

# 第三章

## 形式语言与自动机 及其在NLP中的应用

---

# 目录

---

3.1 基本概念

3.2 形式语言

3.3 自动机

3.4 有限自动机的应用

# 3.1 基本概念

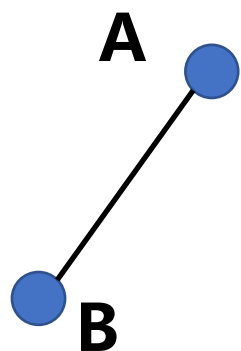
# 3.1 基本概念

## ◆ 树 (tree):

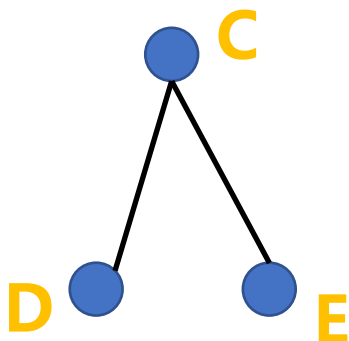
- 树是一种数据结构。
- 一个连通的无回路的无向图称为树(或称自由树)。
- 如果树中有一个结点被特别地标记，则这棵树被称之为**根树**，这个被特别标记的结点被称之为**根结点**。

# 3.1 基本概念

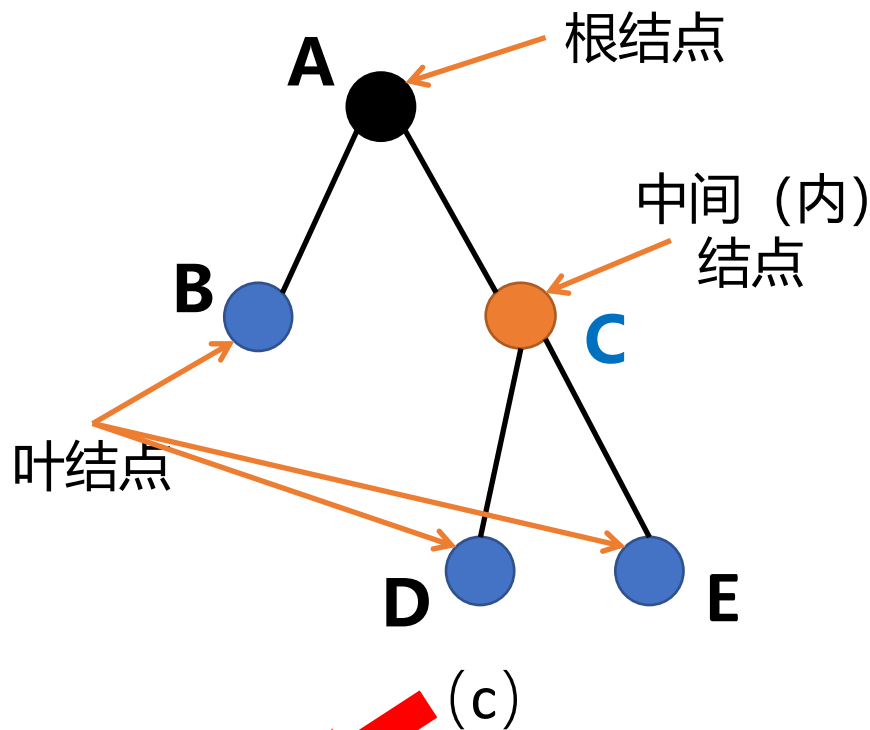
## ◆ 树 (tree):



(a)



(b)



**父结点:** A 是 B、C 结点的父结点; C 是 D、E 的父结点。  
**子结点:** B、C 是 A 的子结点; D、E 是 C 的子结点。  
**兄弟结点:** B、C 互为兄弟结点; D、E 互为兄弟结点。

# 3.1 基本概念

## ◆ 字符串 (string):

- **定义:** 假定  $\Sigma$  是字符的有限集合, 由  $\Sigma$  中字符相连而成的有限序列被称之为  $\Sigma$  上的字符串(或称符号串, 或称链)。特殊地, 不包括任何字符的字符串称为空串, 记作  $\varepsilon$ 。
- **符号串的长度:** 符号串中符号的个数。符号串  $x$  的长度用  $|x|$  表示。  $|\varepsilon| = 0$ 。  
包括空串的  $\Sigma$  上字符串的全体记为  $\Sigma^*$ 。

# 3.1 基本概念

## ◆ 字符串操作:

假定  $\Sigma$  是字符的有限集合,  $x, y$  是  $\Sigma$  上的符号串

### 1. 字符串连接:

则把  $y$  的各个符号写在  $x$  的符号之后得到的符号串称为  $x$  与  $y$  的连接, 记作  $xy$ 。

例如:

$$\Sigma = \{a, b, c\}, x = ab, y = cba$$

$$\text{那么, } xy = abcba$$

# 3.1 基本概念

## ◆ 字符串操作:

### 1. 字符串连接(方幂):

设  $x$  是符号串, 把  $x$  自身连接  $n$  次得到的符号串, 即  $z = xx \dots x$  ( $n$  个  $x$ ), 称为  $x$  的  $n$  次方幂, 记作  $x^n$ 。

注意:  $x^0 = \varepsilon$

$$x^n = xx^{n-1} = x^{n-1}x (n \geq 1)$$

$$x^* = x^n (n \geq 0), x^+ = x^n (n \geq 1)$$

例如: 如果  $x = a$ , 则  $x^1 = a$ ,  $x^2 = aa$ ,  $x^3 = aaa$ ,

如果  $x = ab$ , 则  $x^0 = \varepsilon$ ,  $x^3 = ababab$



# 3.1 基本概念

## ◆ 字符串操作:

### 2. 符号串集合的乘积:

设  $A, B$  是符号串的集合, 则  $A, B$  的乘积定义为:

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

相应的,  $A^0 = \{\varepsilon\}$ ,  $A^n = A^{n-1}A = AA^{n-1}$

例如: 设  $A = \{aa, bb\}$ ,  $B = \{cc, dd, ee\}$ , 则

$$AB = \{aacc, aadd, aae, bbcc, bbdd, bbee\}$$

$$A^2 = \{aaaa, aabb, bbaa, bbbb\}$$

# 3.1 基本概念

## ◆ 字符串操作:

### 3. 闭包:

如果  $V$  是字符表  $\Sigma$  上的字符串集合, 那么  $V$  的闭包  
定义为:  $V^* = V^0 \cup V^1 \cup V^2 \cup \dots$

$$V^+ = V^1 \cup V^2 \cup \dots \quad (\text{称为 } V \text{ 的正闭包})$$

$$V^+ = V^* - \{\varepsilon\}$$

例如:  $V = \{a, b\}$

$$V^* = \{\varepsilon, a, b, aa, ab, bb, ba, aaa, \dots\}$$

$$V^+ = \{a, b, aa, ab, ba, bb, aaa, \dots\}$$

# 3.1 基本概念

## ◆ 正则表达式(regular experssion):

正则表达式简称正则式，对应于  $\Sigma$  上的一些子集(正则集)，并通过递归定义：

- ① 空集  $\phi$  和空字符串  $\varepsilon$  是正则式，它们的正则集分别为  $\phi$  和  $\{\varepsilon\}$ ;
- ② 任何  $x \in \Sigma$ ,  $x$  是正则式，它对应的正则集是  $\{x\}$ ;
- ③ 如果  $X, Y$  是  $\Sigma$  上的正则式，并且它们对应的正则集分别为  $U, V$ ，那么， $X|Y$ ,  $X \cdot Y$  和  $X^*$  也是正则式，且它们对应的正则集分别为  $U \cup V$ ,  $U \cdot V$  和  $U^*$ 。

# 3.1 基本概念

## ◆ 正则表达式(regular experssion):

例如: 假设  $\Sigma = \{0, 1\}$ , 那么 0 和 1 都是正则表达式。

如果令  $x = 0, y = 1$ , 那么

$y^* = 1^*$  也是正则式, 对应的正则集为:

$$U = \{\varepsilon, 1, 11, \dots\}$$

$xy^* = 01^*$  也是正则式, 且它对应的正则集:

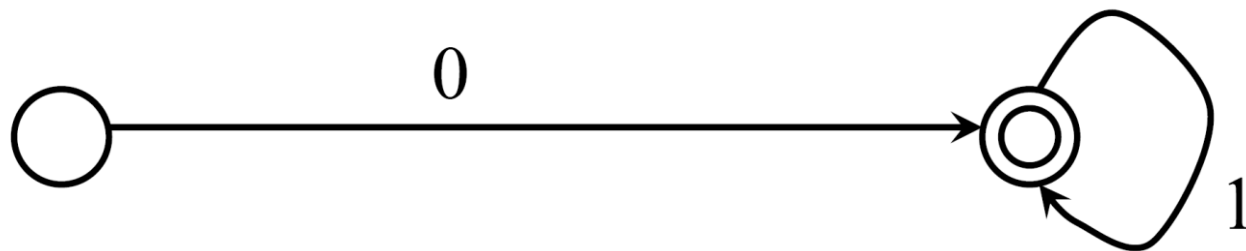
$$V = \{0, 01, 011, 0111, \dots\}$$

$$x|y^* = \{x\} \cup U = \{0, \varepsilon, 1, 11, 111, \dots\}$$

# 3.1 基本概念

## ◆ 正则表达式与有限状态图：

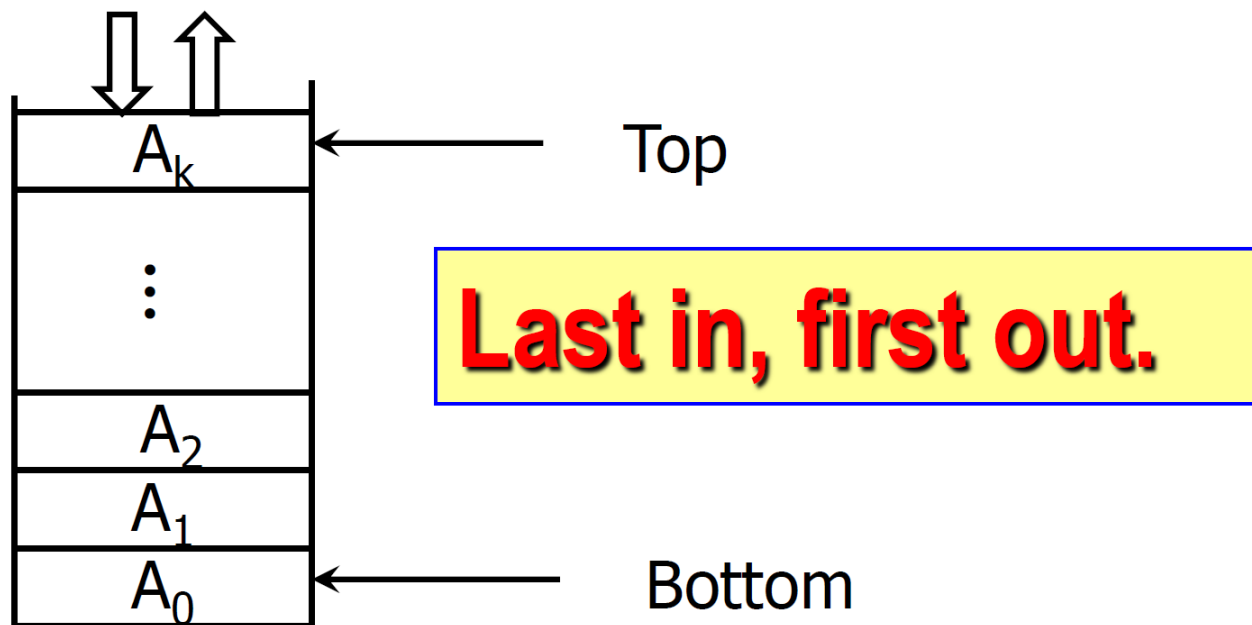
正则表达式可以用有向图表示，图的结点是状态，有一个起始结点和一个终止结点。起始结点只有出边，终止结点用双圆圈表示。边上的符号表示从一个状态到另一个状态结点允许出现的字符，这种图称之为有限状态图。正则式  $01^*$  对应的有限状态图为：



# 3.1 基本概念

## ◆ 栈(stack):

栈是一种线性表,  $A = A_0, A_1 \cdots A_k$ , 其中  $A_0$  是栈底,  $A_k$  是栈顶。当栈为空时,  $A_0$  既是栈顶也是栈底。



## 3.2 形式语言

## 3.2 形式语言

### ◆ 关于语言的定义：

人类所特有的用来表达意思、交流思想的工具，是一种特殊的社会现象，由语音、词汇和语法构成一定的系统。

——商务印书馆，《现代汉语词典》，1996

语言可以被看成一个抽象的数学系统。

——吴蔚天，1994

按照一定规律构成的句子和符号串的有限或无限的集合。

——Chomsky



## 3.2 形式语言

### ◆ 语言描述的三种途径：

- **穷举法**：只适合句子数目有限的语言。
- **语法描述**：生成语言中合格的句子。
- **自动机**：对输入的句子进行检验，区别哪些是语言中的句子，哪些不是语言中的句子。

