(q，a，A)的充分必要条件是A→aγ∈P。

7.2.2 PDA与CFG等价

(2) 证明构造的正确性：**N(M)=L-{ε}**。

施归纳于**w**的长度**n**，证明

(q，w，S) $\vdash_M^n$**(q，ε，α)**的充分必要条件为$S\Rightarrow^n w\alpha$。

并且在假设结论对**n=k**成立，而证明结论对**n=k+1**成立时，取**w=xa**，**|x|=k**，**a∈T**。在证明必要性时有如下过程，充分性的证明过程是倒退回来。

7.2.2 PDA与CFG等价

$(q,w,S)=(q,\ xa,\ S)\vdash_M^k(q,\ a,\ \gamma)\vdash_M(q,\ \varepsilon,\ \alpha)$

此 时 必 定 存 在 $A\in V$ ， $\gamma=A\beta_1$ ，$(q,\beta_2)\in\delta(q,a,A)$。

$(q,a,\gamma)=(q,a,A\beta_1)\vdash_M(q,\varepsilon,\beta_2\beta_1)=(q\ ,\ \varepsilon\ ,\ \alpha)$。

由$(q,\ \beta_2)\in\delta(q,\ a,\ A)$就可以得到$A\to a\beta_2\in P$，再由归纳假设，得到

$S\Rightarrow^k xA\beta_1$。

合起来就有

$S\Rightarrow^k xA\beta_1\Rightarrow xa\beta_2\beta_1$。

7.2.2 PDA与CFG等价

(3)考虑 ε ∈**L**的情况。

先按照**(1)**的构造方法构造出**PDA**

$M=(\{q\}, T, V, \delta, q, S, \Phi)$

使得$N(M)=L-\{\varepsilon\}$。然后取

$M_1=(\{q, q_0\}, T, V\cup\{Z\}, \delta_1, q_0, Z, \Phi)$

其中，$q_0\neq q$，$Z\notin V$，令

$\delta_1(q_0, \varepsilon, Z)=\{(q_0, \varepsilon), (q, Z_0)\}$,

对于$\forall(a, A)\in T\times V$

$\delta_1(q, a, A)=\delta(q, a, A)$

7.2.2 PDA与CFG等价

定理 7-4 对于任意的**PDA M**，存在**CFG G**使得**L(G)=N(M)**。

证明要点：

(1) 构造

对应$(q_1，A_1A_2…A_n)\in\delta(q，a，A)$难以用产生式$[q，A]\rightarrow a[q_1，A_1A_2…A_n]$模拟。

同样也难以用$[q，A]\rightarrow a[q_1，A_1][q_2，A_2]…[q_n，A_n]$模拟。

7.2.2 PDA与CFG等价

- PDA的移动$(q_1，A_1A_2…A_n)\in\delta(q，a，A)$需要用如下形式的产生式模拟。
  - $[q，A，q_{n+1}]\rightarrow a[q_1，A_1，q_2][q_2，A_2，q_3]…[q_n，A_n，q_{n+1}]$
  - $q_2，…，q_n$是分别对应PDA恰“处理完”A_1进而处理$A_2，…，$恰“处理完”A_{n-1}进而处理A_n的状态。当然就有了恰“处理完”A_n而进入的状态q_{n+1}，这个状态就是“处理完”A后其次栈顶变为栈顶的状态。

7.2.2 PDA与CFG等价

取CFG $G=(V, \Sigma, P, S)$，其中：

$V=\{S\} \cup Q \times \Gamma \times Q$

$P=\{S \rightarrow [q_0, Z_0, q] | q \in Q\} \cup$

$\cup \{[q, A, q_{n+1}] \rightarrow a[q_1, A_1, q_2][q_2, A_2, q_3] \ldots [q_n, A_n, q_{n+1}] \mid (q_1, A_1A_2 \ldots A_n) \in \delta(q, a, A)$ 且 $a \in \Sigma \cup \{\varepsilon\}$，$q_2$、$q_3$、…、$q_n$、$q_{n+1} \in Q$ 且 $n \geqslant 1\} \cup$

$\cup \{[q, A, q_1] \rightarrow a | (q_1, \varepsilon) \in \delta(q, a, A)\}$

7.2.2 PDA与CFG等价

(2) 构造的正确性。

- 先证一个更一般的结论：**[q，A，p]** $\Rightarrow^{*}$**x**的充分必要条件是**(q，x，A)** $\vdash$**(p，ε，ε)**，然后根据这个一般的结论得到**q=q_0，A=S**时的特殊结论——构造的正确性。
- 必要性：设 **[q，A，p]** $\Rightarrow^{i}$ **x**，施归纳于**i**，证明**(q，x，A)** $\vdash^{*}$**(p，ε，ε)**
- 充分性：设**(q，x，A)** $\vdash^{i}$**(p，ε，ε)**成立，施归纳于**i**证明**[q，A，p]** $\Rightarrow^{*}_{x}$。

7.3 小结

- **PDA M是一个七元组：M= (Q，Σ，Γ，δ，q_0，Z_0，F)**
- 它是CFL的识别模型，它比FA多了栈符号，这些符号和状态一起用来记录相关的语法信息。
- 在决定移动时，它将栈顶符号作为考虑的因素之一。

7.3 小结

- **PDA**可以用终态接受语言，也可以用空栈接受语言。
- 与**DFA**不同，$\forall(q, a, Z) \in Q \times \Sigma \times \Gamma$，**DPDA**仅要求$|\delta(q, a, Z)|+|\delta(q, \varepsilon, Z)| \leqslant 1$

7.3 小结

- 关于CFG和PDA主要有如下结论：
- (1)　对于任意PDA M_1，存在PDA M_2，使得$N(M_2)= L(M_1)$;
- (2)　对于任意PDA M_1，存在PDA M_2，使得$L(M_2)= N(M_1)$;
- (3)　对于任意CFL L，存在PDA M，使得N(M)=L;
- (4)　对于任意的PDA M，存在CFG G使得L(G)=N(M)。

第8章 CFL的性质

- 本章讨论CFL的性质
- 主要内容
  - CFL的泵引理及其应用、Ogden引理。
  - 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

第8章 上下文无关语言的性质

- **CFL的判定算法。**
  - 判定**CFG**产生的语言是否为空、有穷、无穷。
  - 一个给定的符号串是否为该文法产生的语言的一个句子等问题。
- 重点
  - **CFL**的封闭性、CFL的泵引理。
- 难点
  - **CFL**的泵引理的应用、**CFL**的同态原象是**CFL**。

8.1 上下文无关语言的泵引理

- 启发
- **RG G=(V，T，P，S)，使得L(G)=L，当x足够长时，如|x|≥|V|+1时，存在u、v、w∈T*，|v|≥1，使得x=uvw，当G为右线性文法时，必定存在语法变量A，使得如下派生成立：**
- $S \Rightarrow^* uA \Rightarrow^* uvA \Rightarrow^* \ldots\ldots \Rightarrow^* uv^iA \Rightarrow^* uv^iw$
- **V是可以被重复任意多次的字串！**
- **CFL也有类似性质？**

8.1 上下文无关语言的泵引理

- **CFL的自嵌套特性：如果L是一个CFL，CFG G=(V，T，P，S)是产生L的文法。当L是一个无穷语言时，必存在$w \in L$，$A \in V$，α，$\beta \in (V \cup T)^*$，且α和β中至少有一个不为ε，使得如下派生成立**

$$S \Rightarrow^* \gamma A \delta \Rightarrow^+ \gamma \alpha A \beta \delta \Rightarrow^+ z$$

- 文法G中存在有如下形式的派生

$$A \Rightarrow^+ \alpha A \beta$$

8.1 上下文无关语言的泵引理

- 这种类型的派生预示着
- $S \Rightarrow^* \gamma A \delta \Rightarrow^+ \gamma \alpha A \beta \delta \Rightarrow^+ z$
- 并且
- $S \Rightarrow^* \gamma A \delta \Rightarrow^+ \gamma \alpha^n A \beta^n \delta \Rightarrow^+ z'$
- 设 $\alpha \Rightarrow^* v$ ， $\beta \Rightarrow^* x$ ， $\gamma \Rightarrow^* u$ ， $A \Rightarrow^* w$ ， $\delta \Rightarrow^* y$
- 可以得到如下派生

8.1 上下文无关语言的泵引理

$S \Rightarrow^* \gamma A \delta$

$\Rightarrow^* u \alpha A \beta \delta$

$\Rightarrow^* u \alpha A \beta y$

…

$\Rightarrow^* u \alpha^n A \beta^n y$

$\Rightarrow^* uv^nAx^ny$

$\Rightarrow^* uv^nwx^ny$

8.1 上下文无关语言的泵引理

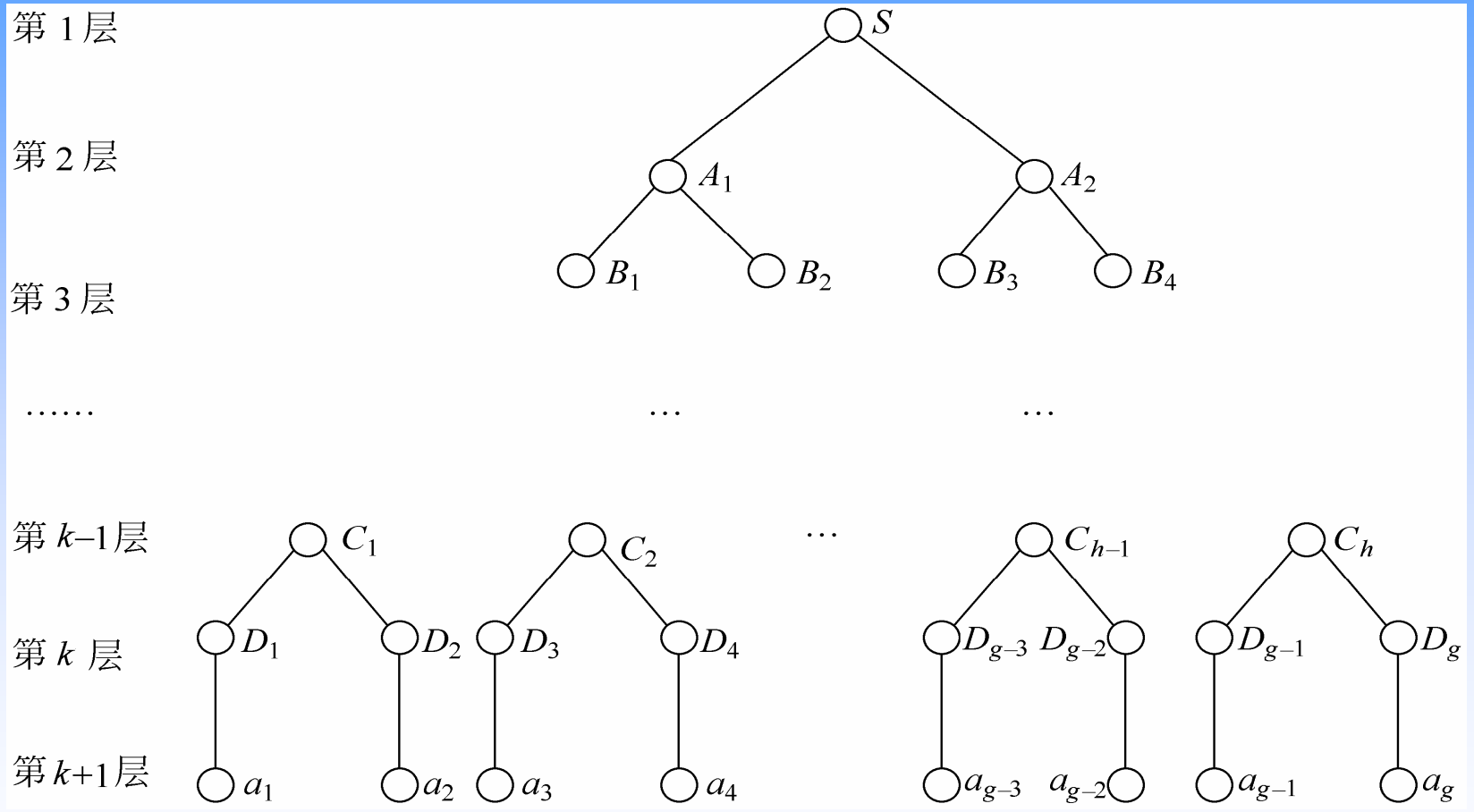
引理 8-1 （CFL的泵引理）对于任意的CFL L，存在仅仅依赖于L的正整数N，对于任意的$z \in L$，当$|z| \geqslant N$时，存在u，v，w，x，y，使得z=uvwxy，同时满足：

(1) $|vwx| \leqslant N$;

(2) $|vx| \geqslant 1$;

(3) 对于任意的非负整数i，$uv^iwx^iy \in L$。

8.1 上下文无关语言的泵引理



8.1 上下文无关语言的泵引理

- 证明要点：

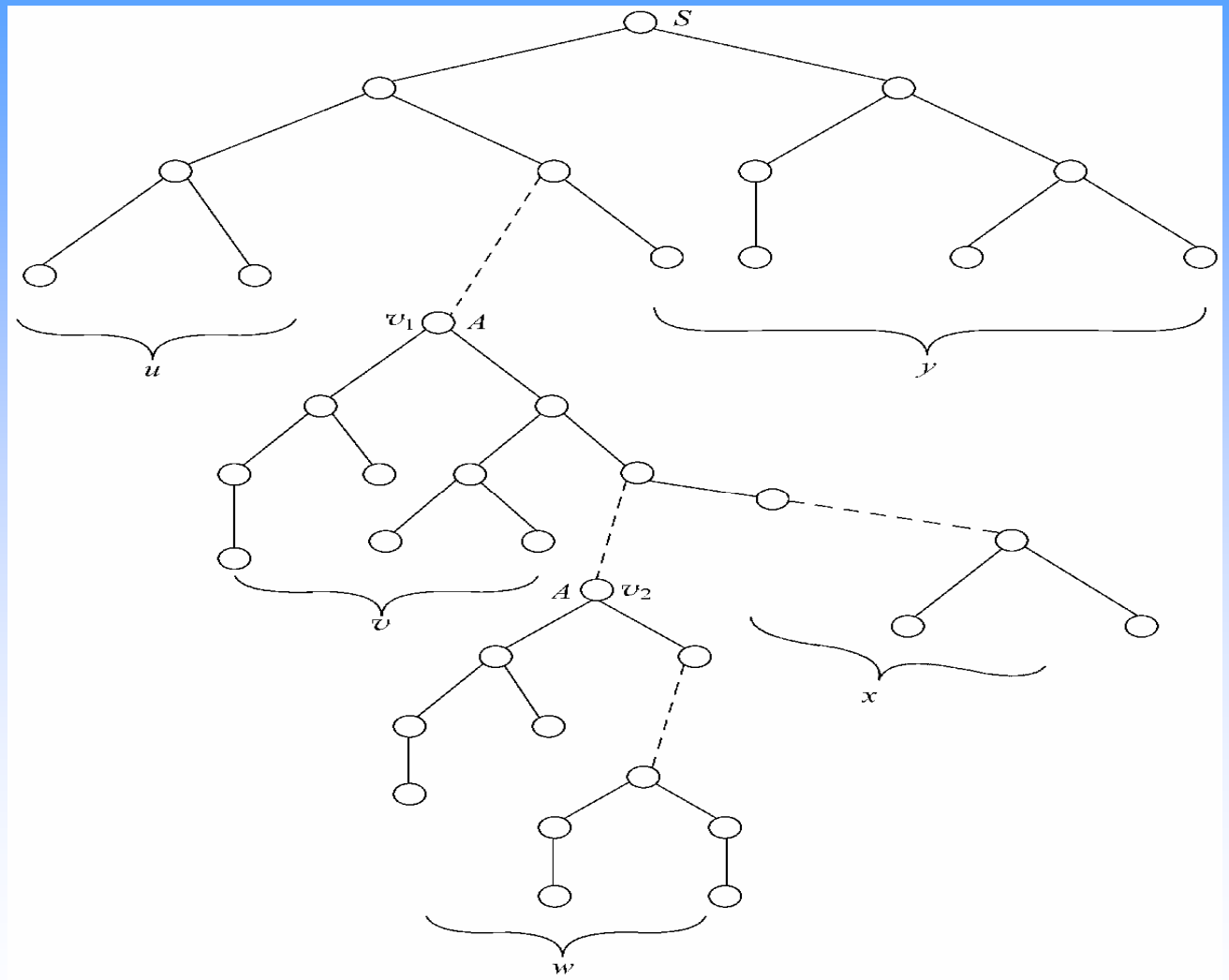
（1）用CNF作为CFL的描述工具。

（2）对于任意的$z \in L$，当k是z的语法树的最大路长时，必有$|z| \leqslant 2^{k-1}$成立。

（3）仅当z的语法树呈图8-1所示的满二元树时，才有$|z|=2^{k-1}$，其他时候均有$|z|<2^{k-1}$。

（4） 取$N=2^{|V|}=2^{|V|+1-1}$，$z \in L$，$|z| \geqslant N$。

8.1 上下文无关语言的泵引理



8.1 上下文无关语言的泵引理

（5） 取z的语法树中的最长的一条路p，p中的非叶子结点中必定有不同的结点标有相同的语法变量。

（6） p中最接近叶子且都标有相同的语法变量A的两个结点为v_1、v_2，如图8-2所示。

8.1 上下文无关语言的泵引理

- 结点v_1左边的所有叶子结点的标记从左到右构成的字符串为**u**；
- 结点v_1的为根的子树中，v_2左边的所有叶子结点的标记从左到右构成的字符串为**v**；
- 结点v_2为根的子树的结果为**w**；
- 结点v_1为根的子树中v_2右边的所有叶子结点的标记从左到右构成的字符串为**x**；
- 结点v_1右边的所有叶子结点的标记从左到右构成的字符串为**y**。

