第三章 形式语言与自动机及其在NLP中的应用

No.95, Zhongguancun East Road
Beijing 100080, China

http://www.nlpr.ia.ac.cn
Tel. No.: +86-10-6255 4263

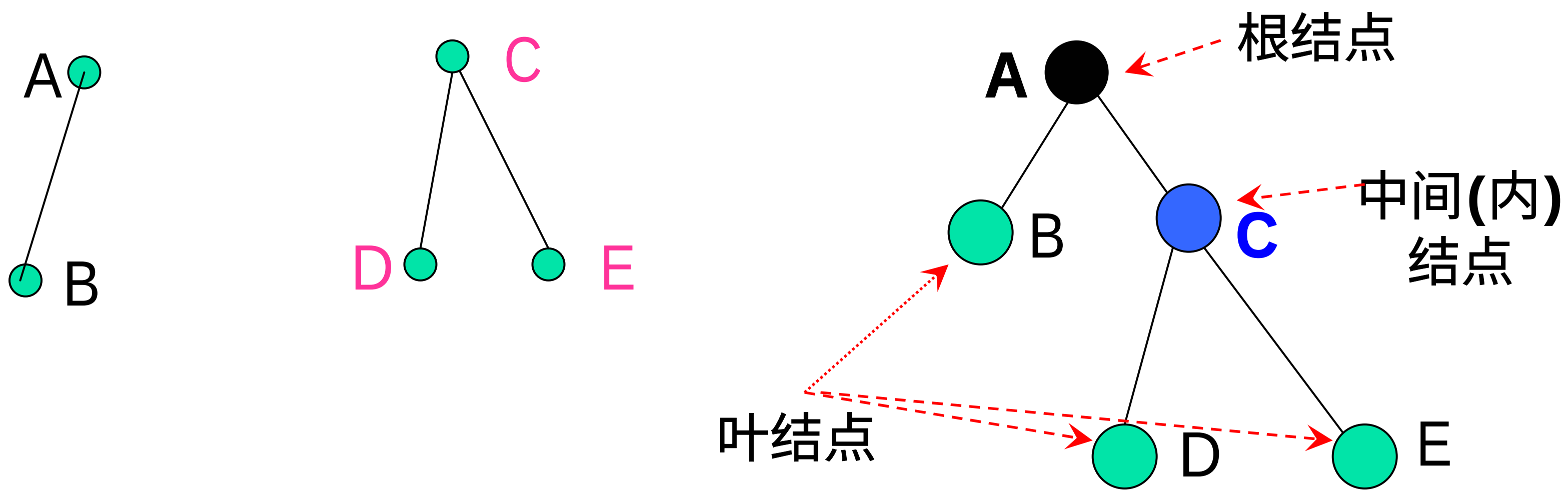
3.1 几个基本概念

- 树 (tree)：

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 几个基本概念



父结点： A是B、C结点的父结点；C是D、E 结点的父结点。

子结点：B、C是A结点的子结点；D、E 是 C 结点的子结点。

兄弟结点： B、C互为兄弟结点； D、E 互为兄弟结点。

3.1 几个基本概念

❑ 字符串 (string)

<u>字符串定义</u>：假定 Σ 是字符的有限集合，它的每一个元素称之为字符。由 Σ 中字符相连而成的有限序列被称之为Σ 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<u>空串</u>，记作 ε。

<u>符号串的长度</u>：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 Σ 上字符串的全体记为 Σ^*。

3.1 几个基本概念

❑ 字符串操作

假定 Σ 是字符的有限集合， x, y 是Σ上的符号串

(1) <u>字符串连接</u>：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： Σ={a, b, c}, x=ab, y=cba

那么，xy=abcba

3.1 几个基本概念

- 设x是符号串，把x自身连接n次得到的符号串，即 z=xx…x (n个x), 称为x的n次方幂，记作x^n。
- 注意：$x^0=\varepsilon$

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

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

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 几个基本概念

(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：

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

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

例如：设A={aa, bb}, B={cc, dd, ee}，则

AB={aacc, aadd, aaee, bbcc, bbdd, bbee}

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

3.1 几个基本概念

(3) 闭包

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

$V^+ = V^1 \cup V^2 \cup \ldots$ （称为V的正闭包）

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

例如：$V = \{a, b\}$

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

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

3.1 几个基本概念

- 正则表达式（简称正则式）：

正则式对应于Σ上的一些子集（正则集），并通过递归定义：

1）空集ϕ和空字符串ε是正则式，它们的正则集分别为ϕ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Σ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 几个基本概念