# 上次课回顾

## 2.3 应用实例

- 基于上下文分类的消歧方法
- 基于最大熵的消歧方法

## 3.1 基本概念

—树、字符串及操作、正则表达式、栈

## 3.2 形式语言

—语言描述途径：穷举法、语法描述、自动机

## 3.2 形式语言

### ◆ 形式语言的直观意义：

形式语言是用来精确地描述语言（包括人工语言和自然语言）及其结构的手段。形式语言学也称代数语言学。

以重写规则  $\alpha \rightarrow \beta$  的形式表示，其中， $\alpha, \beta$  均为字符串。  
顾名思义：字符串  $\alpha$  可以被改写成  $\beta$ 。一个初步的字符串通过不断地运用重写规则，就可以得到另一个字符串。通过选择不同的规则并以不同的顺序来运用这些规则，就可以得到不同的新字符串。

## 3.2 形式语言

### ◆形式语法的定义：

形式语法是一个4元组  $G = (N, \Sigma, P, S)$ , 其中  $N$  是非终结符的有限集合(有时也叫变量集或句法种类集);  $\Sigma$  是终结符的有限集合,  $N \cap \Sigma = \Phi$ ;  $V = N \cup \Sigma$  称为总词汇表;  $P$  是一组重写规则的有限集合:  $P = \{\alpha \rightarrow \beta\}$ , 其中,  $\alpha, \beta$  是  $V$  中元素构成的串, 但  $\alpha$  中至少应含有一个非终结符号;  $S \in N$ , 称为句子符或初始符。

例如:  $G = (\{A, S\}, \{0, 1\}, P, S)$

$$P: S \rightarrow 0 A 1 \quad 0 A \rightarrow 00A1 \quad A \rightarrow 1$$

## 3.2 形式语言

### ◆ 推导的定义:

设  $G = (N, \Sigma, P, S)$  是一个文法, 在  $(N \cup \Sigma)^*$  上定义关系  $\Rightarrow_G$  (直接派生或推导)如下:

如果  $\alpha\beta\gamma$  是  $(N \cup \Sigma)^*$  中的符号串, 且  $\beta \rightarrow \delta$  是  $P$  的产生式, 那么  $\alpha\beta\gamma \Rightarrow_G \alpha\delta\gamma$ 。

## 3.2 形式语言

### ◆ 推导的定义：

用  $\xRightarrow{+}_G$  (按非平凡方式派生) 表示  $\Rightarrow_G$  的传递闭包, 也就是  $(N \cup \Sigma)^*$  上的符号串  $\xi_i$  到  $\xi_{i+1}$  的  $n (n \geq 1)$  步推导或派生。

用  $\xRightarrow{*}_G$  (派生) 表示  $\Rightarrow_G$  的自反和传递闭包, 即由  $(N \cup \Sigma)^*$  上的符号串  $\xi_i$  到  $\xi_{i+1}$  经过  $n (n \geq 0)$  步的推导或派生。

如果清楚某个推导是文法  $G$  所产生的, 则符号  $\xRightarrow{+}_G$  或  $\xRightarrow{*}_G$  中的  $G$  可以省略不写。

## 3.2 形式语言

### ◆ 最左推导、最右推导和规范推导：

约定每步推导中只改写最左边的那个非终结符，这种推导称为“**最左推导**”。

约定每步推导中只改写最右边的那个非终结符，这种推导称为“**最右推导**”。

最右推导也称**规范推导**。

## 3.2 形式语言

### 例3-1:

给定  $G = (\{E, T, F\}, \{a, +, *, (, )\}, P, E)$

$$P: E \rightarrow E + T \mid T \quad T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid a$$

字符串  $a + a * a$  的两种推导过程:

$$E \Rightarrow E + T$$




## 3.2 形式语言


### 例3-1:

给定  $G = (\{E, T, F\}, \{a, +, *, (, )\}, P, E)$

$$P: E \rightarrow E + T \mid T \quad T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid a$$

字符串  $a + a * a$  的两种推导过程:

$$E \Rightarrow E + T \Rightarrow T + T$$


## 3.2 形式语言


### 例3-1:

给定  $G = (\{E, T, F\}, \{a, +, *, (, )\}, P, E)$

$$P: E \rightarrow E + T | T \quad T \rightarrow T * F | F$$

$$F \rightarrow (E) | a$$

字符串  $a + a * a$  的两种推导过程:

$$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T$$


## 3.2 形式语言


### 例3-1:

给定  $G = (\{E, T, F\}, \{a, +, *, (, )\}, P, E)$

$$P: E \rightarrow E + T \mid T \quad T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid a$$

字符串  $a + a * a$  的两种推导过程:

$$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow a + T$$


## 3.2 形式语言


### 例3-1:

给定  $G = (\{E, T, F\}, \{a, +, *, (, )\}, P, E)$

$$P: E \rightarrow E + T | T \quad T \rightarrow T * F | F$$

$$F \rightarrow (E) | a$$

字符串  $a + a * a$  的两种推导过程:

$$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow a + T \Rightarrow a + T * F$$


## 3.2 形式语言


### 例3-1:

给定  $G = (\{E, T, F\}, \{a, +, *, (, )\}, P, E)$

$$P: E \rightarrow E + T \mid T \quad T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid a$$

字符串  $a + a * a$  的两种推导过程:

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


## 3.2 形式语言


### 例3-1:

给定  $G = (\{E, T, F\}, \{a, +, *, (, )\}, P, E)$

$$P: E \rightarrow E + T \mid T \quad T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid a$$

字符串  $a + a * a$  的两种推导过程:

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


## 3.2 形式语言

### 例3-1:


给定  $G = (\{E, T, F\}, \{a, +, *, (, )\}, P, E)$

$$P: E \rightarrow E + T | T \quad T \rightarrow T * F | F$$

$$F \rightarrow (E) | a$$

字符串  $a + a * a$  的两种推导过程:

$$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow a + T \Rightarrow a + T * F$$

$$\Rightarrow a + F * F \Rightarrow a + a * F \Rightarrow a + a * a \quad \text{(最左推导)}$$


## 3.2 形式语言

### 例3-1:

给定  $G = (\{E, T, F\}, \{a, +, *, (, )\}, P, E)$

$$P: E \rightarrow E + T \mid T \quad T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid a$$

字符串  $a + a * a$  的两种推导过程:

$$\begin{aligned} E &\Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow a + T \Rightarrow a + T * F \\ &\Rightarrow a + F * F \Rightarrow a + a * F \Rightarrow a + a * a \end{aligned} \quad \textbf{(最左推导)}$$

$$\begin{aligned} E &\Rightarrow E + T \Rightarrow E + T * F \Rightarrow E + T * a \Rightarrow E + F * a \\ &\Rightarrow E + a * a \Rightarrow T + a * a \Rightarrow F + a * a \Rightarrow a + a * a \end{aligned} \quad \textbf{(最右推导)}$$



## 3.2 形式语言

### ◆ 句型与句子:

一些特殊类型的符号串为文法  $G = (N, \Sigma, P, S)$  的句子形式(句型):

- (1)  $S$  是一个句子形式;
- (2) 如果  $\alpha\beta\gamma$  是一个句子形式, 且  $\beta \rightarrow \delta$  是  $P$  的产生式, 则  $\alpha\delta\gamma$  也是一个句子形式;

文法  $G$  的不含非终结符的句子形式称为  $G$  生成的句子。由文法  $G$  生成的语言, 记作  $L(G)$ , 指  $G$  生成的所有句子的集合。即:  $L(G) = \{x \mid x \in \Sigma, S \xrightarrow[G]{*} x\}$

## 3.2 形式语言

### ◆ 句型与句子:

在乔姆斯基的语法理论中，文法被划分为4种类型：

- 3型文法——正则文法
- 2型文法——上下文无关文法
- 1型文法——上下文有关文法
- 0型文法——无约束文法

## 3.2 形式语言

### ◆ 正则文法:

如果文法  $G = (N, \Sigma, P, S)$  的规则集  $P$  中所有的规则满足如下形式:  $A \rightarrow Bx$ , 或  $A \rightarrow x$ , 其中  $A, B \in N$ ,  $x \in \Sigma$ , 则称该文法为正则文法或称3型文法。(左线性正则文法)

如果一正则文法中所有含非终结符号的规则形式为  $A \rightarrow xB$ , 则该文法称为右线性正则文法。

## 3.2 形式语言

### 例3-2:

$$G = (N, \Sigma, P, S)$$

$$N = \{S, A, B\}, \quad \Sigma = \{a, b\},$$

$$P: (a) S \rightarrow aA \quad (b) A \rightarrow aA$$

$$(c) A \rightarrow bbB \quad (d) B \rightarrow bB$$

$$(e) B \rightarrow b$$

$$\begin{array}{l} A \rightarrow bB' \\ B' \rightarrow bB' \end{array}$$

$$L(G) = \{a^n b^m\}, n \geq 1, m \geq 3$$

## 3.2 形式语言

### ◆ 上下文无关文法: (context-free grammar, CFG)

如果文法  $G$  的规则集  $P$  中所有规则均满足如下形式:

$$A \rightarrow \alpha, \text{ 其中 } A \in N, \alpha \in (N \cup \Sigma)^*$$

则称该文法为上下文无关文法(CFG)或称 2型文法

## 3.2 形式语言

例3-3:

$$G = (N, \Sigma, P, S)$$

$$N = \{S, A, B, C\}, \quad \Sigma = \{a, b, c\},$$

$$P: \text{(a) } S \rightarrow ABC \quad \text{(b) } A \rightarrow aA|a$$

$$\text{(c) } B \rightarrow bB|b \quad \text{(d) } C \rightarrow BA|c$$

$$L(G) = \{a^n b^m a^k c^\alpha\}, n \geq 1, m \geq 1, k \geq 0, \alpha \in \{0, 1\}$$

如果  $k > 0$  的话,  $\alpha = 0$ , 否则,  $\alpha = 1$

## 3.2 形式语言

### ◆ 上下文有关文法： (context-sensitive grammar, CSG)

如果文法  $G$  的规则集合  $P$  中所有规则满足如下形式：  
 $\alpha A \beta \rightarrow \alpha \gamma \beta$ , 其中  $A \in N$ ,  $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$ , 且  $\gamma$  至少包含一个字符, 则称该文法为上下文有关文法(CSG)或称1型文法。

**另一种定义:** if  $x \rightarrow y, x \in (N \cup \Sigma)^+, y \in (N \cup \Sigma)^*$   
并且  $|y| \geq |x|$ 。

## 3.2 形式语言

例3-4:

$$G = (N, \Sigma, P, S)$$

$$N = \{S, A, B, C\}, \quad \Sigma = \{a, b, c\},$$

$$P: \text{(a) } S \rightarrow ABC \quad \text{(b) } A \rightarrow aA|a$$

$$\text{(c) } B \rightarrow bB|b \quad \text{(d) } B\textcolor{red}{C} \rightarrow B\textcolor{red}{c}c$$

$$L(G) = \{\textcolor{red}{a}^n \textcolor{red}{b}^m \textcolor{red}{c}^2\}, n \geq 1, m \geq 1$$



## 3.2 形式语言

### ◆ 无约束文法(无限制重写系统):

如果文法  $G$  的规则集  $P$  中所有规则满足如下形式:

$\alpha \rightarrow \beta$ ,  $\alpha, \beta$  是字符串, 则称  $G$  为无约束文法或无限制重写系统, 或称0型文法。

## 3.2 形式语言

注意：

每一个正则文法都是上下文无关文法，每一个上下文无关文法都是上下文有关文法，而每一个上下文有关文法都是0型文法，即：

$$L(G3) \subseteq L(G2) \subseteq L(G1) \subseteq L(G0)$$

## 3.2 形式语言

### ◆ 语言与文法类型的约定：

如果一种语言能由几种文法所产生，则把这种语言称为在这几种文法中受限制最多的那种文法所产生的语言。

### 例3-5：

$$G = (\{S, A, B\}, \{a, b\}, P, S)$$

$$P: S \rightarrow aB \quad S \rightarrow bA \quad A \rightarrow aS \quad A \rightarrow bAA$$

$$A \rightarrow a \quad B \rightarrow bS \quad B \rightarrow ABB \quad B \rightarrow b$$

$G$  为上下文无关文法。

$$L(G) = \{\text{等数量的 } a \text{ 和 } b \text{ 构成的链}\}$$

## 3.2 形式语言

### ◆ CFG产生的语言句子的派生树表示:

一个上下文无关文法  $G = (N, \Sigma, P, S)$  产生句子的过程可由派生树表示, 其构造步骤如下:

- (1) 对于  $\forall x \in N \cup \Sigma$  给一个标记作为节点,  $S$  作为树的根节点。
- (2) 如果一个节点的标记为  $A$ , 并且它至少有一个除它自身以外的后裔, 则  $A \in N$ 。
- (3) 如果一个节点的标记为  $A$ , 它的  $k(k > 0)$  个直接后裔节点按从左到右的次序依次标记为  $A_1, A_2 \cdots A_k$ , 则  $A \rightarrow A_1 A_2 \cdots A_k$  一定是  $P$  中的一个产生式。

## 3.2 形式语言

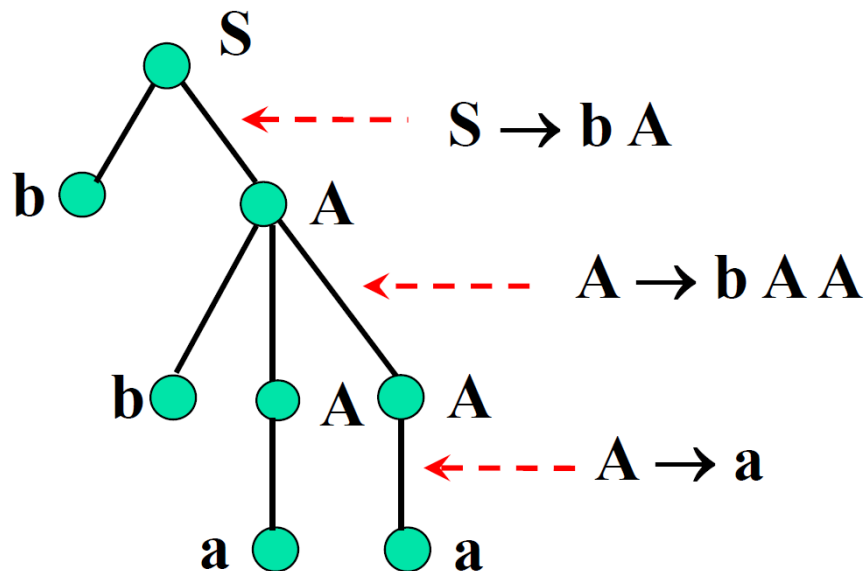
### ◆ CFG产生的语言句子的派生树表示:

例如:  $G = (\{S, A\}, \{a, b\}, P, S)$

$P: S \rightarrow bA \quad A \rightarrow bAA \quad A \rightarrow a$

$G$  所产生的句子  $bbaa$  可以由下面的生树表示:

$S \Rightarrow bA$   
 $\Rightarrow bbAA$   
 $\Rightarrow bbaA$   
 $\Rightarrow bbaa$



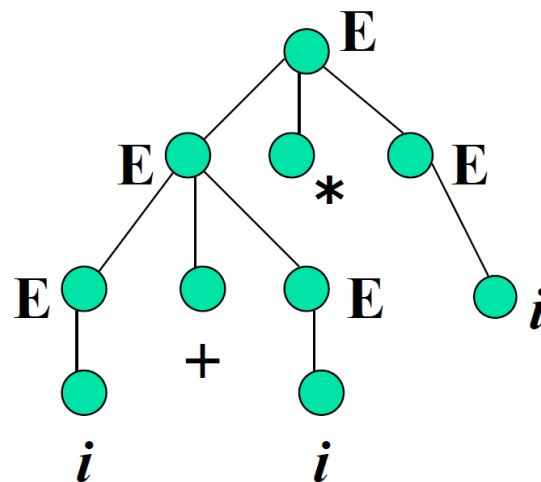
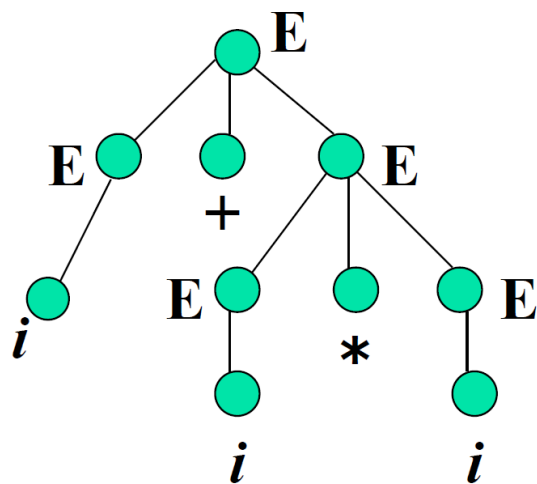
## 3.2 形式语言

### ◆ 上下文无关文法的二义性：

一个文法  $G$ ，如果存在某个句子有不只一棵分析树与之对应，那么称这个文法是二义的。

例如： $G(E): E \rightarrow E + E | E * E | (E) | E - E | i$

对于句子  $i + i * i$  有两棵对应的分析树。



## 3.2 形式语言

例:

给定文法  $G(S)$ :

①  $S \rightarrow P\ NP|PP\ Aux\ NP$  ②  $PP \rightarrow P\ NP$

③  $NP \rightarrow NN|NP\ Aux\ NP$  ④  $P \rightarrow \text{关于}$

⑤  $NN \rightarrow \text{鲁迅}| \text{文章}$  ⑥  $Aux \rightarrow \text{的}$

短语 “关于鲁迅的文章” 的推导。

$$\begin{aligned} S &\Rightarrow PP\ Aux\ NP \Rightarrow P\ NP\ Aux\ NP \Rightarrow \text{关于}NP\ Aux\ NP \\ &\Rightarrow \text{关于}NN\ Aux\ NP \Rightarrow \text{关于鲁迅} Aux\ NP \\ &\Rightarrow \text{关于鲁迅的} NP \Rightarrow \text{关于鲁迅的} NN \\ &\Rightarrow \text{关于鲁迅的文章} \end{aligned}$$

## 3.2 形式语言

例:

给定文法  $G(S)$ :

①  $S \rightarrow P\ NP|PP\ Aux\ NP$     ②  $PP \rightarrow P\ NP$

③  $NP \rightarrow NN|NP\ Aux\ NP$     ④  $P \rightarrow \text{关于}$

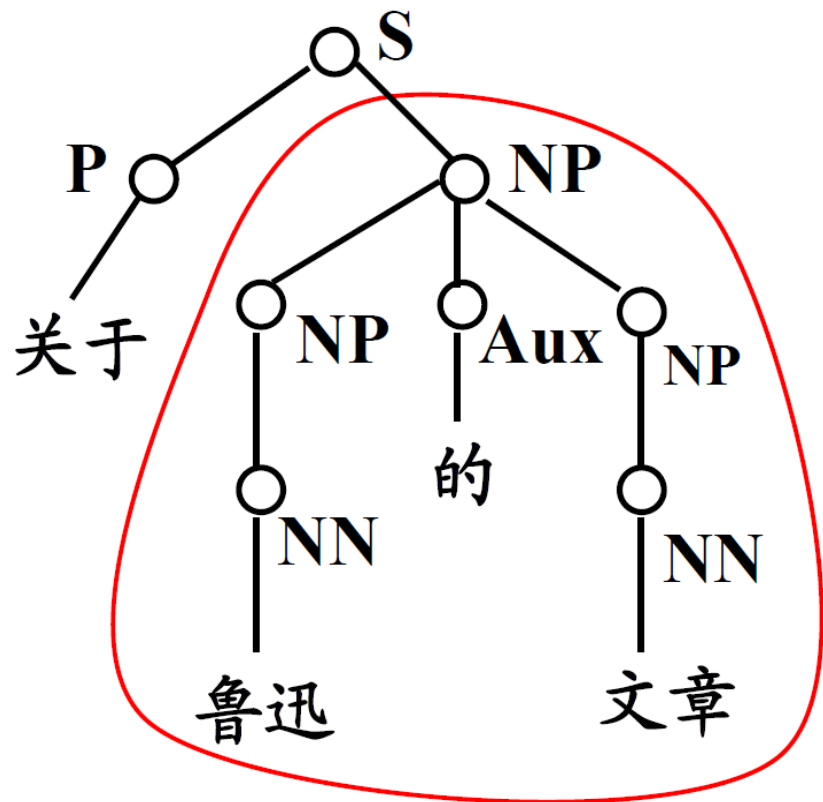
⑤  $NN \rightarrow \text{鲁迅}| \text{文章}$     ⑥  $Aux \rightarrow \text{的}$

短语 “关于鲁迅的文章” 的推导。

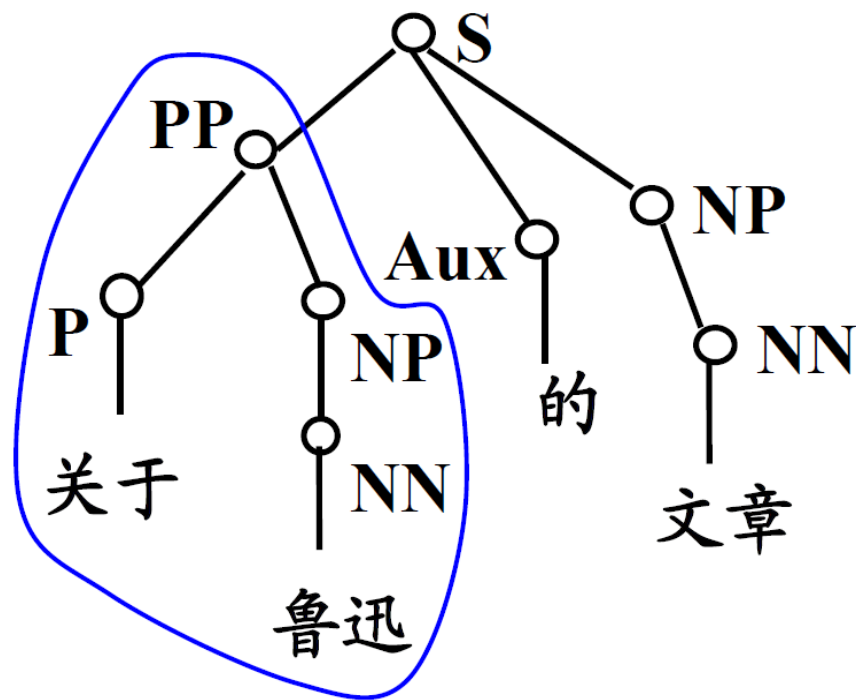
$$\begin{aligned} S &\Rightarrow P\ NP \Rightarrow \text{关于}\ NP \Rightarrow \text{关于}\ NP\ Aux\ NP \\ &\Rightarrow \text{关于}\ NN\ Aux\ NP \Rightarrow \text{关于}\ \text{鲁迅}\ Aux\ NP \\ &\Rightarrow \text{关于}\ \text{鲁迅的}\ NP \Rightarrow \text{关于}\ \text{鲁迅的}\ NN \\ &\Rightarrow \text{关于}\ \text{鲁迅的文章} \end{aligned}$$



## 3.2 形式语言



短语的派生树一(1)



短语的派生树一(2)

---

## 3.3 自动机

# 语言与识别器的对应关系

识别器是有穷地表示无穷语言的另一种方法。每一个语言的句子都能被一定的识别器所接受。

语言类型	识别器类型
0 型	图灵机
1 型	线性带限自动机
2 型	下推自动机
3 型	有限自动机

3.3.1 有限自动机与正则文法

3.3.2 CFG与下推自动机

3.3.3 图灵机和线性带限自动机

3.3.4 各类自动机的区别与联系

## 3.3.1 有限自动机与正则文法

## 3.3.1 有限自动机与正则文法

### ◆ 确定性有限自动机(definite automata, DFA)

确定的有限自动机  $M$  是一个五元组:

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

其中,  $\Sigma$  是输入符号的有穷集合;

$Q$  是状态的有限集合;

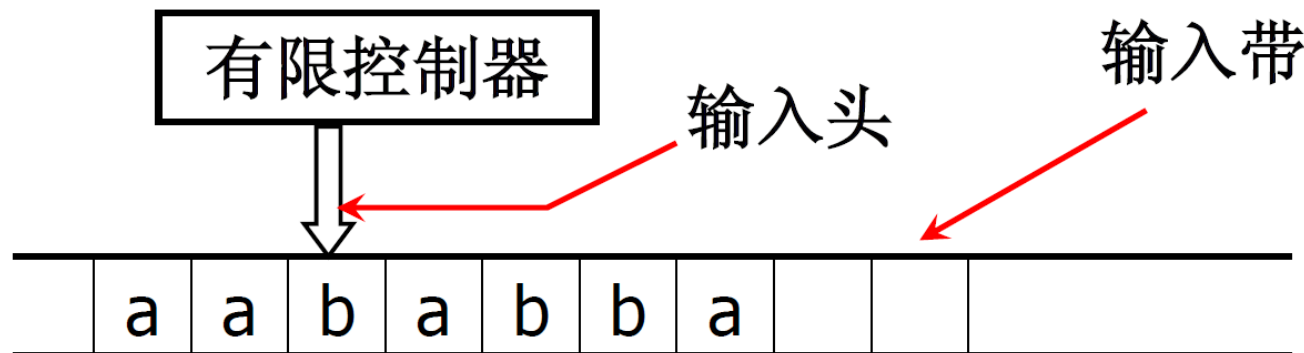
$q_0 \in Q$  是初始状态;