8.1 上下文无关语言的泵引理

- **z=uvwxy**
- 注意到v_1-子树的最大路长小于等于|V|+1，所以，v_1的结果vwx满足：$|vwx| \leqslant 2^{(|V|+1)-1}=2^{|V|}=N$
- v_1的后代v_2标记为变量A，所以，$|vx| \geqslant 1$。
- 此时有
- $S \Rightarrow^* uAy \Rightarrow^+ uvAxy \Rightarrow^+ uvwxy$
- 显然，对于i=0，1，2，3，…
- $A \Rightarrow^* v^iAx^i \Rightarrow^+ v^iwx^i$

8.1 上下文无关语言的泵引理

- 总结上述推导

（7） $S \Rightarrow^{*} uAy \Rightarrow^{+} uvAxy \Rightarrow^{+} uvwxy=z$，$|vwx| \leqslant 2^{(|V|+1)-1}=2^{|V|}=N$，$|vx| \geqslant 1$。

（8）对于任意非负整数i，$S \Rightarrow^{*} uAy \Rightarrow^{+} uv^{i}Ax^{i}y \Rightarrow^{+} uv^{i}wx^{i}y$。

8.1 上下文无关语言的泵引理

- **例 8-1** 证明$L=\{a^nb^nc^n|n\geqslant 1\}$不是CFL。

  证明：

  取$z=a^Nb^Nc^N\in L$

  ⑴ $v=a^h$，$x=b^f$，$h+f\geqslant 1$。

- $uv^iwx^iy=a^{N+(i-1)h}b^{N+(i-1)f}c^N$，
  - 当$i\neq 1$时，$N+(i-1)h\neq N$和$N+(i-1)f\neq N$中至少有一个成立。
  - $uv^iwx^iy=a^{N+(i-1)h}b^{N+(i-1)f}c^N\notin L$。

8.1 上下文无关语言的泵引理

⑵ $v=b^h$，$x=c^f$，$h+f \geqslant 1$。

- $uv^iwx^iy=a^Nb^{N+(i-1)h}c^{N+(i-1)f}$
  - 当$i \neq 1$时，$N+(i-1)h \neq N$和$N+(i-1)f \neq N$中至少有一个成立。
  - $uv^iwx^iy=a^{Nh}b^{N+(i-1)}c^{N+(i-1)f} \notin L$。
- 这些都与泵引理矛盾，所以，L不是CFL。

8.1 上下文无关语言的泵引理

- **例8-2** 证明$L=\{a^nb^mc^nd^m|n，m\geqslant 1\}$不是CFL。

  证明：

  取$z= a^Nb^Nc^Nd^N$

  $v=a^h$、$x=b^f$；$v=b^h$、$x=c^f$；$v=c^h$、$x=d^f$，$h+f\geqslant 1$等4种情况。

8.1 上下文无关语言的泵引理

- 设$v=a^h$、$x=b^f$，并且$h+f \geqslant 1$
- $uv^iwx^iy=a^{N+(i-1)h}b^{N+(i-1)f}c^Nd^N$
- 当$i \neq 1$时，$N+(i-1)h \neq N$和$N+(i-1)f \neq N$至少有一个成立。
- $uv^iwx^iy=a^{N+(i-1)h}b^{N+(i-1)f}c^Nd^N \notin L$

8.1 上下文无关语言的泵引理

- 同理可以证明，当$v=b^h$、$x=c^f$或者$v=c^h$、$x=d^f$，$h+f \geqslant 1$时，

  $uv^iwx^iy=a^{N+(i-1)h}b^{N+(i-1)f}c^Nd^N \notin L$

  对$i \neq 1$成立。
- 由泵引理，L不是CFL。

8.1 上下文无关语言的泵引理

- **$L=\{a^n b^m c^k | n\neq m，m\neq k，k\neq n\}$ 是CFL么？**
- **取$z=a^N b^{N+n} c^{N+m}$其中，$n\neq m$，$n\neq 0$，$m\neq 0$。**
- **只能让v或者x由若干个a组成，才有可能随着i的变化，使得uv^iwx^iy中a的个数与b的个数或者c的个数相同。**
- **当$v=a^k$，$x=b^h$（$N\geqslant k\geqslant 1$），显然，只要令k=h，就能保证对于任意的i，uv^iwx^iy中字符a的个数永远不能等于字符b的个数。**
- **希望uv^iwx^iy中字符a的个数等于字符c的个数。**

8.1 上下文无关语言的泵引理

- 想办法保证在**N≥k≥1**的条件下，总能找到一个**i**，使得**uv^iwx^iy**中字符**a**的个数能够等于**c**的个数。
- 在**uv^iwx^iy**中，字符**a**的个数为**N+(i-1)k**，而**c**的个数为**N+m**，即：**N+(i-1)k=N+m**。
- 从而要求，无论**k**取**N**到**1**的任何一个值，**i=m/k+1**必须是一个整数。

8.1 上下文无关语言的泵引理

- 取$z=a^N b^{N+N!} c^{N+2N!}$
- $v=a^k$，$x=b^h$，$N \geqslant k \geqslant 1$
- 当$i= 2N!/k+1$时，有

$$uv^i wx^i y=a^{N+(i-1)k} b^{N+N!+(i-1)h} c^{N+2N!}$$

$$= a^{N+(2N!/k+1-1)k} b^{N+N!+(2N!/k+1-1)h} c^{N+2N!}$$

$$= a^{N+(2N!/k)k} b^{N+N!+(2N!/k)h} c^{N+2N!}$$

$$= a^{N+2N!} b^{N+N!+(2N!/k)h} c^{N+2N!}$$

- $uv^i wx^i y= a^{N+2N!} b^{N+N!+(2N!/k)h} c^{N+2N!} \notin L$。

8.1 上下文无关语言的泵引理

- 问题：当取$v=b^k$，$x=c^h$，$N \geqslant k \geqslant 1$，时，我们就无法在找到矛盾了。
- 我们必须把目标锁定在**a**上！
- 因为这能保证找出“矛盾”！
- 所以我们实在是对**a**太“感兴趣”了！
- 称这些令我们感兴趣的字符为**特异点(distinguished position)**。

8.1 上下文无关语言的泵引理

- ？是否可以要求**v**和**x**中必须含有**a**。
- 这样必须修改原有的泵引理！
- 定义**z**的语法树满足下列条件的非叶子结点为**分支点(branch point)**：该结点有两个儿子，并且它的这两个儿子均有特异点后代。

8.1 上下文无关语言的泵引理

引理 8-2 （Ogden引理） 对于任意的CFL L，存在仅仅依赖于L的正整数N，对于任意的$z \in L$，当z中至少含有N个特异点时，存在u，v，w，x，y，使得$z=uvwxy$，同时满足:

(1) $|vwx|$中特异点的个数$\leqslant N$;

(2) $|vx|$中特异点的个数$\geqslant 1$;

(3) 对于任意的非负整数i，$uv^iwx^iy \in L$。

8.1 上下文无关语言的泵引理

- 证明要点：

（1） 与CFL的泵引理的证明类似。

（2） 用CNF作为CFL的描述工具。

（3） 取$N= 2^{|V|}+1$。

（4） 构造从树根到叶子的含有最多分支点的路径p。参考图8-2。

8.1 上下文无关语言的泵引理

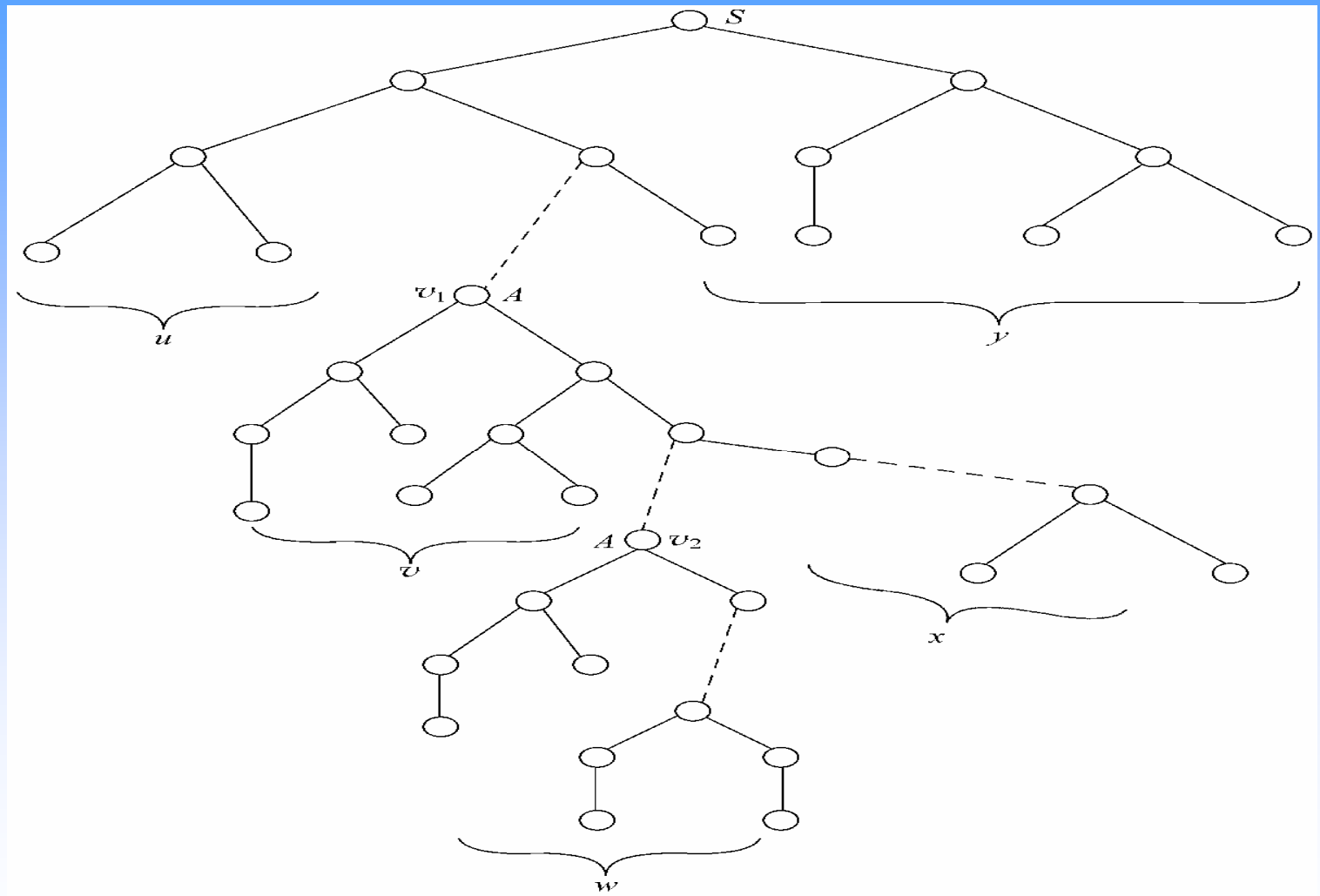


图8-2 z的派生树

8.1 上下文无关语言的泵引理

其中：

- 结点v_1左边的所有叶子结点的标记从左到右构成的字符串为**u**；
- 结点v_1的为根的子树中v_2左边的所有叶子结点的标记从左到右构成的字符串为**v**；
- 结点v_2为根的子树的结果为**w**；
- 结点v_1的为根的子树中v_2右边的所有叶子结点的标记从左到右构成的字符串为**x**；
- 结点v_1右边的所有叶子结点的标记从左到右构成的字符串为**y**；

8.1 上下文无关语言的泵引理

（5） p中至少有|V|+1个分支点，在这些分支结点中，至少有两个不同的结点标记有相同的变量A。

（6） 仍然参照图8-2，p中最接近叶子的且都标有相同的语法变量A的两个分支结点为v_1、v_2。

（7） $S\Rightarrow^* uAy\Rightarrow^+ uvAxy\Rightarrow^+ uvwxy=z$。vwx中最多有N个特异点，vx至少有1个特异点。

（8）对于任意的非负整数i， $S\Rightarrow^* uAy\Rightarrow^+ uv^iAx^iy\Rightarrow^+ uv^iwx^iy$。

8.1 上下文无关语言的泵引理

- **例8-3** 证明$L=\{a^nb^mc^hd^j|n=0$或者$m=h=j\}$不是CFL。

取$z=ab^Nc^Nd^N$

⑴ $v=a$，$x=b^k$，$k\neq0$

此时$uv^2wx^2y=aab^{N+k}c^Nd^N\notin L$;

⑵ $v=b^k$，$x=c^g$，$k\neq0$，$g\neq0$

此时$uv^2wx^2y=ab^{N+k}c^{N+g}d^N\notin L$;

⑶ $v=b^k$，$x=d^g$，$k\neq0$，$g\neq0$

此时$uv^2wx^2y=aab^{N+k}c^Nd^{N+g}\notin L$;

与Ogden引理矛盾，所以，L不是CFL。

8.2 CFL的封闭性

定理 8-1 CFL 在并、乘积、闭包运算下是封闭的。

证明要点：

令CFG

- $G_1=(V_1, T_1, P_1, S_1)$, $L(G_1)=L_1$,
- $G_2=(V_2, T_2, P_2, S_2)$, $L(G_2)=L_2$,
- $V_1 \cap V_2 = \Phi$, 且S_3, S_4, $S_5 \notin V_1 \cup V_2$。

8.2 CFL的封闭性

$G_3=(V_1 \cup V_2 \cup \{S_3\}, T_1 \cup T_2, P_1 \cup P_2 \cup \{ S_3 \rightarrow S_1 | S_2 \}, S_3)$

$G_4=(V_1 \cup V_2 \cup \{S_4\}, T_1 \cup T_2, P_1 \cup P_2 \cup \{ S_4 \rightarrow S_1 S_2 \}, S_4)$

$G_5=(V_1 \cup \{S_5\}, T_1, P_1 \cup \{ S_5 \rightarrow S_1 S_5 | \varepsilon \}, S_5)$

显然，G_3、G_4、G_5都是CFG，并且

$L(G_3)=L_1 \cup L_2$

$L(G_4)= L_1 L_2$

$L(G_5)= L_1^*$

8.2 CFL的封闭性

定理 8-2 CFL 在交运算下不封闭的。

证明要点：

设$L_1=\{0^n1^n2^m|n,m\geqslant 1\}$，$L_2=\{0^n1^m2^m|n,m\geqslant 1\}$。

G_1：$S_1\rightarrow AB$

$A\rightarrow 0A1|01$

$B\rightarrow 2B|2$

G_2：$S\rightarrow AB$

$A\rightarrow 0A|0$

$B\rightarrow 1B2|12$

8.2 CFL的封闭性

- 显然，$L(G_1)=L_1$、$L(G_2)=L_2$ 。
- $L= L_1 \cap L_2=\{0^n1^n2^n | n \geqslant 1\}$ 不是CFL。
- 所以，CFL在交运算下是不封闭的。

8.2 CFL的封闭性

推论8-1 CFL在补运算下是不封闭的。

证明要点：

由于CFL对并运算的封闭性、对交运算的不封闭性和下列式子，可以得出CFL对补运算是不封闭的结论。

$$L_1 \cap L_2 = \overline{\overline{L_1 \cap L_2}} = \overline{\overline{L_1} \cup \overline{L_2}}$$