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：$U=\{\varepsilon, 1, 11, \ldots\}$

$xy^* = 01^*$ 也是正则式, 且它对应的正则集：

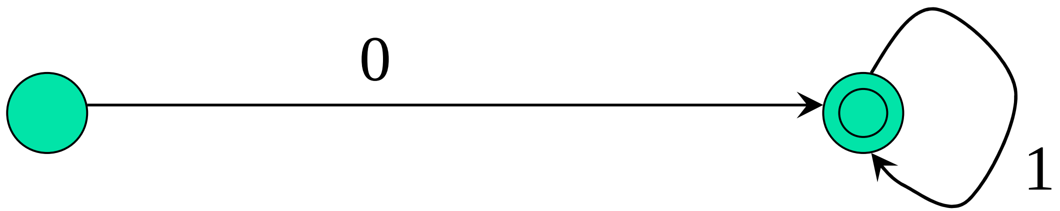
$$V=\{ 0, 01, 011, 0111, \ldots\}$$

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

3.1 几个基本概念

❑ 正则表达式与有限状态图

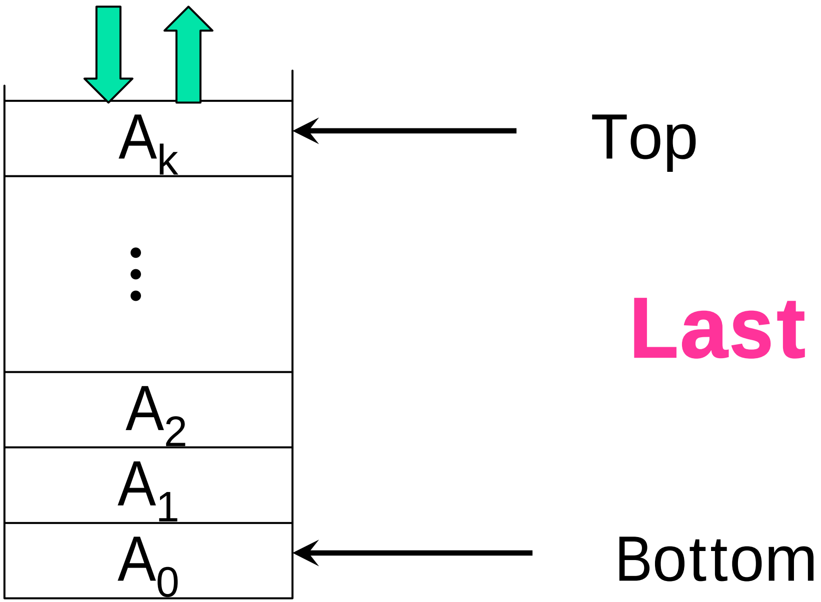
正则表达式可以用有向图表示，图的结点是状态，有一个起始结点和一个终止结点。起始结点只有出边，终止结点用双圆圈表示。边上的符号表示从一个状态到另一个状态结点允许出现的字符，这种图称之为有限状态图。正则式01* 对应的有限状态图为：



3.1 几个基本概念

- 栈 (stack)

栈是一种线性表，$A = A_0, A_1, \ldots, A_k$. 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。



Last in, first out.