$F$  是终止状态集合,  $F \subseteq Q$ ;

$\delta$  是  $Q$  与  $\Sigma$  的直积  $Q \times \Sigma$  到  $Q$  (下一个状态) 的映射, 支配着有限状态控制的行为, 有时也称为状态转移函数。

# 3.3.1 有限自动机与正则文法

## ◆ DFA 原理示意图

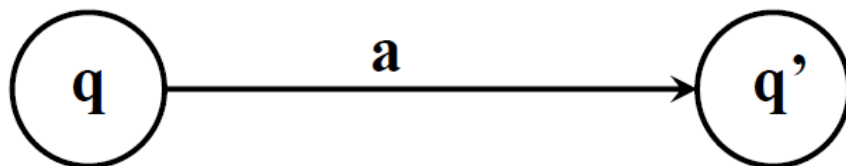


处在状态  $q \in Q$  中的有限控制器，从左到右依次从输入带上读入字符。开始时有限控制器处在状态  $q_0$ ，并注视  $\Sigma^*$  中一个链的最左符号。映射  $\delta(q, a) = q'$  ( $q, q' \in Q, a \in \Sigma$ ) 表示在状态  $q$  时，若输入符号为  $a$ ，则自动机进入状态  $q'$  并且将输入头向右移动一个字符。

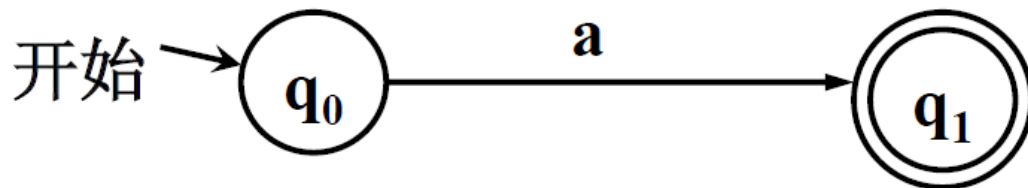
# 3.3.1 有限自动机与正则文法

## ◆ 状态变换图

映射  $\delta(q, a) = q'$  可以由状态变换图描述。



为了明确起见，终止状态用双圈表示，起始状态用有“开始”标记的箭头表示。如：





# 3.3.1 有限自动机与正则文法

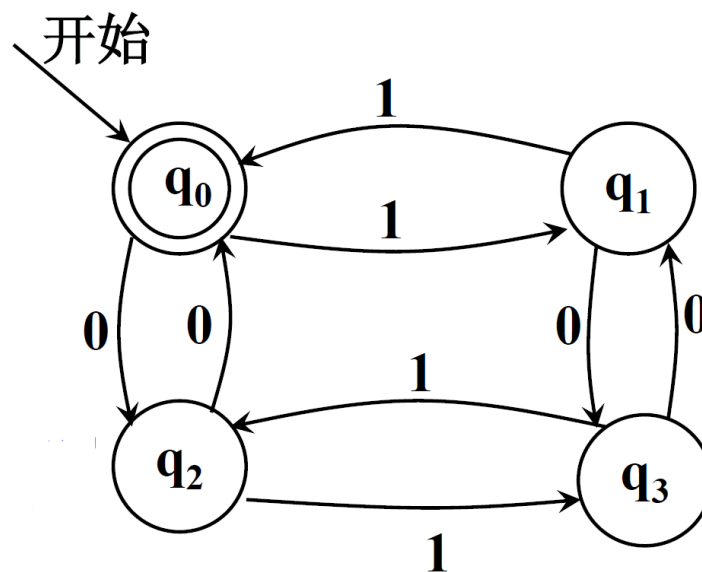
## ◆ DFA 定义的语言

如果一个句子  $x$  使得有限自动机  $M$  有  $\delta(q_0, x) = p$ ,  $p \in F$ , 则称句子  $x$  被  $M$  接受。由  $M$  定义的语言  $T(M)$  就是被  $M$  接受的句子的全集。即:

$$T(M) = \{x | \delta(q_0, x) \in F\}$$

## 3.3.1 有限自动机与正则文法

例3-6:



链  $x = 110101$  被  $M$  接受。

$T(M) = \{\text{含偶数个0和偶数个1的链}\}$

# 3.3.1 有限自动机与正则文法

## ◆ 不确定性有限自动机 (non-definite automata, NFA)

不确定性有限自动机  $M$  是一个五元组：

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

其中， $\Sigma$  是输入符号的有穷集合；

$Q$  是状态的有限集合；

$q_0 \in Q$  是初始状态；

$F$  是终止状态集合， $F \subseteq Q$ ；

$\delta$  是  $Q$  与  $\Sigma$  的直积  $Q \times \Sigma$  到  $Q$  的幂集  $2^Q$  的映射。

## 3.3.1 有限自动机与正则文法

### ◆ 不确定性有限自动机：

根据定义，对于NFA  $M$  有映射：

$$\delta(q, a) = \{q_1, q_2, \dots, q_k\}, k \geq 1$$

其含义是：NFA  $M$  在状态  $q$  时，接受输入符号  $a$  时， $M$  可以选择状态集中的任何一个状态  $q_1, q_2, \dots, q_k$  作为下一个状态，并将输入头向右移动一个字符的位置。

# 3.3.1 有限自动机与正则文法

## ◆ NFA 定义的语言

如果存在一个状态  $p$ , 有  $p \in \delta(q_0, x)$ , 且  $p \in F$ , 则称句子  $x$  被NFA  $M$  接受。由NFA  $M$  定义的语言  $T(M)$  就是被NFA  $M$  接受的句子的全集。即:

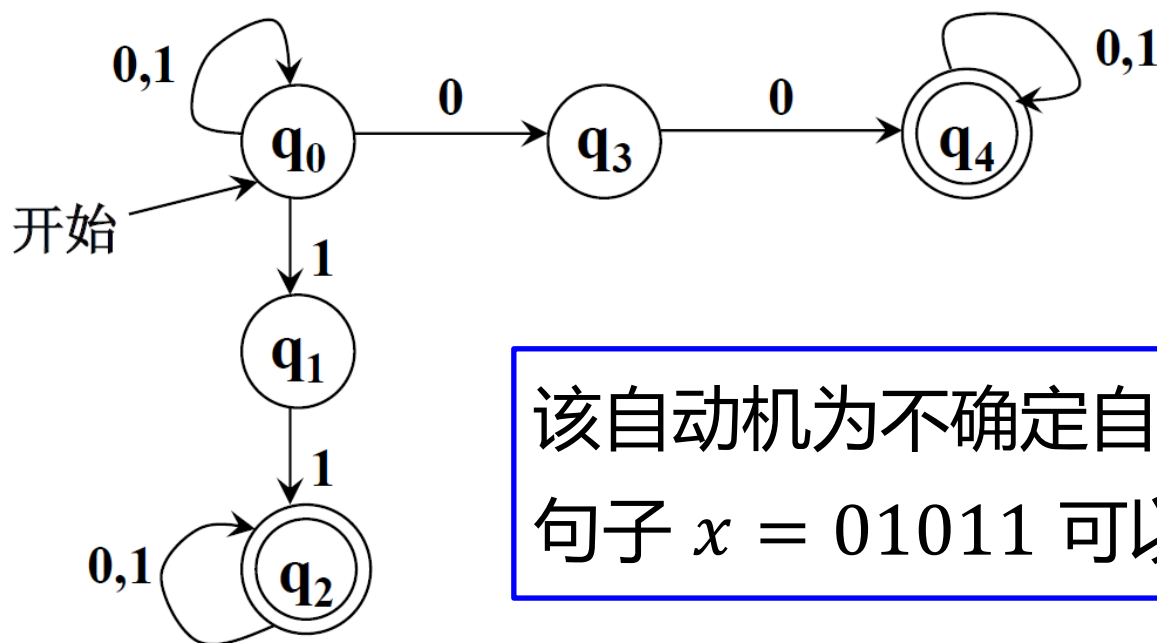
$$T(M) = \{x | p \in \delta(q_0, x) \text{ 且 } p \in F\}$$

# 3.3.1 有限自动机与正则文法

## ◆ NFA与DFA的区别

NFA与DFA的唯一区别是：在NFA中  $\delta(q, a)$  是一个状态集合，而在DFA中  $\delta(q, a)$  是一个状态。

例3-7：



该自动机为不确定自动机；  
句子  $x = 01011$  可以被接受。

# 3.3.1 有限自动机与正则文法

## ◆ NFA与DFA的关系

**定理 3.1:** 设  $L$  是一个被NFA所接受的句子的集合, 则存在一个DFA, 它能够接受  $L$ 。

(证明略, 数学归纳法)

**说明:** 由于DFA与NFA所接受的是同样的链集, 所以一般情况下无需区分它们, 二者统称为有限自动机 (finite automata, FA)。

## 3.3.1 有限自动机与正则文法

### ◆ 正则文法与有限自动机的关系

**定理 3.2:** 若  $G = (V_N, V_T, P, S)$  是一个正则文法, 则存在一个有限自动机  $M = (\Sigma, Q, \delta, q_0, F)$ , 使得:  $T(M) = L(G)$ 。



## 3.3.1 有限自动机与正则文法

◆ 由正则文法  $G$  构造FA  $M$  的一般步骤:

- ① 令  $\Sigma = V_T$ ,  $Q = V_N \cup \{T\}$ ,  $q_0 = S$ , 其中,  $T$  是一个新增加的非终结符。
- ② 如果在  $P$  中有产生式  $S \rightarrow \varepsilon$ , 则  $F = \{S, T\}$ , 否则  $F = \{T\}$ 。
- ③ 如果在  $P$  中有产生式  $B \rightarrow a$ ,  $B \in V_N$ ,  $a \in V_T$ , 则  $T \in \delta(B, a)$ 。
- ④ 如果在  $P$  中有产生式  $B \rightarrow aC$ ,  $B, C \in V_N$ ,  $a \in V_T$ , 则  $C \in \delta(B, a)$ 。
- ⑤ 对于每一个  $a \in V_T$ , 有  $\delta(T, a) = \emptyset$ 。

## 3.3.1 有限自动机与正则文法

### 例3-8:

给定正则文法  $G = (V_N, V_T, P, S)$ , 其中  $V_N = \{S, B\}$ ,  $V_T = \{a, b\}$ ,  $P = \{S \rightarrow aB, B \rightarrow bS | aB | a\}$  构造与  $G$  等价的NFA。

(1) 设NFA  $M = (\Sigma, Q, \delta, q_0, F)$ , 根据上述步骤有:

$$\Sigma = V_T = \{a, b\} \quad Q = V_N \cup \{T\} = \{S, B, T\}$$

$$q_0 = S \quad F = \{T\}。$$

## 3.3.1 有限自动机与正则文法

### 例3-8:

(2) 映射  $\delta$  为:

$$\delta(S, a) = \{B\} \quad (\text{因为有规则 } S \rightarrow aB)$$

$$\delta(S, b) = \emptyset$$

$$\delta(B, a) = \{B, T\} \quad (\text{因为有 } B \rightarrow aB, \quad B \rightarrow a)$$

$$\delta(B, b) = \{S\} \quad (\text{因为有 } B \rightarrow bS)$$

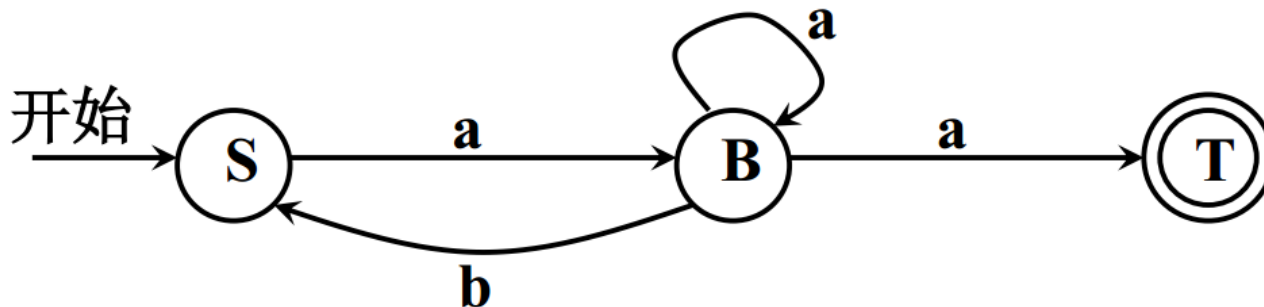
$$\delta(T, a) = \emptyset$$

$$\delta(T, b) = \emptyset$$

## 3.3.1 有限自动机与正则文法

### 例3-8:

等价的NFA的状态变换图为：



$$\delta(S, a) = \{B\}$$

(因为有规则  $S \rightarrow aB$ )

$$\delta(B, a) = \{B, T\}$$

(因为有  $B \rightarrow aB$ ,  $B \rightarrow a$ )

$$\delta(B, b) = \{S\}$$

(因为有  $B \rightarrow bS$ )