8.2 CFL的封闭性

定理8-3 CFL与RL的交是CFL。

证明：

（1）构造

设PDA $M_1=(Q_1, \Sigma, \Gamma, \delta_1, q_{01}, Z_0, F_1)$

$L_1=L(M_1)$

DFA $M_2=(Q_2, \Sigma, \delta_2, q_{02}, F_2)$

$L_2=L(M_2)$

8.2 CFL的封闭性

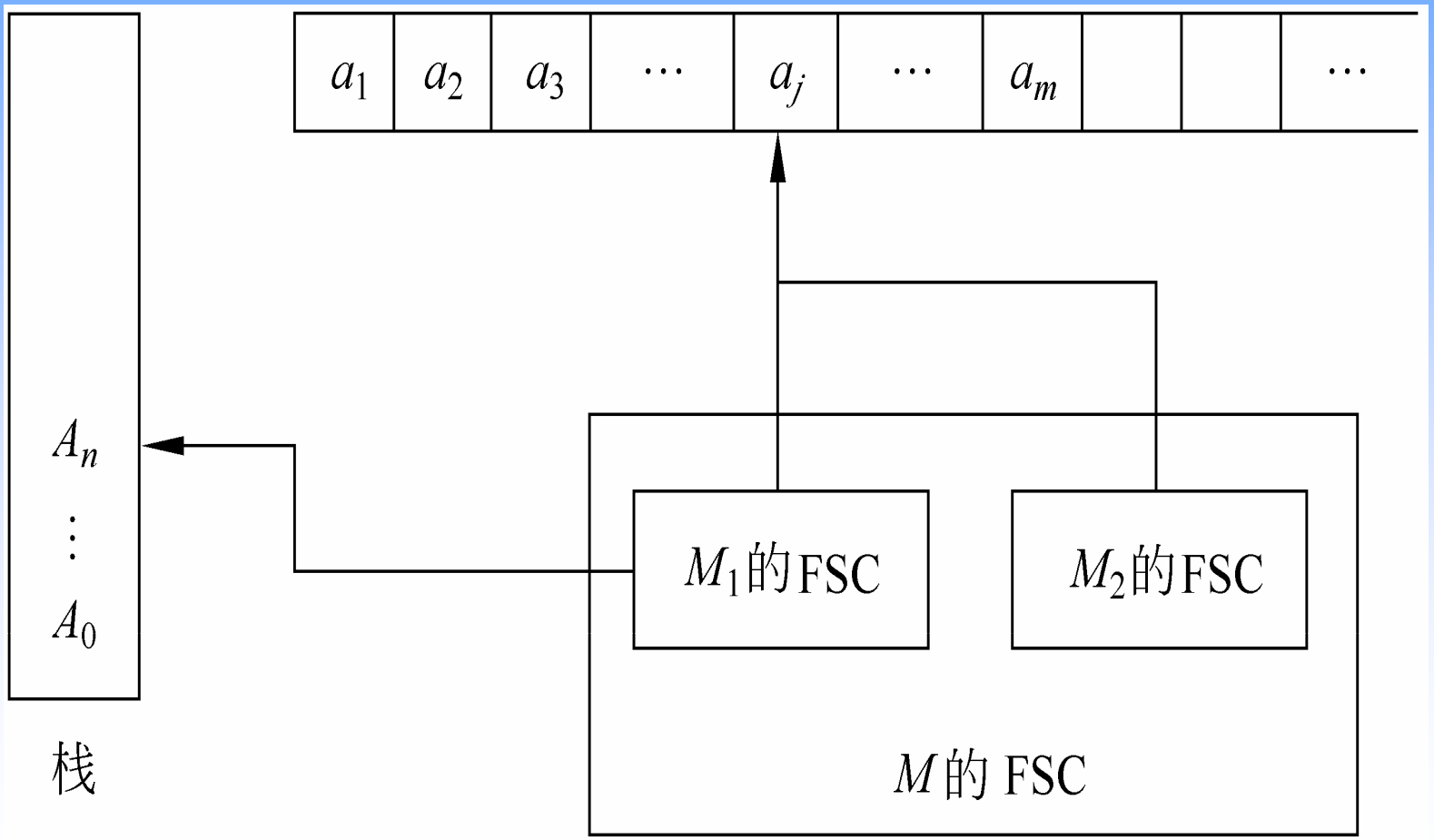
- 取PDA $M=(Q_1\times Q_2, \Sigma, \Gamma, \delta, [q_{01}, q_{02}], Z_0, F_1\times F_2)$

对$\forall([q, p], a, Z)\in (Q_1\times Q_2)\times(\Sigma\cup\{\varepsilon\})\times\Gamma$

$\delta([q, p], a, Z)=\{([q', p'], \gamma) \mid (q', \gamma)\in \delta_1(q, a, Z)$ 且 $p' = \delta(p, a)\}$

- 可以用下图标是构造的基本思想。

8.2 CFL的封闭性



8.2 CFL的封闭性

① M使用的栈就是M_1的栈。

② M的状态包括两个分量：一个为M_1的状态，用来使M的动作能准确地模拟M_1的动作；另一个为M_2的状态，用来使M的动作能准确地模拟M_1的动作。

③ 当M_1执行 ε -移动时，M_2不执行动作。

8.2 CFL的封闭性

(2) 证明构造的正确性。

施归纳于n证明：

$([q_{01}，q_{02}]，w，Z_0) \vdash_M^n ([q，p]，\varepsilon，\gamma)$

$\Leftrightarrow (q_{01}，w，Z_0) \vdash_{M1}^n (q，\varepsilon，\gamma) \& \delta(q_{02}，w) = p$

再注意到$[q，p] \in F_1 \times F_2 \Leftrightarrow q \in F_1 \ \& \ p \in F_2$

这就是说，对于$\forall x \in \Sigma^*$，

$x \in L(M) \Leftrightarrow x \in L(M_1) \ \& \ x \in L(M_2)$

所以，

$L(M)=L(M_1) \cap L(M_2)$

8.2 CFL的封闭性

- 根据本定理证明中给出的方法，可以用N个正则语言L_1、L_2、…、L_N的识别器(DFA)的有穷状态控制器构成这N个正则语言的交的识别器(DFA)：

  $M_{\cap}=(Q_1\times Q_2\times\ldots\times Q_N,\ \Sigma,\ \delta,\ [q_{10},\ q_{20},\ \ldots,\ q_{N0}],\ F_1\times F_2\times\ldots\times F_N)$

8.2 CFL的封闭性

定理8-4 CFL在代换下是封闭的。

① CFG $G=(V，T，P，S)$，使得$L=L(G)$。对于$\forall a\in T$，$f(a)$是Σ上的CFL，且CFG $G_a=(V_a，\Sigma，P_a，S_a)$，使得$f(a)=L(G_a)$。

② $f(a)$的所有句子都是由S_a派生出来的，所以，构造的基本思想是用S_a替换产生L的CFG的产生式中出现的终极符号a。

8.2 CFL的封闭性

③产生**f(L)**的文法

$$G' = (\bigcup_{a \in T} V_a \bigcup V, \sum, \bigcup_{a \in T} P_a \bigcup P', S)$$

$$P' = \{A \to A_1 A_2 \cdots A_n \mid$$

$$A \to X_1 X_2 \cdots X_n \in P \text{且}$$

$$\text{如果} X_i \in V, \text{则} A_i = X_i, \text{否则} A_i = S_{X_i} \}$$

8.2 CFL的封闭性

④ 证明$L(G')=f(L)$

设$x \in L(G')$，从而

$S \Rightarrow^*_{G'} S_aS_b\ldots S_c$

$\Rightarrow^+_{G'} x_aS_b\ldots S_c$

$\Rightarrow^+_{G'} x_ax_b\ldots S_c$

…

$\Rightarrow^+_{G'} x_ax_b\ldots x_c$

$=x$

S_a，S_b，…，$S_c \in \{S_d \mid d \in T\}$。且

$S_a \Rightarrow^*_{G'} x_a$

$S_b \Rightarrow^*_{G'} x_b$

…

$S_c \Rightarrow^*_{G'} x_c$

8.2 CFL的封闭性

由G'的定义知，

$S \Rightarrow^*_{G'} S_a S_b \ldots S_c \Leftrightarrow S \Rightarrow^*_G ab\ldots c$

并且，对于$\forall d \in T$，　$S_d \Rightarrow^*_{G'} x_d \Leftrightarrow S_d \Rightarrow^*_{Gd} x_d$

所以，$ab\ldots c \in L$，$x_a \in L(G_a)$，$x_b \in L(G_b)$，…，$x_c \in L(G_c)$

故：$x = x_a x_b \ldots x_c = f(a)f(b)\ldots f(c) = f(ab\ldots c)$

即：$x \in f(L)$。从而$L(G') \subseteq f(L)$

用类似的方法，不难证明$f(L) \subseteq L(G')$。

综上所述，定理得证。

8.2 CFL的封闭性

推论8-2 CFL的同态像是CFL。

- 注意到对任意的CFG G=(V，T，P，S)，L=L(G)。对于∀a∈T，|f(a)|=1，所以它是∑上的CFL，根据这个定理，得到此推论。

8.2 CFL的封闭性

定理8-5 CFL L的同态原像是CFL。

基本思路:

① 描述工具PDA。

② PDA $M_2=(Q_2, \Sigma_2, \Gamma, \delta_2, q_0, Z_0, F)$接受L

③ 设$T=\{a_1, a_2, \ldots, a_n\}$根据 $f(a_1)=x_1$, $f(a_2)=x_2$, …, $f(a_n)=x_n$, M_1有穷控制器中的缓冲区存放$f(a_1)$, $f(a_2)$, …, $f(a_n)$的任一后缀。

8.2 CFL的封闭性

④ 对于任意的$x \in \Sigma_1^*$，设$x=b_1b_2\ldots b_n$，M_1是否接受x，完全依据于M_2是否接受$f(b_1)f(b_2)\ldots f(b_n)$。

⑤ M_1的形式定义为

$M_1=(Q_1, \Sigma_1, \Gamma, \delta_1, [q_0, \varepsilon], Z_0, F\times\{\varepsilon\})$

$Q_1=\{[q, x] \mid q\in Q_2$，存在$a\in\Sigma_1$，x是f(a)的后缀$\}$

当M_1扫描到$a\in\Sigma_1$时，将f(a)存入有穷控制器，

$([q, f(a)], A) \in \delta_1([q, \varepsilon], a, A)$

8.2 CFL的封闭性

然后，M_1模拟M_2处理存在缓冲区中的f(a)。

- M_1用 ε -移动模拟M_2的非 ε -移动：

$(p, \gamma) \in \delta_2(q, a, A)$，⇔

$([p, x], \gamma) \in \delta_1([q, ax], \varepsilon, A)$

- M_1用 ε -移动模拟M_2的 ε -移动

$(p, \gamma) \in \delta_2(q, \varepsilon, A)$，⇔

$([p, x], \gamma) \in \delta_1([q, x], \varepsilon, A)$

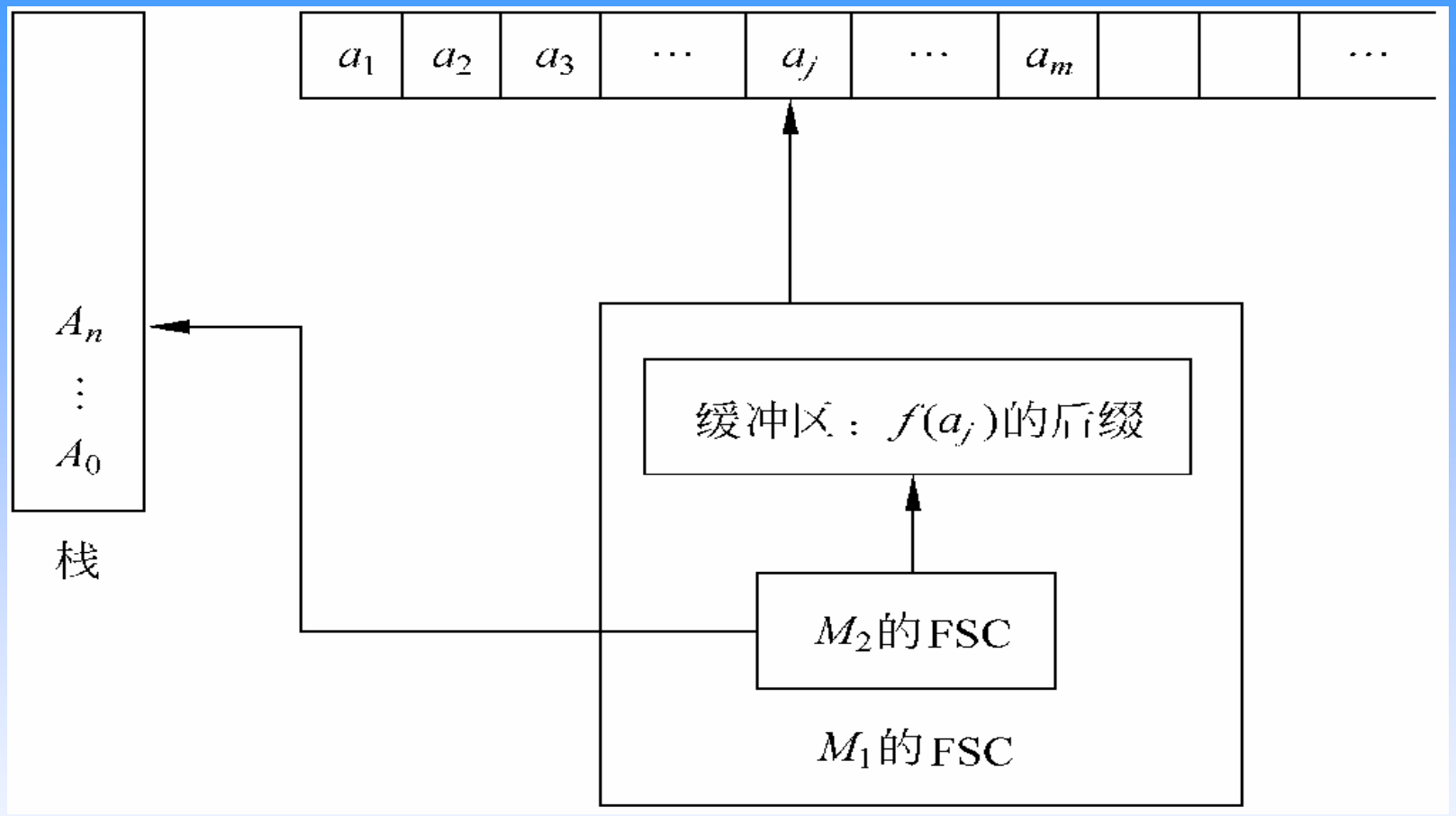


图8-4 M的构造示意图

8.2 CFL的封闭性

- 构造的正确性证明。
- 证明$x=a_1a_2\ldots a_n \in L(M_1)$的充分必要条件是$f(a_1)f(a_2)\ldots f(a_n) \in L(M_2)$，这里的关键是对状态“接续”的理解。也就是

8.2 CFL的封闭性

$([q_i, \varepsilon], a_i \ldots a_n, \gamma_i) \vdash_{M1} ([q_i, f(a_i)], a_{i+1} \ldots a_n, \gamma_i)$

$([q_i, f(a_i)], a_{i+1} \ldots a_n, \gamma_i) \vdash_{M1}^{*} ([q_{i+1}, \varepsilon], a_i \ldots a_n, \gamma_{i+1})$

成立的充分必要条件为：

$(q_0, f(a_1)f(a_2) \ldots f(a_n), Z_0) \vdash_{M2}^{*} (q_1, f(a_2) \ldots f(a_n), \gamma_1)$

$(q_i, f(a_i) \ldots f(a_n), \gamma_i) \vdash_{M2}^{*} (q_{i+1}, f(a_{i+1}) \ldots f(a_n), \gamma_{i+1})$

8.3 CFL的判定算法

- 不存在判断算法的问题：

① CFG G，是不是二义性的？

② CFL L的补是否确实不是CFL？

③ 任意两个给定CFG是等价的么？

存在判断算法的问题：

④ L是非空语言么？

⑤ L是有穷的么？

⑥ 一个给定的字符串x是L的句子么？

8.3.1 L空否的判定

- 基本思想：

  设L为一个CFL，则存在CFG G，使得L(G)=L。由算法 6-1，我们可以求出等价的CFG G′，G′中不含派生不出终极符号行的变量。显然，如果NEWV中包含G的开始符号，则L就是非空的。否则，L就是空的。因此，通过改造算法 6-1，我们可得到判定L是否为空的算法8-1。

8.3.1 L空否的判定

算法 8-1 判定CFL L是否为空。

输入：CFG G=(V，T，P，S)。

输出：G是否为空的判定；CFG G′=(V′，T，P′，S)，V′中不含派生不出终极符号行的变量，并且有L(G′)=L(G)。

主要步骤：

(1) OLDV= Φ;

(2) NEWV={A|A→w∈P且w∈T^*}

8.3.1 L空否的判定

(3) while OLDV≠NEWV do

begin

(4) OLDV=NEWV;

(5) NEWV=OLDV∪{A→α ∈P且 α ∈(T∪OLDV)*};

end

(6) V′ =NEWV;

(7) P′ ={ A→α | A→α ∈P且 A∈V′ 且 α ∈(T∪V′)*};

(8) if S∈NEWV then L(G)非空 else L(G)为空

8.3.2 L是否有穷的判定

- **可派生性图表示**(Derivability Graph of G——DG)

设**CFG G=(V，T，P，S)，G的可派生性图表示**是满足下列条件的有向图：

(1) 对于$\forall X \in V \cup T$，图中有且仅有一个标记为**X**的顶点；

(2) 如果$A \to X_1X_2\ldots X_n \in P$，则图中存在从标记为**A**的顶点分别到标记为$X_1$、$X_2$、…、$X_n$的弧；

(3) 图中只有满足条件**1**和**2**的顶点和弧。

8.3.2 L是否有穷的判定

- G的可派生性图表示中，任意两个顶点之间最多有一条相同方向的弧。
- G的可派生性图表示表达了文法G中的语法变量之间的派生关系。
- 派生$A \Rightarrow^{+} \alpha A \beta$存在的充分必要条件是G的可派生性图表示中存在一条从标记为A的顶点到标记为A的顶点的长度非0的有向回路。

8.3.2 L是否有穷的判定

定理 8-6 设CFG G=(V，T，P，S)不含无用符号，L(G)为无穷语言的充分必要条件是G的可派生性图表示中存在一条有向回路。

证明要点：

对应某一条有向回路，寻找一条从S出发，到达此回路的路，可以构造出下列形式的派生

$$S \Rightarrow^{+} \gamma A \rho \Rightarrow^{+} \gamma \alpha A \beta \rho$$

从而对任意的非负整数i，

$$S \Rightarrow^{+} \gamma A \rho \Rightarrow^{+} \gamma \alpha^{i} A \beta^{i} \rho$$

8.3.2 L是否有穷的判定

- DG中标记为终极符号的结点是无用的
- 简化的可派生性图表示(simplified derivability graph of G，SDG)
- 设CFG G=(V，T，P，S)，G的简化的可派生性图表示是从G的可派生性图表示中删除所有标记为终极符号的顶点后得到的图。

8.3.2 L是否有穷的判定

定理 8-7 设CFG G=(V，T，P，S)不含无用符号，L(G)为无穷语言的充分必要条件是G的简化的可派生性图表示中存在一条有向回路。

- **证明：与定理8-6的证明类似。**

8.3.2 L是否有穷的判定

算法 8-2 判定CFL L是否为无穷语言。

- 输入：CFG G=(V，T，P，S)。
- 输出：G是否为无穷的判定；CFG G′=(V′，T，P′，S)，V′中不含派生不出终极符号行的变量，并且有L(G′)=L(G)。

8.3.2 L是否有穷的判定

(1) 调用算法6-1、6-2；

(2) if $S \notin V'$ then L(G)为有穷的语言

else

begin

(3) 构造G′的简化的可派生性图表示SDG；

(4) if SDG中含有回路 then L(G′)为无穷语言

(5) else L(G′)为有穷有语言。

end

8.3.3 x是否为L的句子的判定

- 判断x是否为给定文法生成的句子的根本方法是看G能否派生出x。一种最简单的算法是用穷举法，这种方法又称为“试错法”,是“带回溯”的，所以效率不高。其时间复杂度为串长的指数函数。
- 典型的自顶向下的分析方法：递归子程序法、LL(1)分析法、状态矩阵法等。
- 典型的自底向上的分析方法：LR分析法、算符优先分析法。
- 这些基本的方法均只可以分析CFG的一个真子类。

8.3.3 x是否为L的句子的判定

- CYK算法的基本思想是从1到|x|，找出x的相应长度的子串的派生变量，效率较高的根本原因是它在求x的长度为i的子串y的“派生变量”时，是根据相应的CNF中的形如A→BC的产生式，使用已经求出的B是y的前缀的“派生变量”，而C是相应的后缀的“派生变量”的结果。

8.3.3 x是否为L的句子的判定

算法8-3 CYK算法。

- 输入：CNF G=(V，T，P，S)，x；
- 输出：$x \in L(G)$或者$x \notin L(G)$；
- 主要数据结构：
  - 集合$V_{i,j}$——可以派生出子串$x_{i,k}$的变量的集合。这里，$x_{i,k}$表示x的从第i个字符开始的，长度为k的字串。

8.3.3 x是否为L的句子的判定

(1) for i=1 to |x| do

(2) $V_{i,1}=\{A|A\rightarrow x_{i,1}\in P\}$;

(3) for k=2 to |x| do

(4) for i=1 to |x|-k+1 do

begin

(5) $V_{i,k}=\Phi$;

(6) for j=1 to k-1 do

(7) $V_{i,k}=V_{i,k}\cup\{A\mid A\rightarrow BC\in P$且 $B\in V_{i,j}$且 $C\in V_{i+j,k-j}\}$;

end

8.3.3 x是否为L的句子的判定

其中：语句(1)和(2)完成长度为1的子串的派生变量的计算。其时间复杂度为 |P|。语句(3)控制算法依次完成长度是2、3、…、|x|的子串的派生变量的计算；语句(4)控制完成对于串x中的所有长度为k的子串的派生变量的计算。这里的计算顺序依次为从第1个字符开始的长度为k的子串、从第2个字符开始的长度为k的子串、…、从第|x|-k+1个字符开始的长度为k的子串。语句(6)控制实现长度为k的子串的不同切分方式下的派生的可能性。

8.4 小结

本章讨论了CFL 的性质和CFL的一些判定问题。

(1) 泵引理：与RL的泵引理类似，CFL的泵引理用来证明一个语言不是 CFL。它不能证明一个语言是 CFL，而且也是采用反证法。Ogden引理是对泵引理的强化，它可以使我们将注意力集中到所给字符串的那些令我们感兴趣的地方——特异点。

8.4 小结

(2) CFL 在并、乘、闭包、代换、同态映射、逆同态映射等运算下是封闭的。

(3) CFL在交、补运算下是不封闭。

(4) 存在判定**CFG**产生的语言是否为空、是否有穷、是否无穷，以及一个给定的符号串是否为该文法产生的语言的一个句子的算法。

第9章 图灵机

- **图灵机(Turing machine)**是由图灵**(Alan MathisomTuring)**在**1936**年提出的，它是一个通用的计算模型。
- 通过研究**TM**，来研究**递归可枚举集(recursively enumerable set)**和**部分地归函数(partial recursive function)**。
- 对算法和可计算性进行研究提供形式化描述工具。

第9章 图灵机

- **有效过程(effective procedure)与算法(algorithm)。**
- 希尔伯特纲领。
- **1931年，奥地利25岁的数理逻辑学家哥德尔(Kuri Gödel)发表了著名的不完整性理论。**
- 具有有穷描述的过程是可数无穷多的，但函数却是不可数无穷多的。
- 世界上存在着许多的问题和函数，是无法用具有有穷描述的过程完成计算的——是**不可计算的(incomputable)** 。

第9章 图灵机

- 主要内容
  - **TM**作为一个计算模型，它的基本定义，即时描述，**TM**接受的语言；**TM**的构造技术；**TM**的变形；**Church-Turing**论题；通用**TM**。可计算语言、不可判定性、**P-NP**问题)。
- 重点
  - **TM**的定义、TM的构造。
- 难点
  - TM的构造。

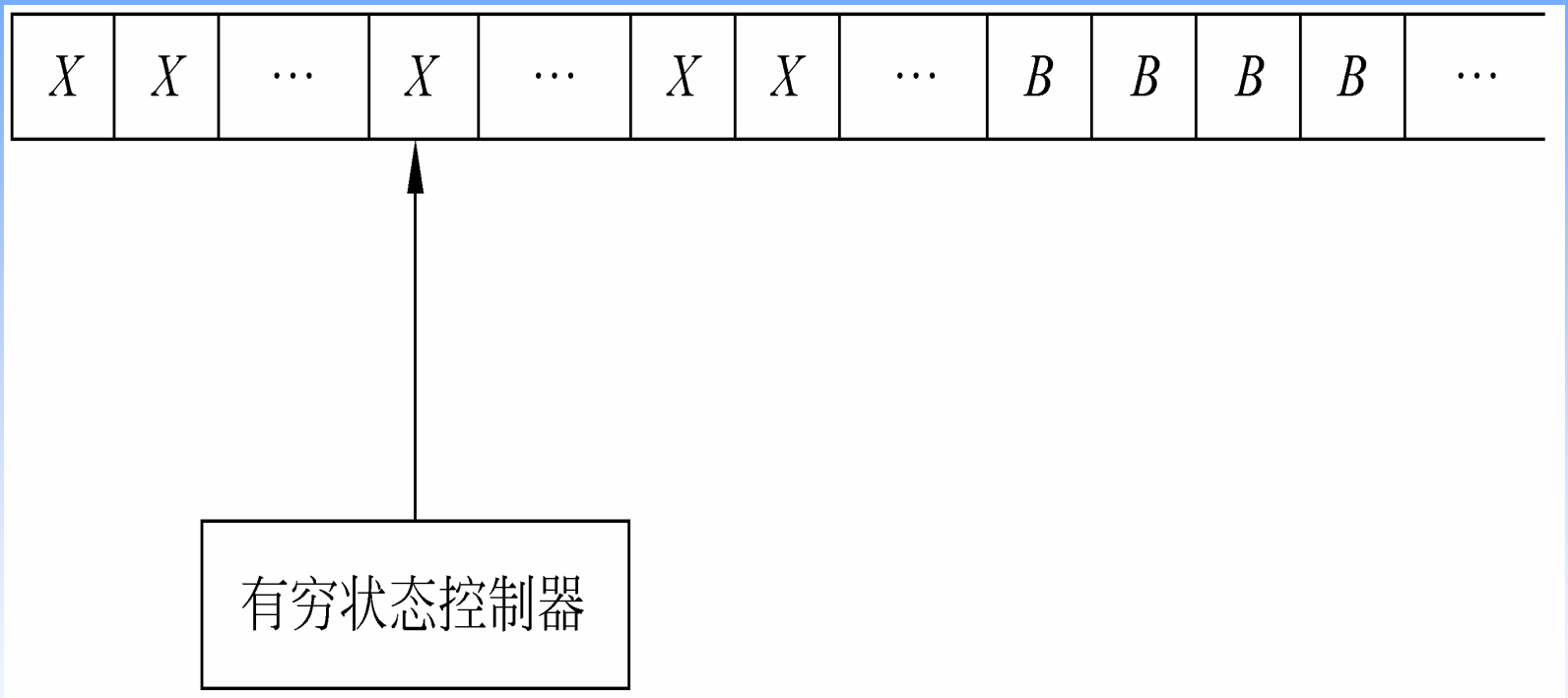
9.1 基本概念

- 图灵提出TM具有以下两个性质
  - 具有有穷描述。
  - 过程必须是由离散的、可以机械执行的步骤组成。
- 基本模型包括
  - 一个有穷控制器。
  - 一条含有无穷多个带方格的输入带。
  - 一个读头。

9.1 基本概念

- 一个移动将完成以下三个动作：
  - 改变有穷控制器的状态；
  - 在当前所读符号所在的带方格中印刷一个符号；
  - 将读头向右或者向左移一格。

直观物理模型



9.1.1 基本TM

- **图灵机(Turing machine)/基本的图灵机**

  TM M=(Q, Σ, Γ, δ ,q_0 , B , F) ,

- **Q**为状态的有穷集合，$\forall$**q**$\in$**Q**，**q**为**M**的一个状态；
- $q_0 \in Q$，是**M**的**开始状态**，对于一个给定的输入串，**M**从状态q_0启动，读头正注视着输入带最左端的符号；

9.1.1 基本TM

- **$F \subseteq Q$，是M的终止状态集，$\forall q \in F$，q为M的一个终止状态。与FA和PDA不同，一般地，一旦M进入终止状态，它就停止运行。**

- **Γ 为带符号表(tape symbol)，$\forall X \in \Gamma$，X为M的一个带符号，表示在M的运行过程中，X可以在某一时刻出现在输入带上；**

9.1.1 基本TM

- **B∈Γ，被称为空白符(blank symbol)，含有空白符的带方格被认为是空的；**
- **Σ⊆Γ-{B}为输入字母表，∀a∈Σ，a为M的一个输入符号。除了空白符号B之外，只有Σ中的符号才能在M启动时出现在输入带上；**

9.1.1 基本TM

- δ：Q×Γ→Q×Γ×{R, L}，为M的移动函数(transaction function)。
- δ(q , X)=(p , Y, R)表示M在状态q读入符号X，将状态改为p，并在这个X所在的带方格中印刷符号Y，然后将读头向右移一格；
- δ(q , X)=(p , Y , L)表示M在状态q读入符号X，将状态改为p，并在这个X所在的带方格中印刷符号Y，然后将读头向左移一格。

9.1.1 基本TM

- **例 9-1** 设$M_1=(\{q_0, q_1, q_2\},\{0, 1\},\{0, 1, B\}, \delta, q_0, B, \{q_2\})$，其中 δ 的定义如下，对于此定义，也可以用表**9-1**表示。

$$\delta(q_0, 0)= (q_0, 0, R)$$

$$\delta(q_0, 1)= (q_1, 1, R)$$

$$\delta(q_1, 0)= (q_1, 0, R)$$

$$\delta(q_1, B)= (q_2, B, R)$$

9.1.1 基本TM

	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	
q_1	$(q_1, 0, R)$		(q_2, B, R)
q_2			

9.1.1 基本TM

- 即时描述(instantaneous description, ID)
- $\alpha_1\alpha_2 \in \Gamma^*$，$q \in Q$，$\alpha_1 q \alpha_2$称为M的即时描述
  - q为M的当前状态。
  - $\alpha_1\alpha_2$为M的输入带最左端到最右的非空白符号组成的符号串或者是M的输入带最左端到M的读头注视的带方格中的符号组成的符号串
  - M正注视着α_2的最左符号。

9.1.1 基本TM

设$X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n$是M的一个ID

如果$\delta(q, X_i)=(p, Y, R)$，则，M的下一个ID为$X_1X_2\cdots X_{i-1}YpX_{i+1}\cdots X_n$

记作

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

- 表示M在ID $X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n$下，经过一次移动，将ID变成$X_1X_2\cdots X_{i-1}YpX_{i+1}\cdots X_n$。

9.1.1 基本TM

- 如果 $\delta(q, X_i)=(p, Y, L)$则，
  - 当$i \neq 1$时，M的下一个ID为

    $X_1X_2\cdots pX_{i-1}YX_{i+1}\cdots X_n$
- 记作

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

  - 表示M在ID $X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n$下，经过一次移动，将ID变成$X_1X_2\cdots pX_{i-1}YX_{i+1}\cdots X_n$;

9.1.1 基本TM

- $\vdash_M$是$\Gamma^*Q\Gamma^* \times \Gamma^*Q\Gamma^*$上的一个二元关系
  - $\vdash_M^n$表示$\vdash_M$的n次幂：$\vdash_M^n = (\vdash_M)^n$
  - $\vdash_M^+$表示$\vdash_M$的正闭包：$\vdash_M^+ = (\vdash_M)^+$
  - $\vdash_M^*$表示$\vdash_M$的克林闭包：$\vdash_M^* = (\vdash_M)^*$
- 在意义明确时，分别用$\vdash$、$\vdash^n$、$\vdash^+$、$\vdash^*$表示$\vdash_M$、$\vdash_M^n$、$\vdash_M^+$、$\vdash_M^*$。

9.1.1 基本TM

- **例 9-2** 例 **9-1**所给的M_1在处理输入串的过程中经历的**ID**变换序列。

（1）处理输入串**000100**的过程中经历的**ID**的变换序列如下：

$$q_0000100 \vdash_M 0q_000100 \vdash_M 00q_00100 \vdash_M 000q_0100$$
$$\vdash_M 0001q_100 \vdash_M 00010q_200 \vdash_M 000100q_30$$
$$\vdash_M 000100B q_4$$

9.1.1 基本TM

（2）处理输入串0001的过程中经历的ID变换序列如下：

$q_0 0001 \vdash_M 0q_0 001 \vdash_M 00q_0 01$

$\vdash_M 000q_0 1 \vdash_M 0001q_1 \vdash_M 0001Bq_2$

（3）处理输入串000101的过程中经历的ID变换序列如下：

$q_0 000101 \vdash_M 0q_0 00101 \vdash_M 00q_0 0101$

$\vdash_M 000q_0 101 \vdash_M 0001q_1 01 \vdash_M 00010q_1 1$

9.1.1 基本TM

（4）处理输入串1的过程中经历的ID变换序列如下：

$q_0 1 \vdash_M 1q_1 \vdash_M 1Bq_2$

（5）处理输入串00000的过程中经历的ID变换序列如下：

$q_0 00000 \vdash_M 0q_0 0000 \vdash_M 00q_0 000$

$\vdash_M 000q_0 00 \vdash_M 0000\ q_0 0 \vdash_M 00000q_0 B$

9.1.1 基本TM

- TM接受的语言

  $L(M)=\{x \mid x\in\Sigma^* \ \&\ q_0x \vdash_M^* \alpha_1 q \alpha_2 \ \&\ q\in F \ \&\ \alpha_1、\ \alpha_2\in\Gamma^*\}$
- TM接受的语言叫做**递归可枚举语言(recursively enumerable language，r.e.)**。
- 如果存在TM $M=(Q, \Sigma, \Gamma, \delta, q_0, B, F)$，$L=L(M)$，并且对每一个输入串x，M都停机，则称L为**递归语言(recursively language)**。

9.1.1 基本TM

- **例 9-3** 设有$M_2=(\{q_0, q_1, q_2, q_3\},\{0, 1\},\{0, 1, B\}, \delta, q_0, B, \{q_3\})$，其中 δ 的定义如下：

$\delta(q_0, 0)= (q_0, 0, R)$

$\delta(q_0, 1)= (q_1, 1, R)$

$\delta(q_1, 0)= (q_1, 0, R)$

$\delta(q_1, 1)= (q_2, 1, R)$

$\delta(q_2, 0)= (q_2, 0, R)$

$\delta(q_2, 1)= (q_3, 1, R)$

9.1.1 基本TM

	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	
q_2	$(q_2, 0, R)$	$(q_3, 1, R)$	
q_3			

9.1.1 基本TM

•为了弄清楚M_2接受的语言，需要分析它的工作过程。

（1）处理输入串**00010101**的过程中经历的**ID**变换序列如下：

$q_0 00010101 \vdash 0q_0 0010101 \vdash 00q_0 010101$

$\vdash 000q_0 10101 \vdash 0001q_1 0101 \vdash 00010q_1 101$

$\vdash 000101\ q_2 01 \vdash 000101\ 0\ q_2 1 \vdash 00010101q_3$

9.1.1 基本TM

- M_2在q_0状态下，遇到0时状态仍然保持为q_0，同时将读头向右移动一格而指向下一个符号；
- 在q_1状态下遇到第一个1时状态改为q_1，并继续右移读头，以寻找下一个1；在遇到第二个1时，动作类似，只是将状态改为q_2；当遇到第三个1时，进入终止状态q_3，此时它正好扫描完整个输入符号串，表示符号串被M_2接受。