## 3.3.1 有限自动机与正则文法

**定理 3.3:** 设若  $M = (\Sigma, Q, \delta, q_0, F)$  是一有限自动机, 则存在正则文法  $G = (V_N, V_T, P, S)$  使  $L(G) = T(M)$ 。

由  $M$  构造  $G$  的一般步骤:

- (1) 令  $V_N = Q$ ,  $V_T = \Sigma$ ,  $S = q_0$ ;
- (2) 如果  $C \in \delta(B, a)$ ,  $B, C \in Q$ ,  $a \in \Sigma$ , 则在  $P$  中有产生式  $B \rightarrow aC$ ;
- (3) 如果  $C \in \delta(B, a)$ ,  $C \in F$ , 则在  $P$  中有产生式  $B \rightarrow a$ 。

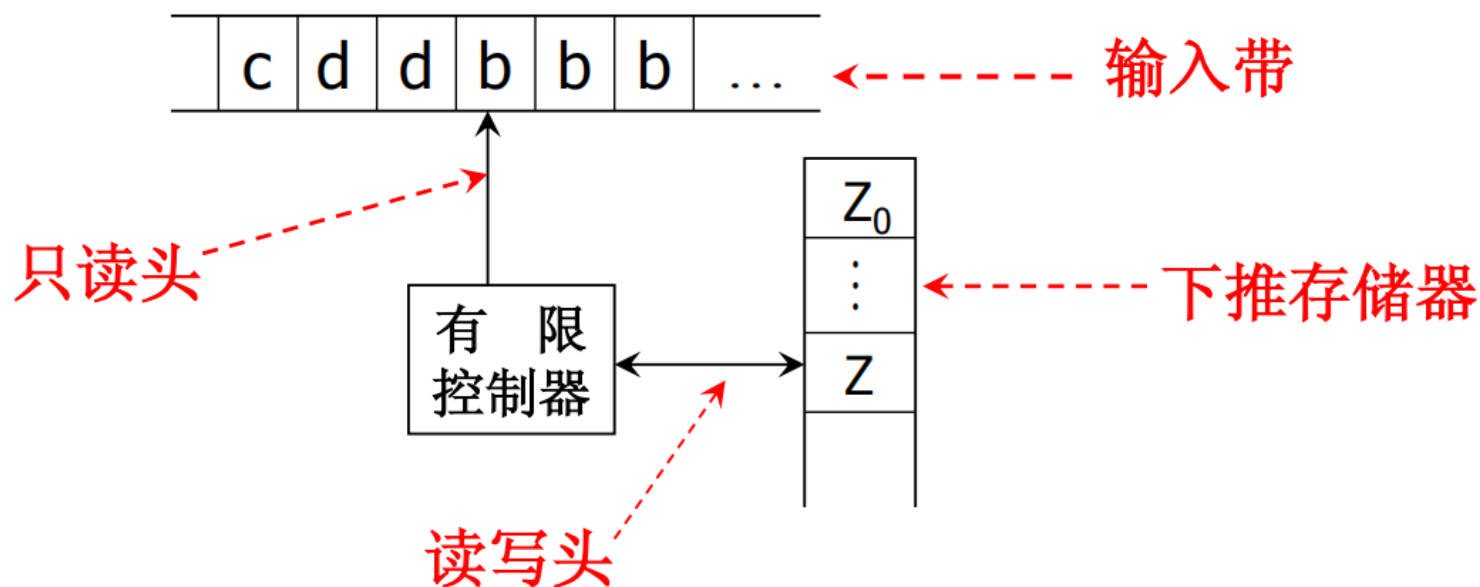
结论: 对于任意一个正则文法所产生的语言, 总可以构造一个确定的有限自动机识别它。

## 3.3.2 CFG与下推自动机

## 3.3.2 CFG与下推自动机

### ◆ 下推自动机 (push-down automata, PDA)

PDA可以看成是一个带有附加的下推存储器的有限自动机，下推存储器是一个栈，如下图所示：



## 3.3.2 CFG与下推自动机

### ◆ PDA 的定义

一个不确定的PDA可以表达成一个7元组:

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

其中,  $\Sigma$  是输入符号的有穷集合;

$Q$  是状态的有限集合;  $q_0 \in Q$  是初始状态;

$\Gamma$  为下推存储器符号的有穷集合;

$Z_0 \in \Gamma$  为最初出现在下推存储器顶端的符号;

$F$  是终止状态集合,  $F \subseteq Q$ ;

$\delta$  是从  $Q \times (\Sigma \times \{\varepsilon\}) \times \Gamma$  到  $Q \times \Gamma^*$  子集的映射。



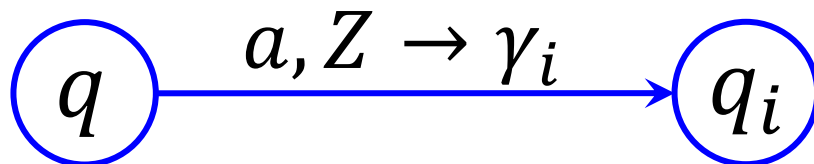
## 3.3.2 CFG与下推自动机

### ◆ 映射关系 $\delta$ 的解释

映射关系  $\delta(q, a, Z) = \{(q_1, \gamma_1), (q_2, \gamma_2) \cdots (q_m, \gamma_m)\}$

其中  $q_1, q_2 \cdots q_m \in Q$ ,  $a \in \Sigma$ ,  $Z \in \Gamma$ ,  $\gamma_1, \gamma_2 \cdots \gamma_m \in \Gamma^*$ 。

该映射的意思是：当PDA处于状态  $q$ ，面临输入符号  $a$  时，自动机将进入  $q_i, i = 1, 2 \cdots m$  状态，并以  $\gamma_i$  来代替下推存储器(栈)顶端符号  $Z$ ，同时将输入头指向下一个字符。当  $Z$  被  $\gamma_i$  取代时， $\gamma_i$  的符号按照从左到右的顺序依次从下向上推入到存储器。



## 3.3.2 CFG与下推自动机

### ◆ 映射关系 $\delta$ 的解释

特殊情况下，当

$$\delta(q, \varepsilon, Z) = \{(q_1, \gamma_1), (q_2, \gamma_2) \cdots (q_m, \gamma_m)\}$$

时，输入头位置不移动，只用于处理下推存储器内部的操作，叫作 “ $\varepsilon$  移动”。

## 3.3.2 CFG与下推自动机

### ◆ 符号约定

设有序对  $(q, \gamma), q \in Q, \gamma \in \Gamma^*$ , 对于  $a \in (\Sigma \cup \{\varepsilon\})$ ,  $\beta \in \Gamma^*, Z \in \Gamma$  如果  $(q', \beta) \in \delta(q, a, Z), q', q \in Q$ , 则表达式

$$a: (q, Z_\gamma) \vdash_M (q', \beta_\gamma)$$

表示根据下推自动机的状态变换规则, 输入  $a$  能使下推自动机  $M$  由格局  $(q, Z_\gamma)$  变换到格局  $(q', \beta_\gamma)$ , 或称  $a: (q, Z_\gamma) \vdash_M (q', \beta_\gamma)$  为合法转移。零次或多次合法转移记为:  $a: (q, Z_\gamma) \vdash_M^* (q', \beta_\gamma)$ ,  $M$  可以省略不写。

## 3.3.2 CFG与下推自动机

### ◆ 下推自动机接受的语言

下推自动机  $M$  所接受的语言定义为:

$$T(M) = \{x \mid x: (q_0, Z_0) \xrightarrow{*}_M (q, \gamma), \gamma \in \Gamma^*, q \in F\}$$

**Definition 1:** A PDA accepts its input by consuming it and entering an accepting state.

**Definition 2:** The set of strings that cause the PDA to empty its stack, starting from the initial instantaneous description (ID).

## 3.3.2 CFG与下推自动机

### 例3-9:

下推自动机  $M = (\Sigma, Q, \Gamma, \delta, q_0, Z_0, F)$  接受语言  $L = \{wcw^R \mid w \in \{a, b\}^*\}$ , 其中  $Q = \{0, 1\}$ ,  $\Sigma = \{a, b, c\}$ ,  $\Gamma = \{A, B\}$ ,  $q_0 = 0$ ,  $Z_0 = \#$ ,  $F = \{1\}$ ,  $\delta$  定义如下:

- (1)  $\delta(0, a, \varepsilon) \mid_M \{(0, A)\}$       (2)  $\delta(0, b, A) \mid_M \{(0, AB)\}$
- (3)  $\delta(0, b, B) \mid_M \{(0, BB)\}$       (4)  $\delta(0, c, B) \mid_M \{(1, B)\}$
- (5)  $\delta(1, b, B) \mid_M \{(1, \varepsilon)\}$       (6)  $\delta(1, a, A) \mid_M \{(1, \varepsilon)\}$

## 3.3.2 CFG与下推自动机

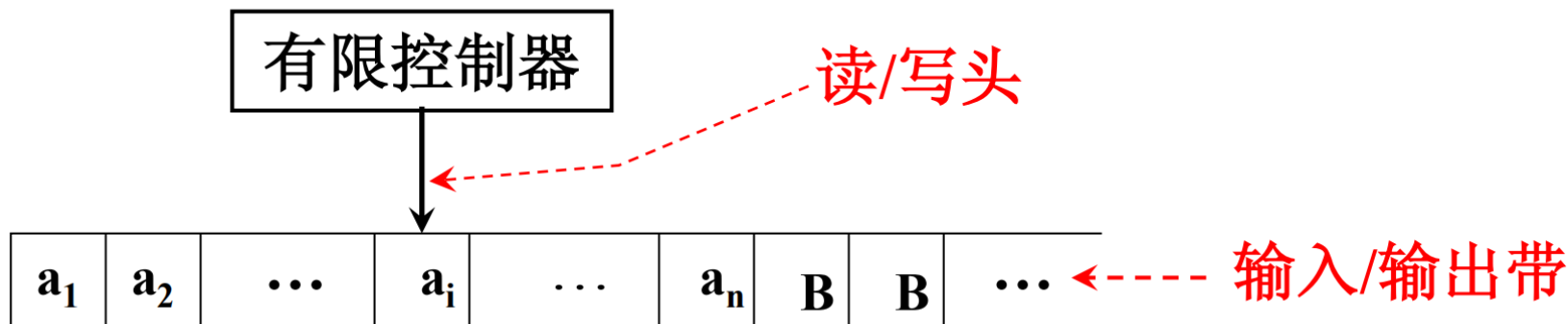
➤ 对于输入 *abbcbbba* 下推自动机 *M* 的处理步骤为：

状态	输入	栈	运用的规则
0	abbcbbba	#	—
0	bcbba	A #	$\delta(0, a, \varepsilon) \Rightarrow \{(0, A)\}$
0	cbba	BA #	$\delta(0, b, A) \Rightarrow \{(0, AB)\}$
0	bba	BBA #	$\delta(0, b, B) \Rightarrow \{(0, BB)\}$
1	ba	BBA #	$\delta(0, c, B) \Rightarrow \{(1, B)\}$
1	a	BA #	$\delta(1, b, B) \Rightarrow \{(1, \varepsilon)\}$
1		A #	$\delta(1, b, B) \Rightarrow \{(1, \varepsilon)\}$
1		#	$\delta(1, a, A) \Rightarrow \{(1, \varepsilon)\}$

### 3.3.3 图灵机和线性带限自动机

# 3.3.3 图灵机和线性带限自动机

## ◆ 图灵机的理解



给定字符串  $\alpha$  存放于输入/输出带上，开始时图灵机  $M$  处于状态  $q_0$ ，它的读/写头扫描着  $\alpha$  的最左字符。根据转移函数  $\delta$  的定义，即对于目前状态以及正扫描着的字符， $M$  改变当前状态、读/写头扫描的字符，以及读写/头的位置。