9.1.1 基本TM

（2）处理输入串**1001100101100**的过程中经历的**ID**变换序列如下：

$q_0$1001100101100 ⊢ 1$q_1$001100101100

⊢ 10 $q_1$01100101100 ⊢ 100$q_1$1100101100

⊢ 1001 $q_2$100101100 ⊢10011$q_3$00101100

- M_2遇到第三个**1**时，进入终止状态q_3，输入串的后缀**00101100**还没有被处理。但是，由于M_2已经进入终止状态，表示符号串**1001100101100**被M_2接受

9.1.1 基本TM

（3）处理输入串000101000的过程中经历的ID变换序列如下：

$q_0000101000 \vdash 0q_000101000 \vdash 00q_00101000 \vdash 000q_0101000$
$\vdash 0001q_101000 \vdash 00010q_11000 \vdash 000101q_2000 \vdash 0001010\ q_200$
$\vdash 00010100\ q_20 \vdash 000101000\ q_2B$

- 当M_2的ID变为$000101000q_2B$时，因为无法进行下一个移动而停机,不接受输入串000101000。

9.1.1 基本TM

- M_2接受的语言是字母表{0，1}上那些至少含有3个1的0、1符号串。请读者考虑，如何构造出接受字母表{0，1}上那些含且恰含有3个1的符号串的TM。

9.1.1 基本TM

- 例 9-4 构造TM M_3，使$L(M)=\{0^n1^n2^n \mid n \geqslant 1\}$。

  分析：
  - 不能通过“数”0、1、或者2的个数来实现检查。
  - 最为原始的方法来比较它们的个数是否是相同的：消除一个0、然后消除一个1，最后消除一个2。
  - 消除的0的带方格上印刷一个X，在消除的1的带方格上印刷一个Y，在消除的2的带方格上印刷一个Z。

9.1.1 基本TM

- 正常情况下，输入带上的符号串的一般形式为

  00…0011…1122…22

- **TM**启动后，经过一段运行，输入带上的符号串的一般情况为

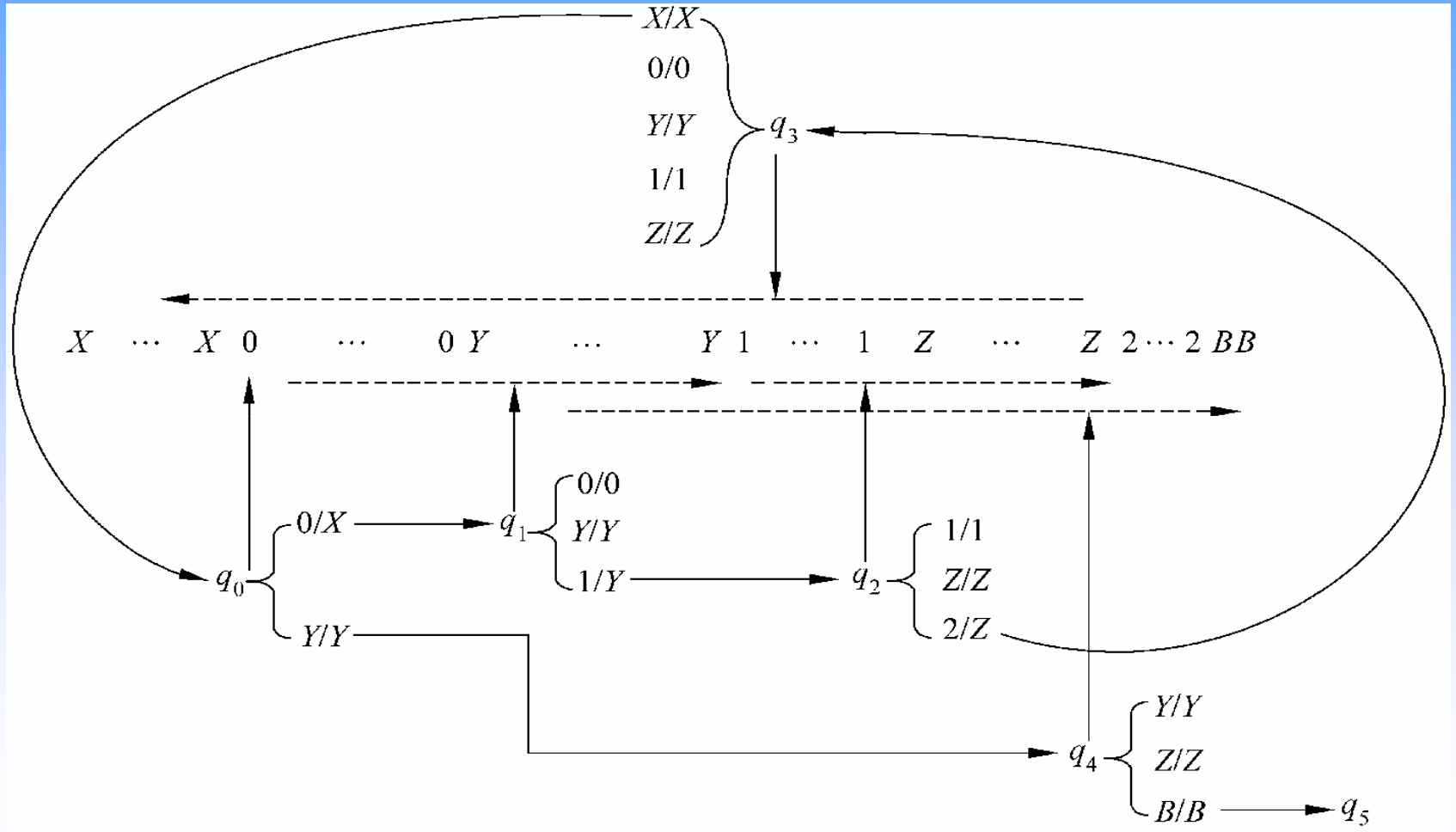
  X…X0…0Y…Y1…1Z…Z2…2BB

- 需要给予边界情况密切的关注。

9.1.1 基本TM

- 边界情况
- **X…XX…XY…YY…YZ…Z2…2BB**
- **X…XX…XY…Y1…1Z…Z2…2BB**
- **X…X0…0Y…YY…YZ…Z2…2BB**
- **X…X0…0Y…Y1…1Z…ZZ…ZBB**
- **X…X0…0Y…YY…YZ…ZZ…ZBB**

构造思路


X/X
0/0
Y/Y
1/1
Z/Z
q_3
X ⋯ X 0 ⋯ 0 Y ⋯ Y 1 ⋯ 1 Z ⋯ Z 2 ⋯ 2 BB
0/X
q_0
Y/Y
q_1
0/0
Y/Y
1/Y
q_2
1/1
Z/Z
2/Z
q_4
Y/Y
Z/Z
B/B
q_5

移动函数

	0	1	2	X	Y	Z	B
q_0	(q_0,X,R)				(q_4,Y,R)		
q_1	$(q_1,0,R)$	(q_2,Y,R)			(q_1,Y,R)		
q_2		$(q_2,1,R)$	(q_3,Z,L)			(q_2,Z,R)	
q_3	$(q_3,0,L)$	$(q_3,1,L)$		(q_0,X,R)	(q_3,Y,L)	(q_3,Z,L)	
q_4					(q_4,Y,R)	(q_4,Z,R)	(q_5,B,R)
q_5							

9.1.2 TM作为非负整函数的计算模型

- 非负整数进行编码 ——1进制
  - 用符号串0^n表示非负整数n。
- 用符号串

$$0^{n_1}10^{n_2}1\cdots10^{n_k}$$

表示k元函数$f(n_1, n_2,\ldots, n_k)$的输入。

•如果$f(n_1, n_2,\ldots, n_k)=m$，则该TM的输出为$0^m$。

9.1.2 TM作为非负整函数的计算模型

- 图灵可计算的(Turing computable)
- 设有k元函数$f(n_1, n_2,\ldots, n_k)=m$，TM $M=(Q, \Sigma, \Gamma, \delta, q_0, B, F)$接受输入串

$$0^{n_1}10^{n_2}1\cdots10^{n_k}$$

输出符号串0^m；当$f(n_1, n_2,\ldots, n_k)$无定义时，TM M没有恰当的输出给出。称TM M计算k元函数$f(n_1, n_2,\ldots, n_k)$，也称$f(n_1, n_2,\ldots, n_k)$为TM M计算的函数。也称f是图灵可计算的。

9.1.2 TM作为非负整函数的计算模型

- 完全递归函数(total recursive function)
  - 设有k元函数$f(n_1, n_2,\ldots, n_k)$，如果对于任意的$n_1, n_2,\ldots, n_k$，f均有定义，也就是计算f的TM总能给出确定的输出，则称f为完全递归函数。
- 部分递归函数(partial recursive function)
  - TM计算的函数称为部分递归函数。

9.1.2 TM作为非负整函数的计算模型

- **例 9-5** 构造TM M_4，对于任意非负整数n，m，M_4计算m+n。

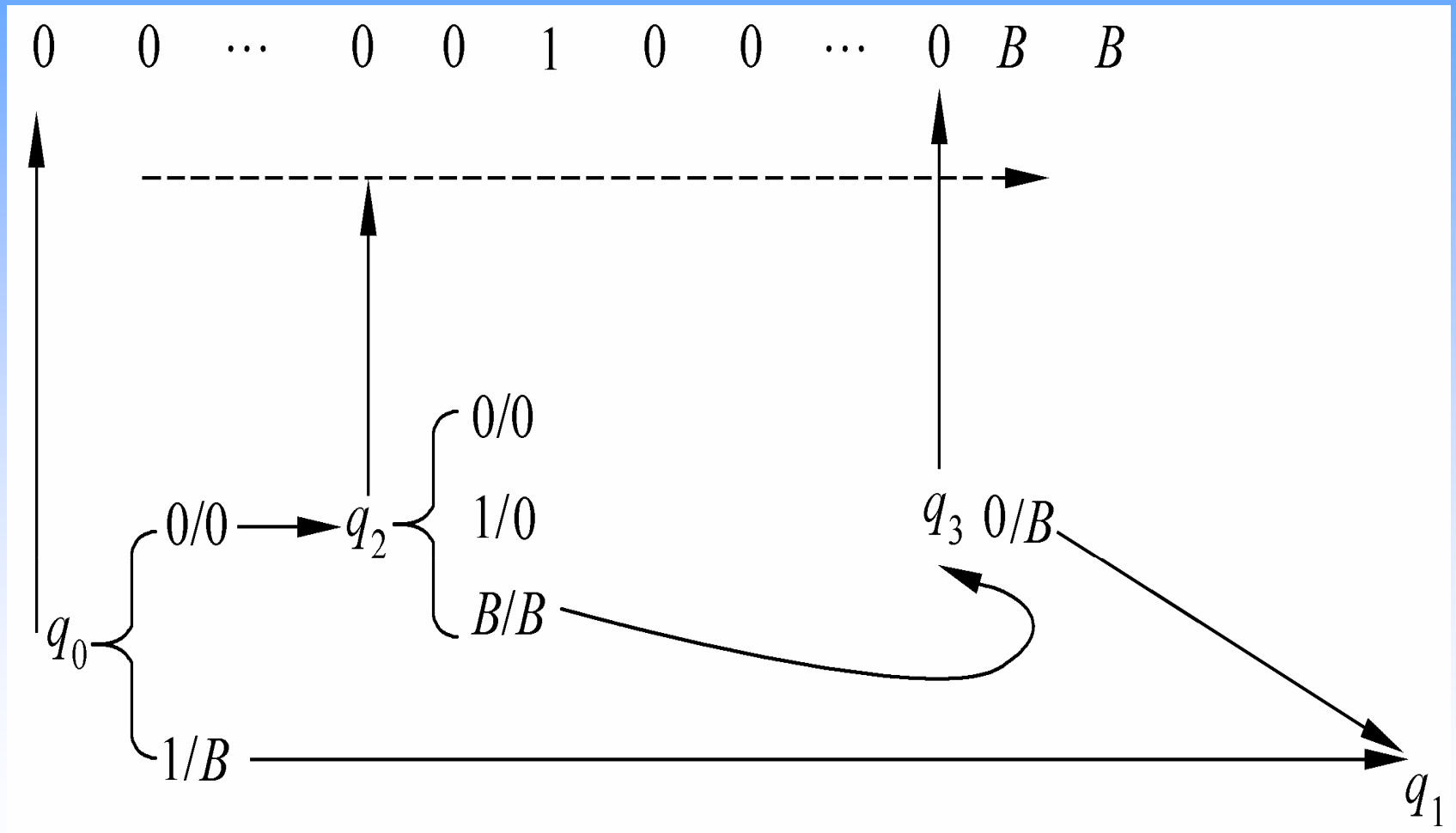
  分析：M_4的输入为0^n10^m，输出0^{n+m}的符号串。n和m为0的情况需要特殊考虑。

  （1）当n为0时，只用将1变成B就完成了计算，此时无需考察m是否为0；

  （2）当m为0时，需要扫描过表示n的符号0，并将1改为B。

  （3）当n和m都不为0时，我们需要将符号1改为0，并将最后一个0改为B。

构造思路



M_4

$M_4=(\{q_0, q_1, q_2, q_3\}, \{0,1\}, \{0,1,B\}, \delta, q_0, B, \{q_1\})$

$\delta(q_0, 1)=(q_1, B, R)$

$\delta(q_0, 0)=(q_2, 0, R)$

$\delta(q_2, 0)=(q_2, 0, R)$

$\delta(q_2, 1)=(q_2, 0, R)$

$\delta(q_2, B)=(q_3, B, L)$

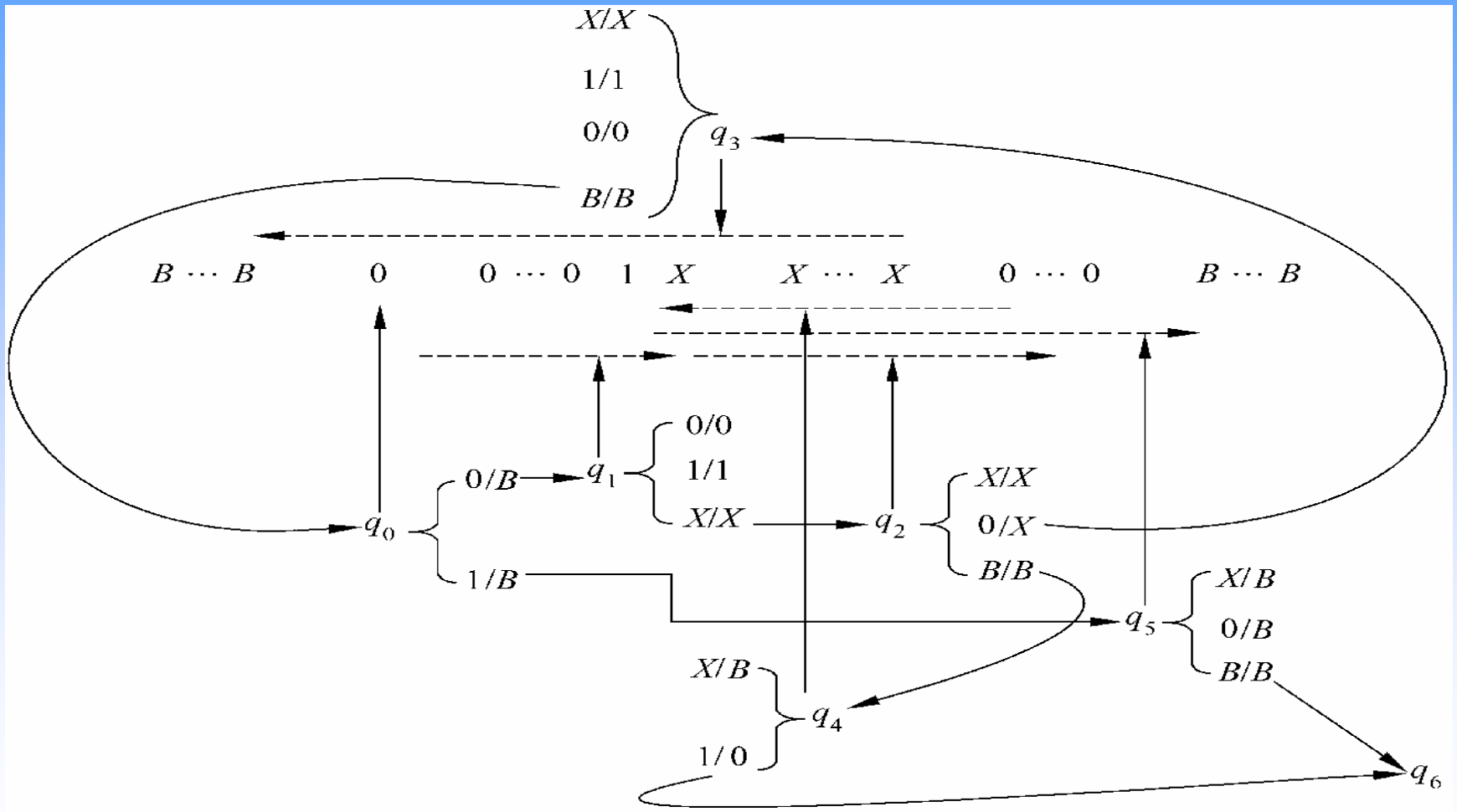
$\delta(q_3, 0)=(q_1, B, R)$

9.1.2 TM作为非负整函数的计算模型

- **例 9-6** 构造TM M_5，对于任意非负整数n，m，M_5计算如下函数：

$$n \dot{-} m = \begin{cases} n - m & n \geqslant m \\ 0 & n < m \end{cases}$$

构造思路


X/X
1/1
0/0
B/B
q3
B ··· B
0
0 ··· 0
1
X
X ··· X
0 ··· 0
B ··· B
0/0
1/1
X/X
0/B
q1
q0
1/B
X/X
0/X
B/B
q2
X/B
0/B
B/B
q5
X/B
1/0
q4
q6

M_5

- $M_5=(\{q_0, q_1, q_2, q_3, q_4, q_5, q_6,\}, \{0,1\}, \{0, 1, X, B\}, \delta, q_0, B, \{q_6\})$

$\delta(q_0, 0)=(q_1, B, R)$

$\delta(q_0, 1)=(q_5, B, R)$

$\delta(q_1, 0)=(q_1, 0, R)$

$\delta(q_1, 1)=(q_1, 1, R)$

$\delta(q_1, X)=(q_2, X, R)$

$\delta(q_2, X)=(q_2, X, R)$

$\delta(q_2, 0)=(q_3, X, L)$

$\delta(q_2, B)=(q_4, B, L)$

$\delta(q_3, X)=(q_3, X, L)$

$\delta(q_3, 1)=(q_3, 1, L)$

$\delta(q_3, 0)=(q_3, 0, L)$

$\delta(q_3, B)=(q_0, B, R)$

M_5

$\delta(q_4, X)=(q_4, B, L)$

$\delta(q_4, 1)=(q_6, 0, R)$

$\delta(q_5, X)=(q_5, B, R)$

$\delta(q_5, 0)=(q_5, B, R)$

$\delta(q_5, B)=(q_6, B, R)$。

9.1.3 TM的构造

1. 状态的有穷存储功能的利用

- 例 9-7 构造TM M_6，使得$L(M_6)=\{x \mid x\in\{0,1\}^* \& x$中至多含3个1$\}$。

  分析：M_6只用记录已经读到的1的个数。

  q[0]表示当前已经读到0个1；

  q[1]表示当前已经读到1个1；

  q[2]表示当前已经读到2个1；

  q[3]表示当前已经读到3个1。

1. 状态的有穷存储功能的利用

- M_6=({q[0], q[1], q[2], q[3], q[f]}, {0,1}, {0,1,B}, δ , q[0], B, {q[f]})

δ (q[0], 0)=(q[0], 0, R)

δ (q[0], 1)=(q[1], 1, R)

δ (q[0], B)=(q[f], B, R)

δ (q[1], 0)=(q[1], 0, R)

δ (q[1], 1)=(q[2], 1, R)

δ (q[1], B)=(q[f], B, R)

δ (q[2], 0)=(q[2], 0, R)

δ (q[2], 1)=(q[3], 1, R)

δ (q[2], B)=(q[f], B, R)

δ (q[3], 0)=(q[3], 0, R)

δ (q[3], B)=(q[f], B, R)

1. 状态的有穷存储功能的利用

- **TM是要接受且仅接受恰含3个1的0、1串的TM，对M_6进行修改，得到M_7**
- **$L(M_7)$ ={x | x∈{0,1}* & x中含且仅含3个1}**
- **M_7=({q[0], q[1], q[2], q[3], q[f]}, {0, 1}, {0, 1, B}, δ , q[0], B, {q[f]})**

$L(M_7) = \{x \mid x \in \{0,1\}^*$且 x中仅含3个1$\}$

- **δ (q[0], 0)=(q[0], 0, R)**
- **δ (q[0], 1)=(q[1], 1, R)**
- **δ (q[1], 0)=(q[1], 0, R)**
- **δ (q[1], 1)=(q[2], 1, R)**
- **δ (q[2], 0)=(q2), 0, R)**
- **δ (q[2], 1)=(q[3], 1, R)**
- **δ (q[3], 0)=(q[3], 0, R)**
- **δ (q[3], B)=(q[f], B, R)**

$L(M_8)=\{x \mid x\in\{0,1\}^*$ & x中至少含3个1}

$M_8=(\{q[0], q[1], q[2], q[f]\}, \{0, 1\}, \{0, 1, B\}, \delta, q[0], B, \{q[f]\})$

$\delta(q[0], 0)=(q[0], 0, R)$

$\delta(q[0], 1)=(q[1], 1, R)$

$\delta(q[1], 0)=(q[1], 0, R)$

$\delta(q[1], 1)=(q[2], 1, R)$

$\delta(q[2], 0)=(q[2], 0, R)$

$\delta(q[2], 1)=(q[f], 1, R)$

1. 状态的有穷存储功能的利用

- **例9-8** 构造TM M_9它的输入字母表为{0，1}，现在要求M_9在它的输入符号串的尾部添加子串101。

  分析：
  - 将待添加子串101存入穷控制器。
  - 首先找到符号串的尾部。
  - 将给定符号串中的符号依次地印刷在输入带上
  - 每印刷一个符号，就将它从有穷控制器的“存储器”中删去，当该“存储器”空时，TM就完成了工作。

1. 状态的有穷存储功能的利用

M_9=({q[101], q[01], q[1], q[ε]}, {0, 1}, {0, 1, B}, δ , q[101], B, {q[ε]})

其中 δ 的定义为：

δ (q[101], 0)=(q[101], 0, R)

δ (q[101], 1)=(q[101], 1, R)

δ (q[101], B)=(q[01], 1, R)

δ (q[01], B)=(q[1], 0, R)

δ (q[1], B)=(q[ε], 1, R)

1. 状态的有穷存储功能的利用

- **例9-9** 构造TM M_{10}它的输入字母表为{0，1}，要求M_{10}在它的输入符号串的开始处添加子串101。
- 将有穷控制器中的“存储器”分成两部分
  - 第一部分用来存放待添加的子串。
  - 第二部分用来存储因添加符号串当前需要移动的输入带上暂时无带方格存放的子串。
  - 一般形式为**q[x, y]**
    - **x**待添加子串
    - **y**当前需要移动的输入带上暂时无带方格存放的子串。

1.状态的有穷存储功能的利用

- **q[x, ε]为开始状态；**
- **q[ε，ε]为终止状态。**
- **设a、b为输入符号**
- **δ(q[ax, y], b) = (q[x, yb], a, R)**
  - 表示在没有完成待插入子串的印刷之前，要将待插入子串的首字符印刷在**TM**当前扫描的带方格上。

1. 状态的有穷存储功能的利用

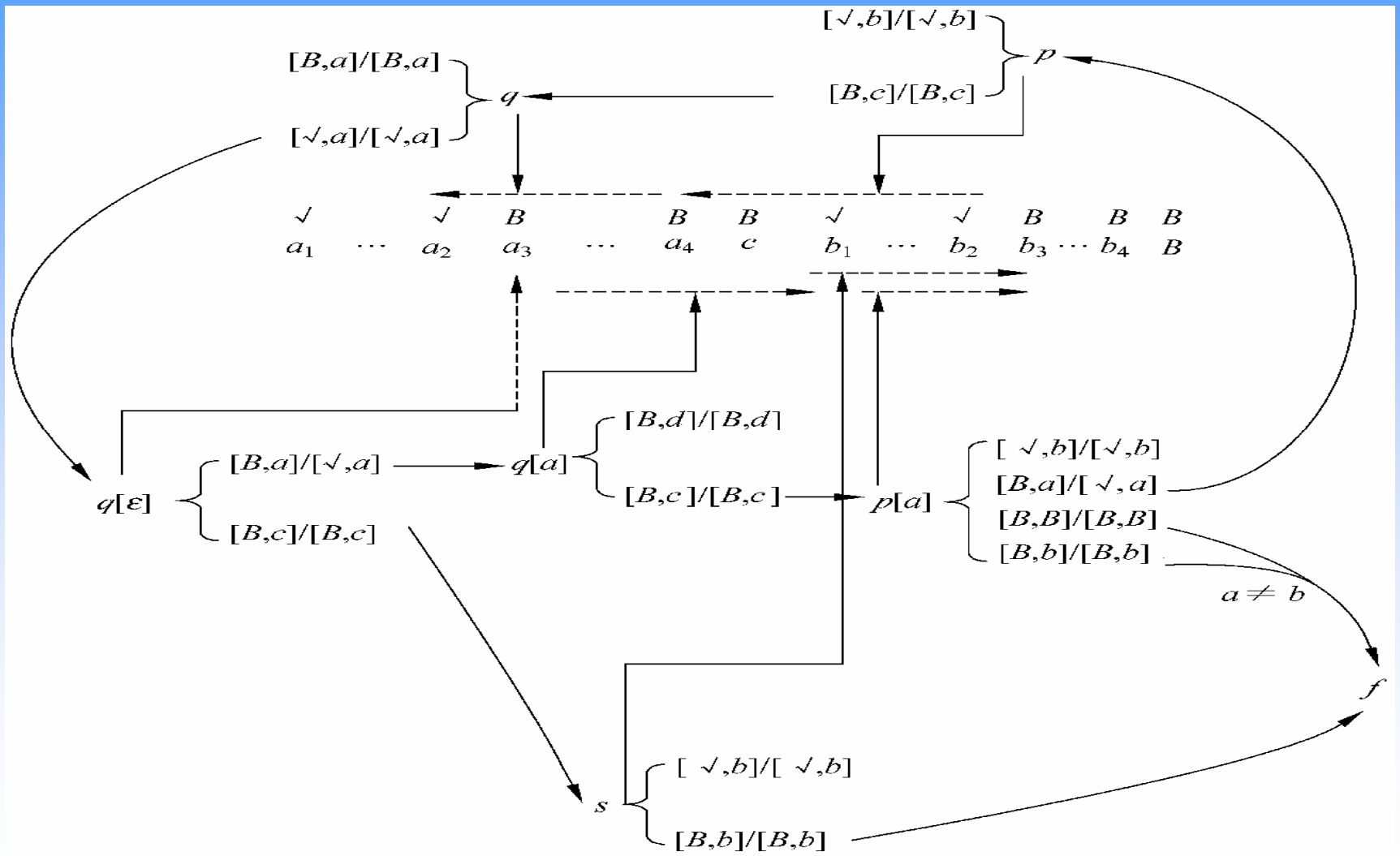
- **δ (q[ε ,a y], b) = (q[ε , yb], a, R)**
  - 表示当完成待插入子串的插入工作之后，必须将插入点之后的子串顺序地向后移动。
- **δ (q[ε ,a y], B) = (q[ε , y], a, R)**
  - 表示读头当前所指的带方格为空白，现将“存储器”的第二部分中的当前首符号**a**印刷在此带方格上，同时将这个符号从存储器中删除。

2. 多道(multi-track)技术

- **例9-10 构造M_{11}，使$L(M_{11})=\{xcy \mid x,y\in\{0,1\}^+$ 且 $x\neq y\}$。**

  分析：
  - 以符号c为分界线，逐个地将c前的符号与c后的符号进行比较。
  - 当发现对应符号不同时，就进入终止状态。
  - 当发现x与y的长度不相同而进入终止状态。
  - 发现它们相同而停机。
  - 一个道存放被检查的符号串，另一个存放标记符。

构造思路


[√,b]/[√,b]
[B,a]/[B,a]
p
q
[B,c]/[B,c]
[√,a]/[√,a]
√ √ B B B √ √ B B B
a1 ⋯ a2 a3 ⋯ a4 c b1 ⋯ b2 b3 ⋯ b4 B
[B,d]/[B,d]
[B,a]/[√,a]
q[a]
q[ε]
[B,c]/[B,c]
[B,c]/[B,c]
p[a]
[√,b]/[√,b]
[B,a]/[√,a]
[B,B]/[B,B]
[B,b]/[B,b]
a ≠ b
f
s
[√,b]/[√,b]
[B,b]/[B,b]

2. 多道(multi-track)技术

- **M_{11}=({q[ε], q[0], q[1], p[0], p[1] , q, p, s, f}, {[B,0], [B,1], [B,c]}, {[B,0], [B,1], [B,c], [√,0], [√,1], [B,B]}, δ , q[ε], [B,B], {f})**

δ (q[ε], [B,0])=(q[0], [√,0], R)

δ (q[ε], [B,1])=(q[1], [√,1], R)

δ (q[a], [B,d])=(q[a], [B,d], R)

δ (q[a], [B,c])=(p[a], [B,c], R)

δ (p[a], [√,b])=(p[a], [√,b], R)

δ (p[a], [B,a])=(p, [√,a], L)

2. 多道(multi-track)技术

- δ (p, [√,b])=(p, [√,b], L)
- δ (p, [B,c])=(q, [B,c], L)
- δ (q, [B,a])=(q, [B,a], L)
- δ (q, [√,a])=(q[ε], [√,a], R)
- δ (p[a], [B,b])=(f, [B,b],R)
- δ (p[a], [B,B])=(p, [B,B], R)
- δ (s, [√,b])=(s, [√,b], R)
- δ (s, [B,a])=(f, [B,a], R)

3. 子程序(subroutine)技术

- 将TM的设计看成是一种特殊的程序设计，将子程序的概念引进来。
- 一个完成某一个给定功能的TM M′从一个状态q开始，到达某一个固定的状态f结束。
- 将这两个状态作为另一个TM M的两个一般的状态。
- 当M进入状态q时，相当于启动M′(调用M′对应的子程序)；当M′进入状态f时，相当于返回到M的状态f。

3. 子程序(subroutine)技术

- **例9-11** 构造M_{12}完成正整数的乘法运算。

  分析:
  - 设两个正整数分别为**m**和**n**。
  - 输入串为0^n10^m 。
  - 输出应该为0^{n*m} 。
  - 算法思想：每次将**n**个**0**中的**1**个**0**改成**B**，就在输入串的后面复写**m**个**0**。
  - 在M_{12}的运行过程中，输入带的内容为

    $$B^h0^{n-h}10^m10^{m*h}B$$

正整数的乘法运算

（1）初始化。完成将第一个0变成B，并在最后一个0后写上1。我们用q_0表示启动状态，用q_1表示完成初始化后的状态。首先，消除前n个0中的第一个0，

$$q_0 0^n 10^m \vdash^+ Bq_1 0^{n-1} 10^m 1$$

（2）主控系统。从状态q_1开始，扫描过前n个0中剩余的0和第一个1，将读头指向m个0的第一个，此时的状态为q_2。其ID变化为

$$B^h q_1 0^{n-h} 10^m 10^{m*(h-1)} B \vdash^+ B^h 0^{n-h} 1\ q_2 0^m 10^{m*(h-1)} B$$

正整数的乘法运算

- 当子程序完成**m**个**0**的复写后，回到q_3。这个状态相当于子程序的返回（终止）状态。然后在q_3状态下，将读头移回到前**n**个**0**中剩余的**0**中的第一个**0**，并将这个**0**改成**B**，进入q_1状态，准备进行下一次循环

$$B^h0^{n-h}1\ q_30^m10^{m*h}B \vdash^+ B^{h+1}q_10^{n-h-1}10^m10^{m*h}B$$

正整数的乘法运算

- 当完成m*n个0的复写之后，清除输入带上除了这m*n个0以外的其他非空白符号。q_4为终止状态

$$B^n q_1 10^m 10^{m*n} B \vdash^+ B^{n+1+m+1} q_4 0^{m*n} B$$

（3）子程序。完成将m个0复写到后面的任务。从q_2启动，到q_3结束，返回到主控程序。

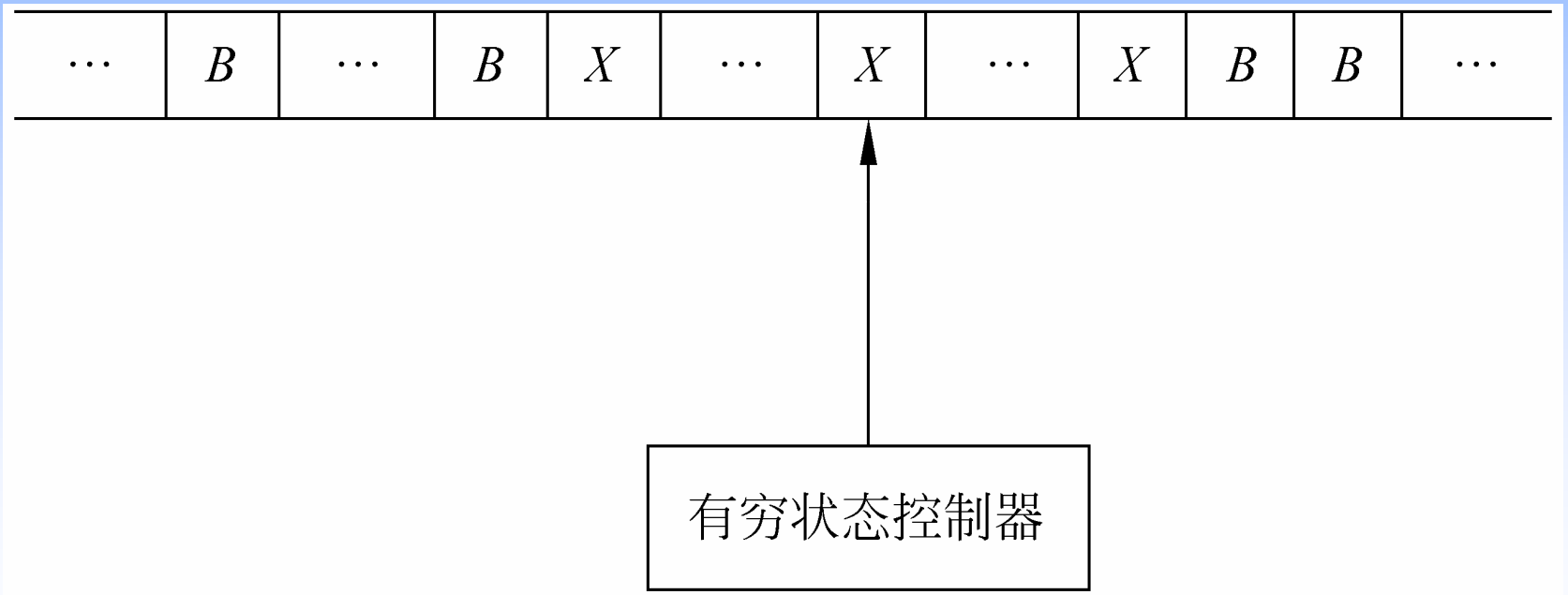
$$B^{h+1} 0^{n-h-1} 1\ q_2 0^m 10^{m*h} B \vdash^+ B^{h+1} 0^{n-h-1} 1\ q_3 0^m 10^{m*h+1} B$$