# 3.3.3 图灵机和线性带限自动机

## ◆ 图灵机与有限自动机的区别

图灵机可以通过其读/写头改变输入带的字符。

## ◆ 图灵机的定义

一个图灵机  $T$  可以表达成一个6元组：

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

其中， $\Sigma$  是输入符号的有穷集合；

$Q$  是状态的有限集合； $q_0 \in Q$  是初始状态；

$\Gamma \in \Sigma \cup \{B\}$ ， $B$  为空白字符； $F$  是终止状态集合， $F \subseteq Q$ ；

$\delta$  是从  $Q \times \Sigma$  到  $Q \times \Sigma \times \{R, S, L\}$  子集的映射。

$R, L, S$  分别表示右移一格、左移一格和停止不动。

# 3.3.3 图灵机和线性带限自动机

## ◆ 图灵机的解释

图灵机  $T$  的一个格局可以定义为  $(q, \alpha, i)$ , 其中  $q \in Q$ ,  $\alpha$  是字符串, 且  $\alpha \in \Sigma$ ,  $i$  是整数, 表示  $T$  的读/写头到  $\alpha$  左端的距离。图灵机  $T$  通过如下转移动作引起格局变化:

假设  $(q, A_1A_2\dots A_n, i)$ ,  $1 \leq i \leq n+1$  是当前  $T$  的格局,

(1) 如果  $\delta(q, A_i) = (p, X, R)$ ,  $1 \leq i \leq n$ , 那么,  $T$  的基本运行(即指定为  $T$  的基本移动)可以表示为:

$$(q, A_1A_2\dots A_n, i) \vdash_T (p, A_1A_2\dots A_{i-1}XA_{i+1}\dots A_n, i+1)$$

即  $T$  的读/写头在  $i$  位置写入符号  $X$ , 并将读/写头向右移动一个位置。

# 3.3.3 图灵机和线性带限自动机

## ◆ 图灵机的解释

(2) 如果  $\delta(q, A_i) = (p, X, L)$ ,  $2 \leq i \leq n$ , 那么,

$$(q, A_1A_2\ldots A_n, i) \mid_T (p, A_1A_2\ldots A_{i-1}XA_{i+1}\ldots A_n, i-1)$$

即 T 的读/写头在  $i$  位置写入符号  $X$ , 并将读/写头向左移动一个位置, 但不超出输入带的左端。

(3) 如果  $i = n+1$ , 读写头超出原字符串的右端, 读到的是空白符号  $B$ , 此时如果有  $\delta(q, B) = (p, X, R)$ , 那么

$$(q, A_1A_2\ldots A_n, n+1) \mid_T (p, A_1A_2\ldots A_nX, n+2)$$

而如果  $\delta(q, B) = (p, X, L)$ , 则:

$$(q, A_1A_2\ldots A_n, n+1) \mid_T (p, A_1A_2\ldots A_nX, n)$$

# 3.3.3 图灵机和线性带限自动机

## ◆ 图灵机接受的语言

如果  $T$  的两个格局  $X$  和  $Y$  之间的基本移动(包括不移动)的次数是有限的, 且互相关联, 则可记为:

$$X \vdash_T^* Y$$

由图灵机  $T$  所接受的语言定义为:

$$L(T) = \{\alpha | \alpha \in \Sigma^*, (q_0, \alpha, 1) \vdash_T^* (q, \beta, i), q \in F, \beta \in \Gamma^*\}$$

# 3.3.3 图灵机和线性带限自动机

## ◆ 图灵机接受的语言

给定一个识别语言  $L$  的图灵机  $T$ ，不失一般性，我们假定每当输入串被接受时， $T$  就停机，即没有下一个动作。另一方面，对于未接受的链， $T$  可能不停机。

- **定理 3.4**：如果  $L$  是一个由 0 型文法产生的语言，则  $L$  可被一个图灵机所接受。
- **定理 3.5**：如果  $L$  可被一个图灵机所接受，则  $L$  是一个由 0 型文法产生的语言。

# 3.3.3 图灵机和线性带限自动机

## ◆ 线性带限自动机

线性带限自动机是一个确定的单带图灵机。其读写头不能超越原输入带上字符串的初始和终止位置。即线性带限自动机的存储空间被输入符号串的长度所限制。

# 3.3.3 图灵机和线性带限自动机

## ◆ 线性带限自动机

定义：一个线性带限自动机  $M$  可以表达成一个6元组：

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

其中， $\Sigma$  是输入/输出带上字符的有穷集合，包括两个特殊符号  $\#$  和  $\$$ ，分别表示输入链的左端和右端标志；

$Q$  是状态的有限集合； $q_0 \in Q$  是初始状态；

$\Gamma$  是输入带上符号的有穷集；

$F$  是终止状态集合， $F \subseteq Q$ ；

$\delta$  是从  $Q \times \Gamma$  到  $Q \times \Gamma \times \{R, L\}$  子集的映射。

# 3.3.3 图灵机和线性带限自动机

## ◆ 线性带限自动机所接受的语言

线性带限自动机  $M$  的格局，以及两个格局之间的关系  $\vdash_T^*$  的定义与图灵机的相同。唯一不同的是对读写头位置的限制。在线性带限自动机中，对于读写头超出输入字符串长度范围时，转移动作没有定义。

线性带限自动机  $M$  接受的语言：

$$L(T) = \{\alpha \mid \alpha \in (\Sigma - \{\#, \$\})^*, (q_0, \# \alpha \$, 1) \vdash_T^* (q, \beta, i), q \in F, \beta \in \Gamma^*\}$$



# 3.3.3 图灵机和线性带限自动机

## ◆ 线性带限自动机所接受的语言

对于任何  $q \in Q$ ,  $A \in \Gamma$ , 如果映射  $\delta(q, A)$  包含的成员不超过一个, 则线性带限自动机是确定的。

➤ **定理 3.6:** 如果  $L$  是一个前后文有关语言, 则  $L$  由一个不确定的线性带限自动机所接受。反之, 如果  $L$  被一个线性带限自动机所接受, 则  $L$  是一个前后文有关语言。

## 3.3.4 各类自动机的区别与联系

## 3.3.4 各类自动机的区别与联系

### ◆ 主要区别：

各类自动机的主要区别是它们能够使用的信息存储空间差异：有限状态自动机只能用状态来存储信息；下推自动机除了可以用状态以外，还可以用下推存储器(栈)；线性带限自动机可以利用状态和输入/输出带本身。因为输入/输出带没有“先进后出”的限制，因此，其功能大于栈；而图灵机的存储空间没有任何限制。

## 3.3.4 各类自动机的区别与联系

### ◆ 识别语言的能力：

有限自动机等价于正则文法；下推自动机等价于上下文无关文法；线性带限自动机等价于上下文有关文法，图灵机等价于无约束文法。

## 3.4 有限自动机在 NLP中的应用

## 3.4 有限自动机的应用

### ◆ 英语单词拼写检查 [Oflazer, 1996]

设  $X$  为拼写错误的字符串，其长度为  $m$ ， $Y$  为  $X$  对应的正确的单词(答案)，其长度为  $n$ 。则  $X$  和  $Y$  的编辑距离  $ed(X[m], Y[n])$  定义为：从字符串  $X$  转换到  $Y$  需要的插入、删除、替换和交换两个相邻的基本单位(字符)的最小个数。如：

$$ed(\text{recogninze}, \text{recognize}) = 1$$

$$ed(\text{sailn}, \text{failing}) = 3$$

## 3.4 有限自动机的应用

### ◆ 英语单词拼写检查 [Oflazer, 1996]

假设  $Z = z_1 z_2 \cdots z_p$  为字母表  $A$  上的  $p$  个字母构成的字符串,  $Z[j]$  表示含有  $j (j \geq 1)$  个字符的子串。  $X[m]$  为拼写错误的字符串, 其长度为  $m$ ,  $Y[n]$  为与  $X$  串接近的字符串(一个候选), 其长度为  $n$ 。则给定两个串  $X$  和  $Y$  的编辑距离  $ed(X[m], Y[n])$  可以通过循环计算出从字符串  $X$  转换到  $Y$  需要进行插入、删除、替换和交换两个相邻的字符操作的最少次数。

## 3.4 有限自动机的应用

### ◆ 英语单词拼写检查 [Ofizer, 1996]

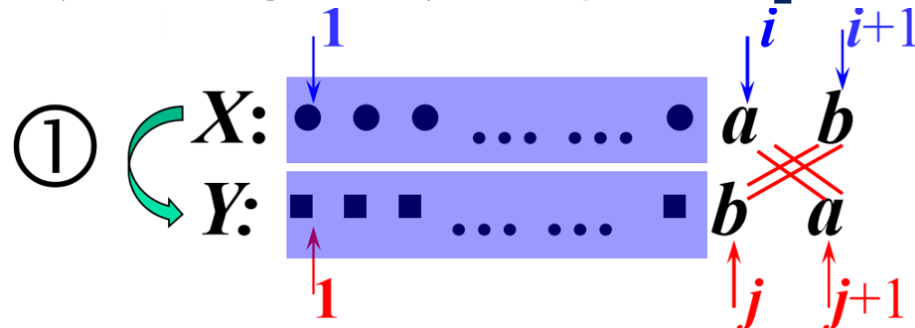
- (1) 如果  $x_{i+1} = y_{i+1}$  (两个串的最后字母相同) ,  
则  $ed(X[i+1], Y[j+1]) = ed(X[i], Y[j]);$
- (2) 如果  $x_i = y_{j+1}$  , 并且  $x_{i+1} = y_j$  (最后两个字符交叉相等) , 则

$$\begin{aligned} & ed(X[i+1], Y[j+1]) \\ &= 1 + \min\{ed(X[i-1], Y[j-1]), \\ &\quad ed(X[i], Y[j+1]), \\ &\quad ed(X[i+1], Y[j])\} \end{aligned}$$



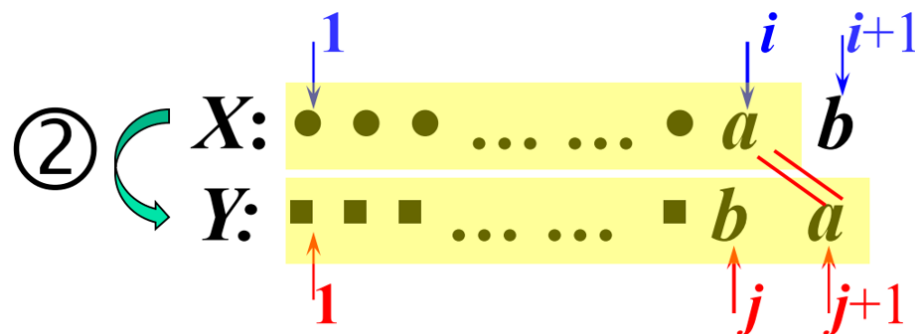
# 3.4 有限自动机的应用

## ◆ 英语单词拼写检查 [Oflazer, 1996]



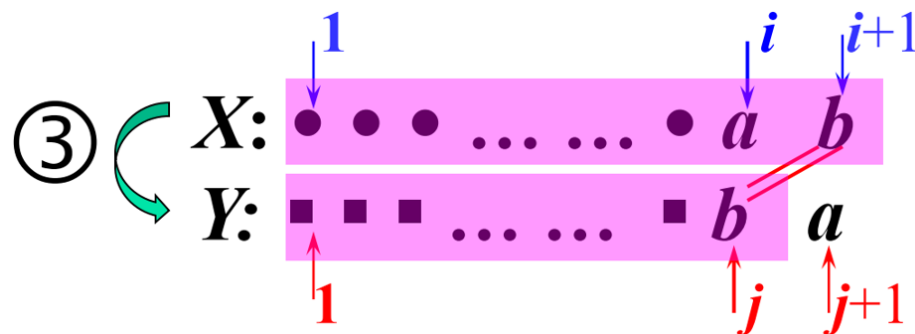
$ed(X[i-1], Y[j-1])$

(交换  $X$  中最末端的两个字符)



$ed(X[i], Y[j+1])$

( $X$  删除  $x_{i+1}$  以后可能与  $Y$  相似)



$ed(X[i+1], Y[j])$

( $X$  可能与  $Y$  的子串相似, 且需要插入  $y_{j+1}$ )

## 3.4 有限自动机的应用

### ◆ 英语单词拼写检查 [Oflazer, 1996]

(3) 其它情况下( $x_{i+1} \neq y_{j+1}$ , 且 $x_i \neq y_{j+1}$ 或 $x_{i+1} \neq y_j$ )

$$ed(X[i+1], Y[j+1]) = 1 + \min\{ed(X[i], Y[j]), \\ ed(X[i], Y[j+1]), \\ ed(X[i+1], Y[j])\}$$

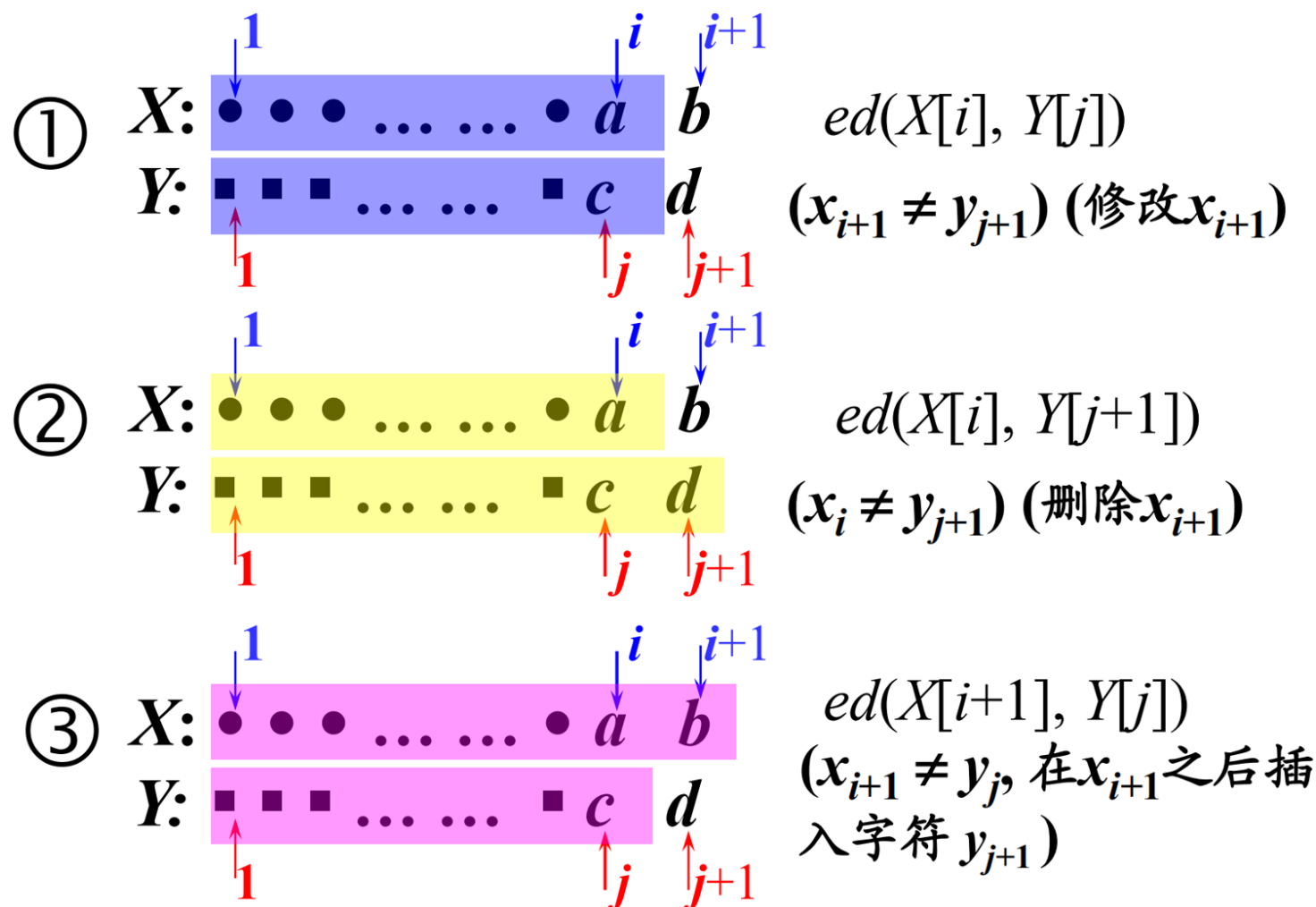
其中  $ed(X[0], Y[j]) = j$  ( $0 \leq j \leq n$ ) ( $X$ 长度为0)

$ed(X[i], Y[0]) = i$  ( $0 \leq i \leq m$ ) ( $Y$ 长度为0)

$ed(X[-1], Y[j]) = ed(X[i], Y[-1]) = \max\{m, n\}$   
(边界约定)

# 3.4 有限自动机的应用

## ◆ 英语单词拼写检查 [Oflazer, 1996]



## 3.4 有限自动机的应用

### ◆ 英语单词拼写检查 [Oflazer, 1996]

下式定义一个确定的有限状态机  $R$ :

$$R = (Q, A, \delta, q_0, F)$$

其中,  $Q$  表示状态集;

$A$  表示输入字符集;

$$\delta: Q \times A \rightarrow Q$$

$q_0 \in Q$  为起始状态;

$F \subseteq Q$  为终止状态集;

## 3.4 有限自动机的应用

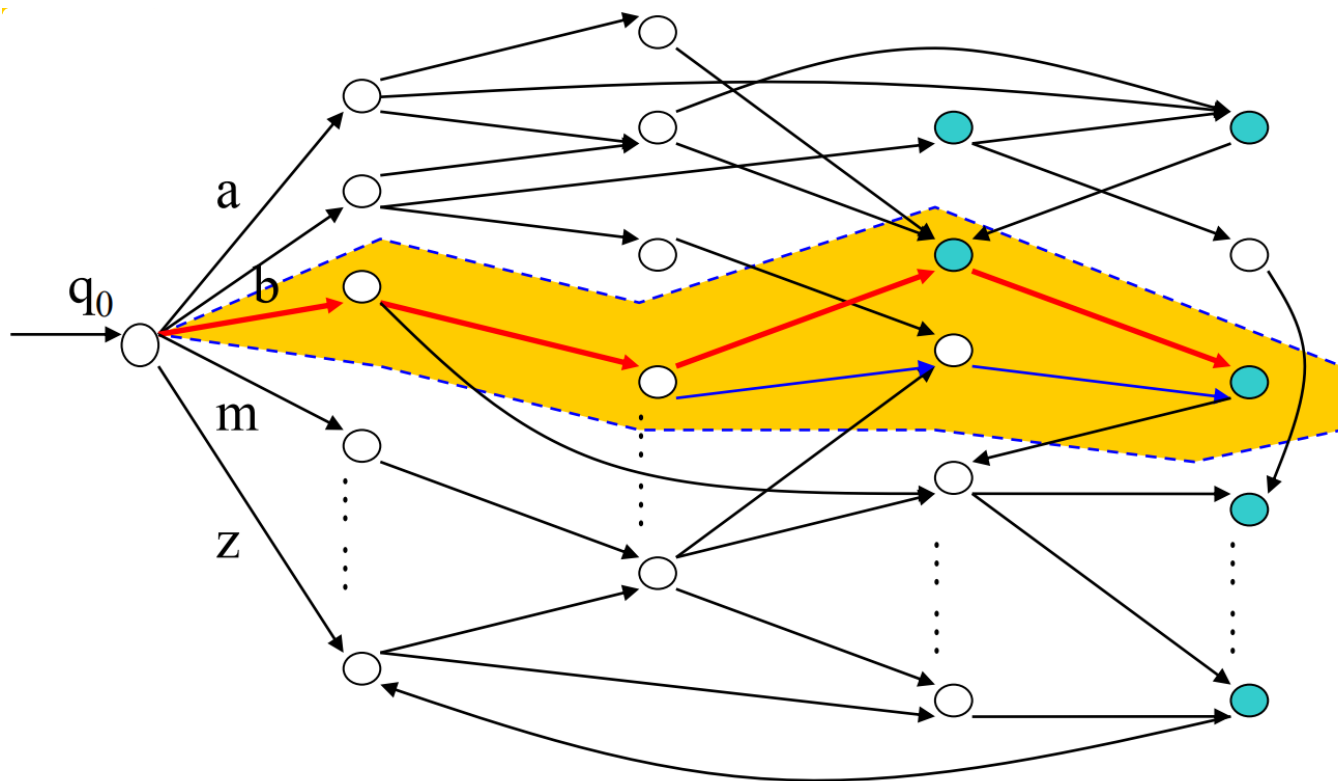
### ◆ 英语单词拼写检查 [Oflazer, 1996]

如果  $L \subseteq A^*$  表示有限状态机  $R$  接受的语言，字母构成的所有合法单词都是有限状态机中的一条路径。给定一个输入串，对其进行检查的过程就是在给定阈值  $t$  ( $t > 0$ ) 的情况下，寻找那些与输入串的编辑距离小于  $t$  的路径。那么，一个字符串  $X[m] \notin L$  能够被  $R$  识别的条件是存在非空集合：

$$C = \{Y[n] | Y[n] \in L \text{ and } ed(X[m], Y[n]) \leq t\}$$

# 3.4 有限自动机的应用

## ◆ 英语单词拼写检查 [Oflazer, 1996]



英文单词可用  
键树(又称数字  
查找树, digital  
search tree)存储

说明: 蓝色节  
点表示终结节  
点, 下同。

对于某一字符串  $X$ , 搜索与之编辑距离最短的单词(路径)。

## 3.4 有限自动机的应用

### ◆ 英语单词拼写检查 [Oflazer, 1996]

为了提高搜索速度，可以将搜索空间限定在一个较小的范围内，尽早把那些编辑距离超过给定阈值  $t$  的路径剪枝。为了判断哪些路径应该被剪枝，Oflazer提出了剪除编辑距离(cut-off edit distance)的概念，用以度量错误的输入串的子串与候选正确串之间的最小编辑距离。

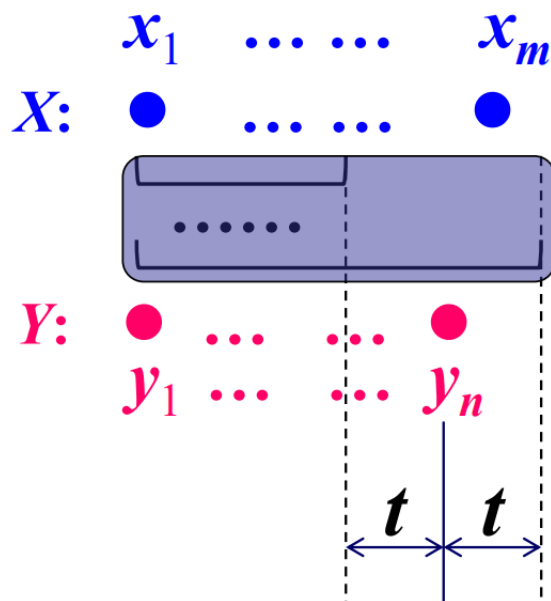
## 3.4 有限自动机的应用

### ◆ 英语单词拼写检查 [Oflazer, 1996]

设  $Y$  是局部候选串,  $X$  是输入串, 则剪除距离的定义:

$$cuted(X[m], Y[n]) = \min_{l \leq i \leq u} \{ed(X[i], Y[n])\}$$

其中,  $l = \max(1, n - t), u = \min(m, n + t)$ 。



**注意:** 阈值  $t$  有两个用途: 确定截取  $X$  的范围; 限定编辑距离。



## 3.4 有限自动机的应用

### ◆ 英语单词拼写检查 [Oflazer, 1996]

例如:  $t = 2$ ,  $X = reprter(m = 7)$ ,  $Y = repo(n = 4)$

那么:  $l = \max\{1, 4 - 2\} = 2$ ;  $u = \min\{7, 4 + 2\} = 6$

$cuted(reprter, repo)$

$= \min\{ed(re, repo) = 2,$

$ed(rep, repo) = 1,$

$ed(repr, repo) = 1,$

$ed(reprt, repo) = 2,$

$ed(reprte, repo) = 3\}$

$= 1$

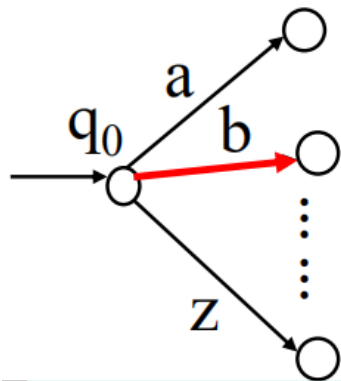
结论: rep和repr为最接近的两个候选。

## 3.4 有限自动机的应用

### ◆ 英语单词拼写检查 [Oflazer, 1996]

采用深度优先搜索算法从自动机中选择路径。

假设  $X = bax$ ,  $t = 2$ 。那么  $Y = a|b|c|\cdots|z$ ,  $l = \max\{1, 1 - 2\} = 1$ ,  $u = \min\{3, 1 + 2\} = 3$ 。即从  $X$  中取长度在  $1 \sim 3$  个字符范围内的子串  $X' = \{b, ba, bax\}$ , 分别计算与  $Y$  之间的编辑距离, 保留那些  $ed(X', Y) \leq t$  的路径, 选择  $ed$  最小的路径继续扩展。