9.2 TM的变形

- 从不同的方面对TM进行扩充。
  - 双向无穷带TM。
  - 多带TM。
  - 不确定的TM。
  - 多维TM等。
- 它们与基本的TM等价。

9.2.1 双向无穷带TM

物理模型



9.2.1 双向无穷带TM

- 双向无穷带 (Turing machine with two-way infinite tape，TM)

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

- Q, Σ, Γ, δ ,q_0 , B , F的意义同定义9-1。
- M的即时描述ID同定义 9 -2。
- 允许M的读头处在输入串的最左端时，仍然可以向左移动。

9.2.1 双向无穷带TM

- M的当前ID $X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n$
- 如果 $\delta(q, X_i)=(p, Y, R)$
  - 当$i\neq 1$并且$Y\neq B$时，M的下一个ID为

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

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

  - 表示M在ID $X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n$下，经过一次移动，将ID变成$X_1X_2\cdots X_{i-1}YpX_{i+1}\cdots X_n$。

9.2.1 双向无穷带TM

- 当i=1并且Y=B时，M的下一个ID为

  $pX_2 \ldots X_n$

- 记作

  $qX_1X_2 \ldots X_n \vdash_M pX_2 \ldots X_n$

- 这就是说，和基本TM在读头右边全部是B时，这些B不在ID中出现一样，当双向无穷带TM的读头左边全部是B时，这些B也不在该TM的ID中出现。

9.2.1 双向无穷带TM

- 如果 $\delta(q, X_i)=(p, Y, L)$
  - 当$i \neq 1$时，M的下一个ID为

    $X_1X_2\cdots pX_{i-1}YX_{i+1}\cdots X_n$
  - 记作

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

  - 表示M在ID $X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots_n$下，经过一次移动，将ID变成$X_1X_2\cdots pX_{i-1}YX_{i+1}\cdots X_n$。

9.2.1 双向无穷带TM

- 当i=1时，M的下一个ID为

  $pBYX_2\ldots X_n$

- 记作

  $qX_1X_2\ldots X_n \vdash_M pBYX_2\ldots X_n$

- 表示M在ID $qX_1X_2\ldots X_n$下，经过一次移动，将ID变成$pBYX_2\ldots X_n$。

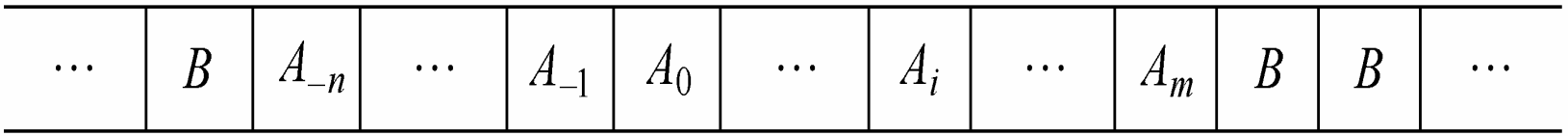
9.2.1 双向无穷带TM

定理9-1 对于任意一个双向无穷带TMM，存在一个等价的基本TMM′。

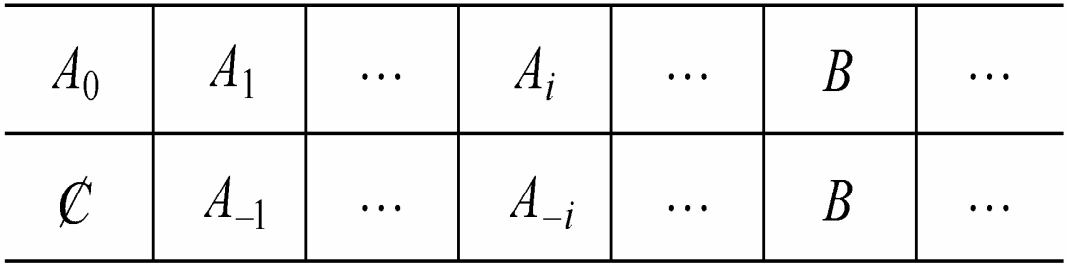
证明要点：

- 双向无穷存储的模拟：用一个具有2个道的基本TM来模拟：一个道存放M开始启动时读头所注视的带方格及其右边的所有带方格中存放的内容；另一个道按照相反的顺序存放开始启动时读头所注视的带方格左边的所有带方格中存放的内容。
- 双向移动的模拟：在第1道上，移动的方向与原来的移动方向一致，在第2道上，移动的方向与原来的移动方向相反。

用单向无穷带模拟双向无穷带



(a) M的双向无穷带



(b) M'用单向无穷带模拟 M的双向无穷带

9.2.2 多带TM

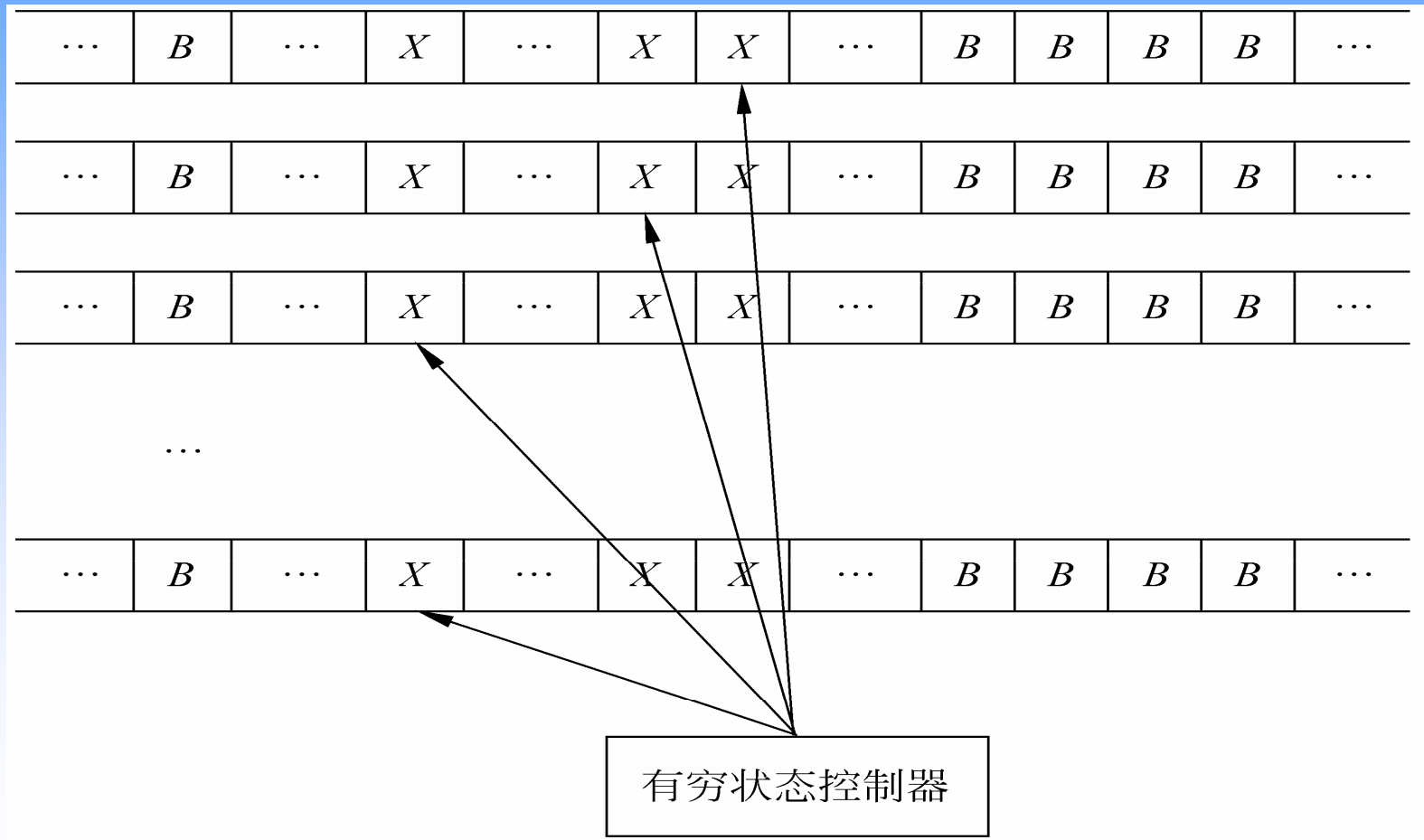
- **多带TM(multi-tape turing machine)**
- 允许TM有多个双向无穷带，每个带上有一个相互独立的读头。
- **k**带**TM**在一次移动中完成如下三个动作

  ⑴ 改变当前状态；

  ⑵ 各个读头在自己所注视的带方格上印刷一个希望的符号。

  ⑶ 各个读头向各自希望的方向移动一个带方格。

9.2.2 多带TM



9.2.2 多带TM

定理 9-2 多带TM与基本的TM等价。

证明要点：

- 对一个k带TM，用一条具有2k道的双向无穷带TMM′，实现对这个k带TMM的模拟。
- 对应M的每一条带，M′用两个道来实现模拟。其中一条道用来存放对应的带的内容，另一条道专门用来标记对应带上的读头所在的位置。

9.2.3 不确定的TM

- 不确定TM与基本TM的区别是对于任意的$(q，X)\in Q\times\Gamma$，

  $\delta(q，X)=\{(q_1，Y_1，D_1)，(q_2，Y_2，D_2)，…，(q_k，Y_k，D_k)\}$
- D_j为读头的移动方向。即$D_j\in\{R，L\}$。
- 表示M在状态q，读到X时，可以有选择地进入状态q_j，印刷字符Y_j，按D_j移动读头
- $L(M)=\{w\mid w\in\Sigma^*$且$ID_1\vdash^*ID_n$，且$ID_n$含M的终止状态}。

9.2.3 不确定的TM

定理 9-3 不确定的TM与基本的TM等价

- 证明要点：
  - 让等价的基本TMM′ 具有3条带。
  - 第1条带用来存放输入。
  - 第2条带上系统地生成M的各种可能的移动序列
  - M′在第3条带上按照第2条带上给出的移动系列处理输入串，如果成功，则接受之，如果不成功，则在第2条带上生成下一个可能的移动系列，开始新一轮的“试处理”。

9.2.4 多维TM

- **多维TM(multi-dimensional Turing machine)**
  - 读头可以沿着多个维移动。
- **k维TM(k-dimensional Turing machine)**
- **TM可以沿着k维移动。**
- **k维TM的带由k维阵列组成，而且在所有的2k个方向上都是无穷的，它的读头可以向着2k个方向中的任一个移动。**

9.2.4 多维TM

定理 9-4 多维TM与基本TM等价。

- 用一维的形式表示k维的内容，就像多维数组在计算机的内存中都被按照一维的形式实现存储一样。
- **段(Segment)**用来表是一维上的内容。
- 用#作为段分割符。
- ¢用作该字符串的开始标志，$用作该字符串的结束标志。

基本TM模拟2维TM

B	a_1	a_2	a_3	a_4	B	B	B	B	B	B
B	a_5	B	a_6	a_7	a_8	a_9	a_{10}	B	B	B
B	a_{11}	B	B	B	B	a_{12}	B	a_{13}	B	a_{14}
a_{15}	a_{16}	B	B	B	B	B	B	B	B	a_{16}
B	B	B	a_{17}	B	B	B	B	B	a_{18}	B
a_{19}	a_{20}	B	B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B	B	B	a_{21}

基本TM模拟2维TM

¢ $Ba_1a_2a_3a_4BBBBBBB \# Ba_5Ba_6a_7a_8a_9a_{10}BBB \#$

$Ba_{11}BBBBa_{12}Ba_{13}Ba_{14} \# a_{15}\ a_{16}BBBBBBBBB\ a_{16}\#$

$BBB\ a_{17}BBBBBa_{18}B \# a_{19}a_{20}BBBBBBBBB \#$

$BBBBBBBBBBa_{21}\$$

9.2.5 其他TM

1. 多头TM
2. 离线TM
3. 作为枚举器的TM
4. 多栈机
5. 计数机
6. Church-Turing论题与随机存取机

1. 多头TM

- **多头TM(multi-head Turing machine)**
- 指在一条带上有多个读头，它们受**M**的有穷控制器的统一控制，**M**根据当前的状态和这多个头当前读到的字符确定要执行的移动。在**M**的每个动作中，各个读头所印刷的字符和所移动的方向都可以是相互独立的**m**

1. 多头TM

定理 9-5 多头TM与基本的TM等价。

- 可以用一条具有k+1个道的基本TM来模拟一个具有k个头的TM（k头TM）。其中一个道用来存放原输入带上的内容，其余k个道分别用来作为k个读头位置的标示。

2. 离线TM

- **离线TM(off-line Turing machine)**
  - 有一条输入带是只读带**(read-only tape)**的多带**TM**。
- 符号¢和**$**用来限定它的输入串存放区域，¢在左边，**$**在右边。
- 不允许该带上的读头移出由¢和**$**限定的区域——离线的**TM**。
- 如果只允许只读带上的读头从左向右移动，则称之为**在线TM(on-line Turing machine)**。

2. 离线TM

定理 9-6 离线TM与基本的TM等价。

证明要点：让模拟M的离线TM比M多一条带，并且用这多出来的带复制M的输入串。然后将这条带看作是M的输入带，模拟M进行相应的处理。

3. 作为枚举器的TM

- **作为枚举器的TM(Turing machine as enumerator)**
  - 多带TM，其中有一条带专门作为输出带，用来记录产生语言的每一个句子。
- 在枚举器中，一旦一个字符被写在了输出带上，它就不能被更改。如果该带上的读头的正常移动方向是向右移动的话，这个带上的读头是不允许向左移动的。
- 如果这个语言有无穷多个句子，则它将永不停机。它每产生一个句子，就在其后打印一个分割符“#”。
- 枚举器产生的语言记为G(M)。

3. 作为枚举器的TM

- **规范的顺序(canonical order)**

定理 9-7 L为递归可枚举语言的充分必要条件是存在一个TM M，使得L=G(M)。

定理 9-8 一个语言L为递归语言的充分必要条件是存在一个TMM，使得L=G(M)，并且L是被M按照规范顺序产生的。

4. 多栈机

- **多栈机(multi-stack machines)**是一个拥有一条只读输入带和多条存储带的不确定**TM**。
  - 多栈机的只读带上的读头不能左移。
  - 存储带上的读头可以向左和向右移动。
    - 右移时，一般都在当前注视的带方格上印刷一个非空白字符
    - 左移时，必须在当前注视的带方格中印刷空白字符**B**。
- 一个确定的**双栈机(double stack machines)**是一个确定的**TM**，它具有一条只读的输入带和两条存储带。存储带上的读头左移时，只能印刷空白符号**B** 。

4. 多栈机

- 下推自动机是一种非确定的多带TM。它有一条只读的输入带，一条存储带。

定理 9-9 一个任意的单带TM可以被一个确定的双栈机模拟。

5. 计数机

- **计数机(counter machine)**
  - 有一条只读输入带和若干个用于计数的单向无穷带的离线**TM**。
  - 拥有**n**个用于计数带的计数机被称为**n**计数机。
  - 用于计数的带上仅有两种字符，一个为相当于是作为栈底符号的**Z**，该字符也可以看作是计数带的首符号，它仅出现在计数带的最左端；另一个就是空白符**B**。这个带上所记的数就是从**Z**开始到读头当前位置所含的**B**的个数。

定理 9-10 TM可以被一个双计数机模拟。

6.丘奇-图灵论题与随机存取机

- 对于任何可以用有效算法解决的问题，都存在解决此问题的**TM**。
- **随机存取机(random access machine，RAM)**含有无穷多个存储单元，这些存储单元被按照**0、1、2、…**进行编号，每个存储单元可以存放一个任意的整数；有穷个能够保存任意整数的算术寄存器。这些整数可以被译码成通常的各类计算机指令。

定理 9-11如果**RAM**的基本指令都能用**TM**来实现，那么就可以用**TM**实现**RAM**。

9.3 通用TM

- **通用TM(universal Turing machine)**
  - 实现对所有TM的模拟。
- 编码系统
  - 它可以在实现对TM的表示的同时，实现对该TM处理的句子的表示。
  - 用0和1对这些除空白符以外的其他的带符号进行编码。同时也可以用0和1对TM的移动函数进行编码。

9.3 通用TM

- **$M=(\{q_1, q_2, \ldots, q_n\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$**
- 用X_1、X_2、X_3分别表示0、1、B，用D_1、D_2分别表示R、L。
- $\delta(q_i, X_j)=(q_k, X_l, D_m)$可以用$0^i10^j10^k10^l10^m$表示。

9.3 通用TM

- **M**可用

  111 code$_1$ 11 code$_2$ 11 …… 11 code$_r$ 111
- **code$_t$** 是动作 **$\delta(q_i, X_j)=(q_k, X_l, D_m)$**的形如 **$0^i10^j10^k10^l10^m$**的编码。
- **TMM**和它的输入串**w**则可以表示成

  111 code$_1$ 11 code$_2$ 11 …… 11 code$_r$ 111w
- 按照规范顺序分别对表示**TM**的符号行和表示输入的符号行进行排序。

9.3 通用TM

- **L_d={w | w是第j个句子，并且第j个TM不接受它}不是递归可枚举语言。**
- **通用语言(universal language)**
  - **L_u={<M，w> | M接受w}**
  - **<M，w>为如下形式的串，表示TMM=({q_1, q_2, …, q_n}, {0，1}, {0，1，B}, δ ,q_1 , B , {q_2})和它的输入串w。**

111 $code_1$ 11 $code_2$ 11 … 11 $code_r$ 111w

9.3 通用TM

- 例 9-12 设TM $M_2=(\{q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, B\}, \delta, q_4, B, \{q_3\})$，其中 δ 的定义如下:

$\delta(q_4, 0)= (q_4, 0, R)$

$\delta(q_4, 1)= (q_1, 1, R)$

$\delta(q_1, 0)= (q_1, 0, R)$

$\delta(q_1, 1)= (q_2, 1, R)$

$\delta(q_2, 0)= (q_2, 0, R)$

$\delta(q_2, 1)= (q_3, 1, R)$

9.3 通用TM

- 编码为

  111000010100001010110000100101001011010101011010010010011001010010010111001001000100101111

- 通用TM检查M是否接受字符串**001101110**

  111000010100001010110000100101001011010101011010010010011001010010010111001001000100101111001101110

9.4 几个相关的概念

- **可计算性(computability)**理论是研究计算的一般性质的数学理论。计算的过程就是执行算法的过程。
- 可计算理论的中心问题是建立计算的数学模型，研究哪些是可计算的，哪些是不可计算的。
- 可计算理论又称为**算法理论(algorithm theory)**。
- 在直观意义下，算法具有有限性、机械可执行性、确定性、终止性等特征。
- 可计算问题可以等同于图灵可计算问题。

9.4 几个相关的概念

- **可判定的(decidable)问题**
  - 它对应的语言是递归的。
- **不可判定的(undecidable)**
  - 没有这样的算法，它以问题的实例为输入，并能给出相应的“是”与“否”的判定。
- 递归语言举例

（1）L_{DFA}={<M, w> | M是一个DFA，w是字符串，M接受w}。

（2）L_{NFA}={<M, w> | M是一个NFA，w是字符串，M接受w}。

9.4 几个相关的概念

（3）L_{RE}={<r, w> | r是一个RE，w是字符串，w是r的一个句子}。

（4）E_{DFA}={<M> | M是一个DFA，且 $L(M)=\Phi$}。

（5）EQ_{DFA}={<M_1, M_2> | M_1, M_2是DFA，且 $L(M_1)=L(M_2)$}。

（6）L_{CFG}={<G, w> | G是一个CFG，w是字符串，G产生w}。

（7）E_{CFG}={<G> | G是一个CFG，且$L(G)=\Phi$}。

9.4 几个相关的概念

- **P类问题(class of P)**
  - **P**表示确定的**TM**在多项式时间(步数)内可判定的语言类。这些语言对应的问题成为是**P类问题**，这种语言称为多项式可判定的。
  - 例如，判定一个有向图中的两个顶点之间是否存在有向路的的问题、检查两个数是否互素的问题、判定一个字符串是否为一个上下文无关语言的句子的问题都是**P**类问题。

9.4 几个相关的概念

- **NP类问题(class of NP)**
- **NP**表示不确定的**TM**在多项式时间(步数)内可判定的语言类。这些语言对应的问题称为是**NP类问题**，也称这些问题是**NP**复杂的，或者**NP**困难的。
- 这种语言称为非确定性多项式可判定的。
- **P=NP**？是理论计算机科学和当代数学中最大的悬而未决的问题之一。

9.4 几个相关的概念

- **NP完全的(NP complete problem)**
  - **NP**类中有某些问题的复杂性与整个类的复杂性相关联。如果能找到这些问题中的任何一个的多项式时间判定算法，那么，所有的**NP**问题都是多项式时间可以判定的。
  - **TSP(**旅行商问题**)**。
  - 划分问题。
  - 可满足性问题。
  - 带有先次序的调度问题。

9.5 小结

TM是一个计算模型，用TM可以完成的计算被称为是图灵可计算的。

(1) TM的基本概念：形式定义、递归可枚举语言、递归语言、完全递归函数、部分递归函数。

(2) 构造技术：状态的有穷存储功能的利用、多道技术、子程序技术。

9.5 小结

(3) TM的变形：双向无穷带TM、多带TM、不确定的TM、多维TM、多头TM、离线TM、多栈TM，它们都与基本TM等价。

(4) Church-Turing论题：如果RAM的基本指令都能用TM来实现，则RAM就可以用TM实现。所以，对于任何可以用有效算法解决的问题，都存在解决此问题的TM。

(5) 通用TM可以实现对所有TM的模拟。

(6) 可计算语言、不可判定性、P-NP问题。

第10章 上下文有关语言

- 主要内容
  - **TM与PSG的等价性。**
  - 线性界限自动机(LBA)。
  - LBA作为CSL的识别器。
- 重点
  - LBA、 LBA作为CSL的识别器。
- 难点
  - LBA作为CSL的识别器。
- 本章的内容是介绍性。

10.1 TM与PSG的等价性

- 例**10-1** 构造产生语言{0^n | n为2的非负整数次幂}的文法。

  设计思想：

  - 在文法中设置变量C，充当TM中的读头的作用，它从左到右扫描0，并且在每次遇到一个0时，都用00替换之，这使得当它从最左端移到最右端时，就完成了当前串的加倍工作，为了使串中的0再次被加倍，变量D充当将这个“读头”从右端移回到最左端的作用。为了标记出端点，文法用A、B分别表示串的最左端和最右端。

10.1 TM与PSG的等价性

- G_1：S→0

  产生句子0。

- S→ACOB

  产生句型ACOB，A、B分别表示左右端点，C为向右的倍增“扫描器”。

- CO→OOC

  C向右扫描，将每一个0变成OO，以实现0个数的加倍。

10.1 TM与PSG的等价性

- CB→DB

  C到达句型的左端点，变成D，准备进行从右到左的扫描，以实现对句型中0的个数的再次加倍。

- CB→E

  C到达句型的左端点，变成E，表示加倍工作已完成，准备结束。

10.1 TM与PSG的等价性

- OD→DO　D移回到左端点。
- AD→AC

  当D到达左端点时，变成C，此时已经做好了进行下一次加倍的准备工作。
- OE→EO　E向右移动，以寻找左端点A。
- AE→ ε

  E找到A后，一同变成 ε ，从而得到一个句子。

10.1 TM与PSG的等价性

另一个相关的文法

G_2：S→ACOB

CO→OOC

CB→DB

OD→DOO

CB→E

AD→AC

AC→F

FO→OF

OE→EO

AE→ ε

FB→ ε

10.1 TM与PSG的等价性

定理10-1 对于任一**PSG** G=(V，T，P，S)，存在TM M，使得L(M)=L(G)。

证明要点：

基本思想如下。

- M具有两条带，其中一条带用来存放输入字符串w，第二条带用来试着产生w。即，第二条带上存放的将是一个句型。我们希望该句型能够派生出w。在开始启动时，这个句型就是S。

10.1 TM与PSG的等价性

- 设第二条带上的句型为γ，M按照某种策略在γ中选择为G的某个产生式左部的子串α，再按照非确定的方式选择α产生式的某一个候选式β，用β替换α。在需要时，利用适当的移动技术，让TM可以实现将句型中的α替换成β的工作。
- 当第二条带上的内容为一个终极符号行时，就把它与第一条带上的w进行比较，如果相等，就接受；如果不相等，就去寻找是否存在可以产生w的派生。

10.1 TM与PSG的等价性

- 由于G为PSG，所以，在整个“试派生”过程中，我们是无法总能根据当前句型的长度来决定该派生是否需要继续进行下去。这样一来，对于一个给定的输入字符串，如果它不是L(G)的句子，我们构造的TM可能会陷入用不停机的工作过程中。这从另一方面说明，短语结构语言不一定是递归语言。

10.1 TM与PSG的等价性

定理10-2 对于任一TM M，存在**PSG**G=(V，T，P，S)，使得L(G)=L(M)。

证明要点：

①设TM M=(Q，Σ，Γ，δ，q_0，B，F)，L=(M)。

②让G可以产生Σ*中的任意一个字符串的变形，然后让G模拟M处理这个字符串。如果M接受它，则G就将此字符串的变形还原成该字符串。

③变形是让每个字符对应一个二元组。∀[a_1，a_1][a_2，a_2]...[a_n，a_n] ∈(Σ×Σ)*，被看成$a_1a_2...a_n$的两个副本。

10.1 TM与PSG的等价性

④G在一个副本上模拟M的识别动作，如果M进入终止状态，则G将句型中除另一个副本外的所有字符消去。

$G=((\Sigma \cup \{\varepsilon\}) \times \Gamma \cup \{A_1, A_2, A_3\} \cup Q, \Sigma, P, A_1)$

(1) $A_1 \rightarrow q_0 A_2$ 准备模拟M从q_0启动；

(2) $A_2 \rightarrow [a, a]A_2$

$\forall a \in \Sigma$，A_2首先生成任意的形如$[a_1, a_1][a_2, a_2]\ldots[a_n, a_n]$的串；

10.1 TM与PSG的等价性

- (3) $A_2 \rightarrow A_3$

  在预生成双副本子串$[a_1, a_1][a_2, a_2]\ldots\ldots[a_n, a_n]$后，准备用$A_3$在该子串之后生成一系列的相当于空白符的子串，为G能够顺利地模拟M在处理相应的输入字符串的过程中，需要将读头移向输入串右侧的初始为B的地方做准备；

10.1 TM与PSG的等价性

- **(4)** $A_3 \rightarrow [\varepsilon, B]\, A_3$

  由于**M**在处理一个字符时，不知道将需要用到输入串右侧的多少个初始为**B**的带方格，所以，我们让A_3生成一系列的相当于空白符的子串$[\varepsilon, B]\,[\varepsilon, B] \ldots\ldots [\varepsilon, B]$。在派生过程中，其个数依据实际需要而定；

10.1 TM与PSG的等价性

- **(5)** $A_3 \rightarrow \varepsilon$
- **(6)** $\forall a \in \Sigma \cup \{\varepsilon\}$，$\forall q, p \in Q$，$\forall X, Y \in \Gamma$，如果 $\delta(q, X)=(p, Y, R)$，则
  - $q[a, X] \rightarrow [a, Y]p$
  - G模拟M的一次右移；

10.1 TM与PSG的等价性

- **(7)　对于∀a, b∈∑∪{ ε }，∀q, p∈Q，∀X, Y, Z∈Γ，如果δ(q, X)=(p, Y, L)，则**
  - **[b, Z]q[a, X] → p[b, Z] [a, Y]**
  - **G模拟M的一次左移；**
- **(8)　对于∀a∈∑∪{ ε }，∀q∈F则**
  - **[a, X]q→qaq　G先将句型中的[、]、X等消除；**
  - **q[a, X] →qaq**
  - **q→ ε　　最后再消除句型中的状态q**

10.2 LBA及其与CSG的等价性

- **线性有界自动机(linear bounded automaton，LBA)**
  - 非确定的TM。
  - 输入字母表包含两个特殊的符号¢和$，其中，¢作为输入符号串的左端标志，$作为输入符号串的右端标志。
  - LBA的读头只能在¢和$之间移动，它不能在端点符号¢和$上面打印另外一个符号。

10.2 LBA及其与CSG的等价性

- LBA可以被看成一个八元组，

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

其中，Q、Σ、Γ、δ、q_0、F与TM中的定义相同，$¢ \in \Sigma$，$\$ \in \Sigma$，M接受的语言

$L(M)=\{w \mid w \in (\Sigma-\{¢, \$\})^* \ \& \ \exists q \in F$使得$q_0 ¢ w \$ \vdash^* ¢ \alpha q \beta \$$。

10.2 LBA及其与CSG的等价性

定理10-3 如果**L**的**CSL**，$\varepsilon \notin L$，则存在**LBA M**，使得**L=L(M)**。

证明要点：

① 设 **CSG G=(V，T，P，S)**，使得**L=L(G)**。。

②用一个两道**TM**模拟**G**。一道存放字符串$\not\subset$**w$**，另一道用来生成**w**的推导。

LBA及其与CSG的等价性

③ CSG保证只用考察长度不超过|w|句型。

④ 将句型的长度限制在|w|以内，所以，M的运行不会超出符号¢和$规定的范围。

⑤ 对于任意输入，LBA均会停机，这表明CSL是递归语言。

10.2 LBA及其与CSG的等价性

定理10-4 对于任意L，$\varepsilon \notin L$，存在LBA M，使得L=L(M)。则L是CSL。

证明：与定理10-2的证明类似，主要是根据给定的LBA M构造出CSG G。这里的双副本串是形如$[a_1, q_0 \not\subset a_1][a_2, a_2]\ldots[a_n, a_n\$]$的符号行，当长度为1时，此符号行为$[a, q_0 \not\subset a\$]$。

10.2 LBA及其与CSG的等价性

（1）对于$\forall a \in \Sigma - \{¢, \$\}$，

$A_1 \rightarrow [a, q_0 ¢ a] A_2$

准备模拟M从q_0启动，生成形如$[a_1, q_0 ¢ a_1][a_2, a_2]\ldots[a_n, a_n\$]$的双副本串(句型)中的$[a_1, q_0 ¢ a_1]$，并将生成子串$[a_2, a_2]\ldots[a_n, a_n\$]$的任务交给$A_2$；

$A_1 \rightarrow [a, q_0 ¢ a\$]$

生成双副本串$[a, q_0 ¢ a\$]$；

10.2 LBA及其与CSG的等价性

（2）对于$\forall a \in \Sigma - \{¢, \$\}$，

$A_2 \rightarrow [a, a]A_2$

A_2首先生成任意的形如$[a_1, q_0 ¢ a_1][a_2, a_2]\ldots[a_n, a_n\$]$的双副本串中的子串$[a_2, a_2]\ldots[a_{n-1}, a_{n-1}]$;

（3）对于$\forall a \in \Sigma - \{¢, \$\}$，

$A_2 \rightarrow [a, a\$]$

A_2最后生成任意的形如$[a_1, q_0 ¢ a_1][a_2, a_2]\ldots[a_n, a_n\$]$的双副本中的子串$[a_n, a_n\$]$;

10.2 LBA及其与CSG的等价性

(4) 对于$\forall a \in \Sigma - \{\$\}$，$\forall q, p \in Q$，$\forall X, Y, Z \in \Gamma$，$X \neq \$$，如果$\delta(q, X) = (p, Y, R)$，则

- $[a, q\ X][b, Z] \rightarrow [a, Y]\ [b, p\ Z]$
- G模拟M的一次右移；

(5) 对于$\forall a, b \in \Sigma - \{\not\subset\}$，$\forall q, p \in Q$，$\forall X, Y, Z \in \Gamma$，如果$\delta(q, X) = (p, Y, L)$，则

- $[b, Z]\ [a, q\ X] \rightarrow [b, p\ Z]\ [a, Y]$
- G模拟M的一次左移；

10.2 LBA及其与CSG的等价性

(6) 对于$\forall a \in \Sigma$，$\forall q \in F$，$\forall X, Y \in \Gamma - \{B\}$，
$[a, XqY] \rightarrow a$

- 由于**q**为终止状态，所以可以消除它

(7) 对于$\forall a \in \Sigma - \{¢, \$\}$，$\forall X \in \Gamma - \{B\}$，

- **$[a, X]b \rightarrow ab$**
- **$a\,[b, X] \rightarrow ab$**

10.3 小结

本章讨论TM与PSG的等价性，介绍了识别CSL的装置——LBA。

(1) 对于任一PSG G=(V，T，P，S)，存在TM M，使得L(M)=L(G)；

(2) 对于任一TM M，存在PSG G=(V，T，P，S)，使得L(G) =L(M)；

(3) LBA是一种非确定的TM，它的输入串被用符号¢和$括起来，而且读头只能在¢和$之间移动；

10.2 LBA及其与CSG的等价性

(4)如果L是CSL，ε∉L，则存在LBA M，使得L=L(M)；

(5)对于任意L，ε∉L，存在LBA M，使得L=L(M)，则L是CSL。