## 3.4 有限自动机的应用

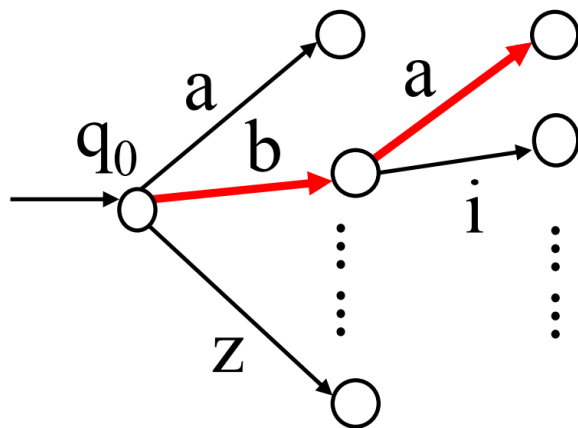
### ◆ 英语单词拼写检查 [Oflazer, 1996]

采用  $X = bax, t = 2$ 。  $Y = \{ba, bi, bo, \dots\}$ 。

截取  $X$ :  $l = \max\{1, 2 - 2\} = 1$ ,  $u = \min\{3, 2 + 2\} = 3$ 。

$$X' = \{b, ba, bax\}$$

保留那些  $ed(X', Y) \leq t$  的路径, 选择  $ed$  最小的路径继续扩展



## 3.4 有限自动机的应用

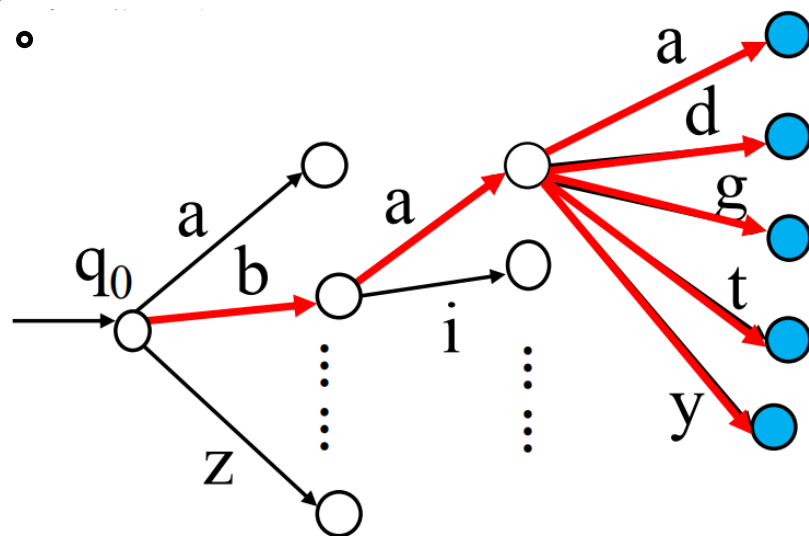
### ◆ 英语单词拼写检查 [Oflazer, 1996]

采用  $X = bax, t = 2$ 。  $Y = \{baa, bad, bag, bat, bay\}$ 。

截取  $X$ :  $l = \max\{1, 3 - 2\} = 1, u = \min\{3, 2 + 2\} = 3$ 。

$$X' = \{b, ba, bax\}$$

保留那些  $ed(X', Y) \leq t$  的路径, 选择  $ed$  最小的路径继续扩展  $Y$ 。



## 3.4 有限自动机的应用

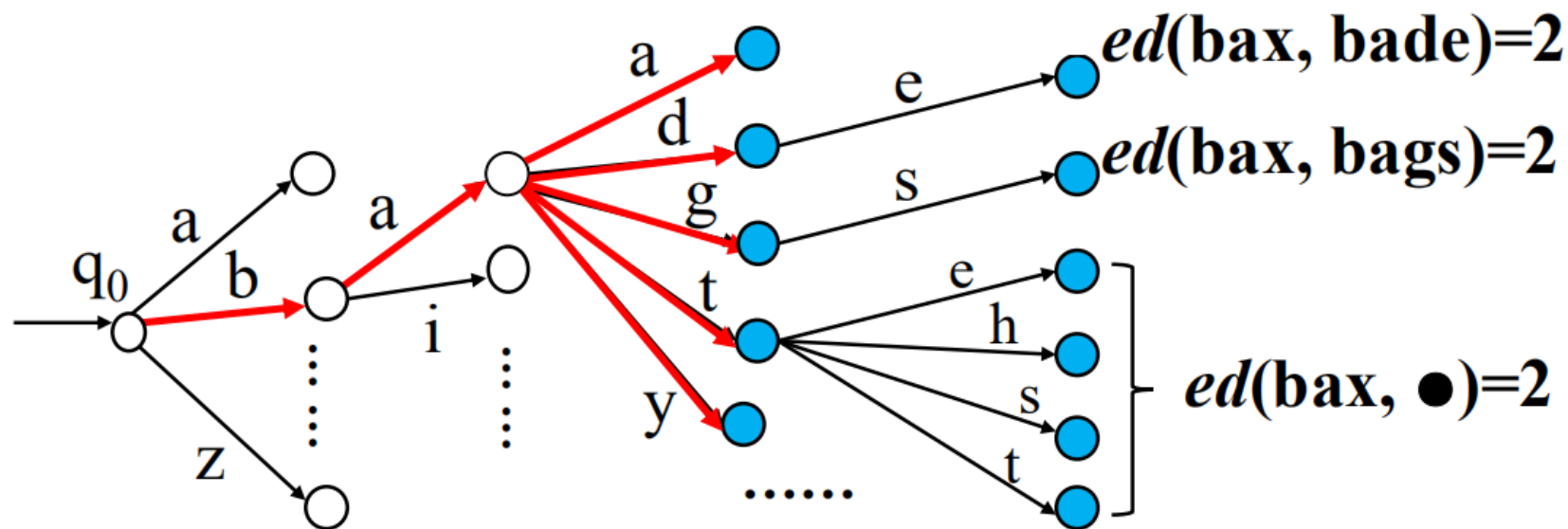
### ◆ 英语单词拼写检查 [Oflazer, 1996]

采用  $X = bax, t = 2$ 。  $Y = \{bade\}$ 。

截取  $X$ :  $l = \max\{1, 4 - 2\} = 2$ ,  $u = \min\{3, 4 + 2\} = 3$ 。

$$X' = \{ba, bax\}$$

保留那些  $ed(X', Y) \leq t$  的路径, 选择  $ed$  最小的路径。



## 3.4 有限自动机的应用

### ◆ 英语单词拼写检查 [Oflazer, 1996]

#### 算法:

```
push( $\epsilon$ ,  $q_0$ ); /* 将一个空的字符串和初始节点压栈 stack */
while stack  $\neq$  NULL {
    pop( $Y'$ ,  $q_i$ ); /* 从栈顶弹出一个部分字符串  $Y'$  和状态节点 */
    for all  $q_j$ ,  $a$ :  $\delta(q_i, a)=q_j$  {
         $Y = \text{concat}(Y', a)$ ; /* 扩展候选串, 把字符  $a$  连接到  $Y'$  上。 */
        if ( $\text{cuted}(X, Y) \leq t$ ); /* 保证剪除距离在限定的范围 */
        push( $Y$ ,  $q_j$ );
        if ( $\text{ed}(X, Y) \leq t$ ) and  $q_j \in F$ 
            output( $Y$ ); /* 输出  $Y$  */
    }
}
```

**说明:** 由于该算法采用深度优先搜索策略, 因此输出结果未必最优。

## 3.4 有限自动机的应用

### ◆ 有限自动机用于英语单词形态分析 [Allen, 1995]

英语单词形态变化非常普遍，例如：

eat: eats, eating, ate, eaten

happy: happier, happiest

seed ?

## 3.4 有限自动机的应用

### ◆ 有限自动机用于英语单词形态分析

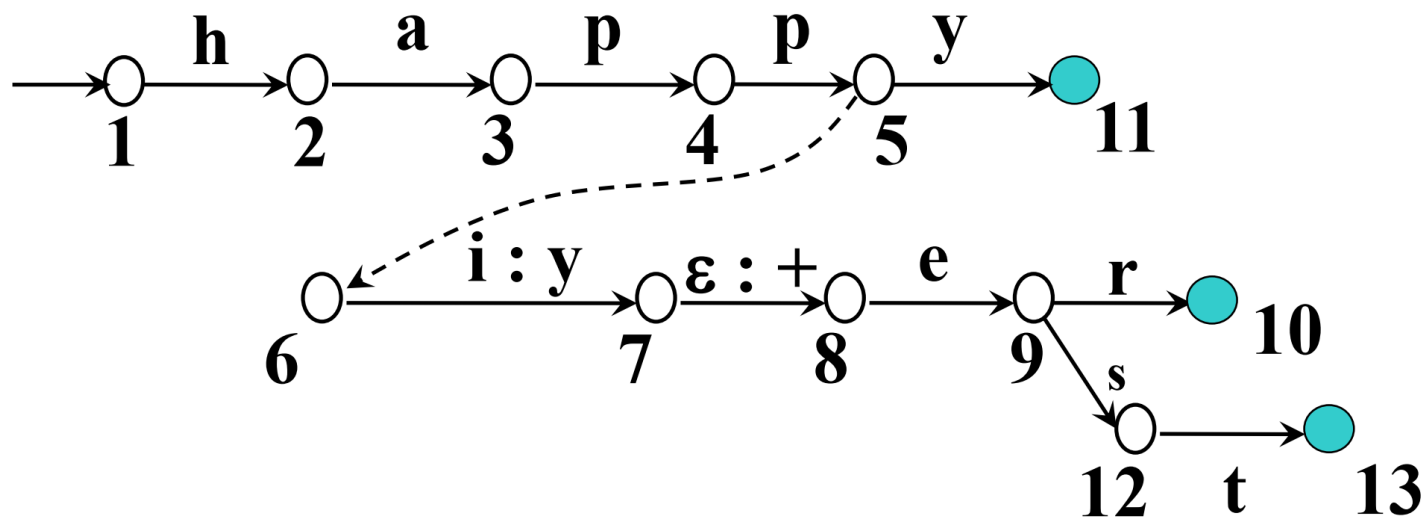
[Allen, 1995]

**说明：**在实际应用中，除了有限状态机以外，还常常使用有限状态转换机(finite state transducer, FST)的概念。粗略地讲，有限状态转换机与有限自动机(或有限状态机)的区别在于：FST在完成状态转移的同时产生一个输出，而 FA (或 FSM) 只实现状态转移，不产生任何输出。



## 3.4 有限自动机的应用

### ◆ 有限自动机用于英语单词形态分析



识别单词: happy, happier, happiest

可转换的形式: happier  $\rightarrow$  happy + er

happiest  $\rightarrow$  happy + est

## 3.4 有限自动机的应用

### ◆ 有限自动机用于英语单词形态分析

一般地，具有相同的前缀或词根，词缀不同的单词可以共用一个有限状态转移机，共享其中的某些状态节点。如：tie, ties, trap, traps, try, tries, to, torch, torches, toss, tosses等。

## 3.4 有限自动机的应用

### ◆ 有限自动机用于词性标注 [Roche, 1995]

- (a) The time **flies like** an arrow.
- (b) May I take a **can**?
- (c) 他这个人其实很**好**，就是**好**吹牛。

**词性 (part-of-speech) 消歧**

## 3.4 有限自动机的应用

除了前面提到的单词拼写检查、词法分析、词性标注等工作以外，有限状态自动机还广泛地应用于句法分析、短语识别、机器翻译和语音识别等很多方面。

# 本章小结

## ◆ 基本概念

- 树、字符串、字符串操作
- 正则表达式、有限状态图

## ◆ 形式文法

- 4 种形式文法的定义和相互关系
- 文法识别的语言(推导)

# 本章小结

## ◆ 关于自动机(Automata)

- 4种自动机的定义和区别
- 自动机识别语言的能力
- 文法与自动机的关系

## ◆ 有限自动机与状态转移机的应用

- 英文单词拼写检查
- 英文单词形态分析
- 词性标注

# 习题

3-1. 构造上下文无关文法用以产生：

(a) 有相同数目的 0 和 1 的所有 0, 1 符号串。

(b)  $\{a_1 a_2 \cdots a_n a_n \cdots a_2 a_1 \mid a_i \in \{0, 1\}, 1 \leq i \leq n\}$ 。

3-2. 有以下文法：  $G = (\{S, B, C\}, \{a, b, c\}, P, S)$ ，其中：

$P: S \rightarrow aSBC \mid abC \qquad CB \rightarrow BC$

$bB \rightarrow bb \qquad bC \rightarrow bc$

$cC \rightarrow cc$

求  $L(G) = ?$

3-3. 写一个程序模拟一个确定性的 PDA。

# 习题

3-4. 设文法  $G$  由如下规则定义：

$$S \rightarrow AB \qquad A \rightarrow Aa|bB$$

$$B \rightarrow a|Sb$$

给出下列句子形式的派生树：

$$(1) \text{ } baabaab \qquad (2) \text{ } bBABb$$

3-5. 写一个程序以正则文法  $G$  作为输入，构造  $G$  相应的有限自动机。

3-6. 实现编辑距离计算方法：对于任意给定的两个相似的英语单词计算出其编辑距离。



---

# 谢谢!