3.2 形式语言

- 关于语言的定义

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

语言可以被看成一个抽象的数学系统。

—吴蔚天(1994)

3.2 形式语言

- 语言描述的三种途径
  - 穷举法 —— 只适合句子数目有效的语言。
  - 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。

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，α，β 均为字符串。顾名思义：字符串 α 可以被改写成 β。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)；Σ 是终结符的有限集合，$N \cap \Sigma = \Phi$；$V = N \cup \Sigma$ 称总词汇表；P 是一组重写规则的有限集合：$P=\{\alpha \to \beta\}$, 其中，α，β 是 V 中元素构成的串，但 α 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: $S \to 0\ A\ 1$　　$0\ A \to 00A1$　　$A \to 1$

3.2 形式语言

❑ 推导的定义

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

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

3.2 形式语言

用 $\overset{+}{\underset{G}{\Rightarrow}}$（按非平凡方式派生）表示 $\underset{G}{\Rightarrow}$ 的传递闭包，也就是 $(N \cup \Sigma)^*$上的符号串 ξ_i 到 ξ_{i+1} 的 n ($n \geq 1$) 步推导或派生。

用 $\overset{*}{\underset{G}{\Rightarrow}}$ (派生)表示 $\underset{G}{\Rightarrow}$ 的自反和传递闭包，即由$(N \cup \Sigma)^*$上的符号串 ξ_i 到 ξ_{i+1} 经过n ($n \geq 0$)步推导或派生。

如果清楚某个推导是文法 G 所产生的，则符号 $\overset{*}{\underset{G}{\Rightarrow}}$ 或 $\overset{+}{\underset{G}{\Rightarrow}}$ 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：$G = (\{E, T, F\}, \{a, +, *, (,)\}, P, E)$

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

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

字符串 a+a*a 的两种推导过程：

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

$$\Rightarrow a + F*F \Rightarrow a + a*F \Rightarrow a + a*a$$ （<u>最左推导</u>）

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

$$\Rightarrow T + a*a \Rightarrow F + a*a \Rightarrow a + a*a$$ （<u>最右推导</u>）

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：$L(G) = \{x \mid x \in \Sigma, S \overset{*}{\underset{G}{\Rightarrow}} x\}$

3.2 形式语言

❑ 正则文法

如果文法 $G=(N, \Sigma, P, S)$ 的 P 中的规则满足如下形式：$A \to B\,x$，或 $A \to x$，其中 $A, B \in N$，$x \in \Sigma$，则称该文法为正则文法（简写为 FSG）或称3型文法。（左线性正则文法）（如果 $A \to x\,B$，则该文法称为右线性正则文法。）

3.2 形式语言

例 3.2：$G = (N, \Sigma, P, S)$,

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

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

(c) $A \rightarrow b\ b\ B$ (d) $B \rightarrow b\ B$

(e) $B \rightarrow b$

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

3.2 形式语言

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

如果 P 中的规则满足如下形式：$A \rightarrow \alpha$，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： $G = (N, \Sigma, P, S)$,

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

P: (a) $S \rightarrow A\ B\ C$ (b) $A \rightarrow a\ A \mid a$

(c) $B \rightarrow b\ B \mid b$ (d) $C \rightarrow B\ A \mid c$

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

3.2 形式语言

- 上下文有关文法

（CSG, context-sensitive grammar）

如果 P 中的规则满足如下形式: $\alpha A \beta \rightarrow \alpha \gamma \beta$, 其中 $A \in N$， $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$，且 γ 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。

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

3.2 形式语言

例 3.4： $G = (N, \Sigma, P, S)$

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

P: (a) $S \rightarrow A\ B\ C$ (b) $A \rightarrow a\ A \mid a$

(c) $B \rightarrow b\ B \mid b$ (d) $B\ \mathbf{C} \rightarrow B\ \mathbf{c}\ \mathbf{c}$

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

3.2 形式语言

- 无约束文法（无限制重写系统）

　　如果 P 中的规则满足如下形式：$\alpha \to \beta$，α，β 是字符串，则称 G 为无约束文法，或称 0 型文法。

3.2 形式语言

显然，每一个正则文法都是上下文无关文法，每一个上下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：

$$\mathbf{L(G0) \supseteq L(G1) \supseteq L(G2) \supseteq L(G3)}$$

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

$A \rightarrow a$　$B \rightarrow bS$　$B \rightarrow aBB$　$B \rightarrow b$

G 为上下文无关文法。

L(G) = {等数量的a和b构成的链}

3.2 形式语言

- CFG 产生的语言句子的派生树表示

CFG G=(N, Σ, P, S) 产生一个句子的派生树由如下步骤构成：

(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 $A \in N$。

3.2 形式语言

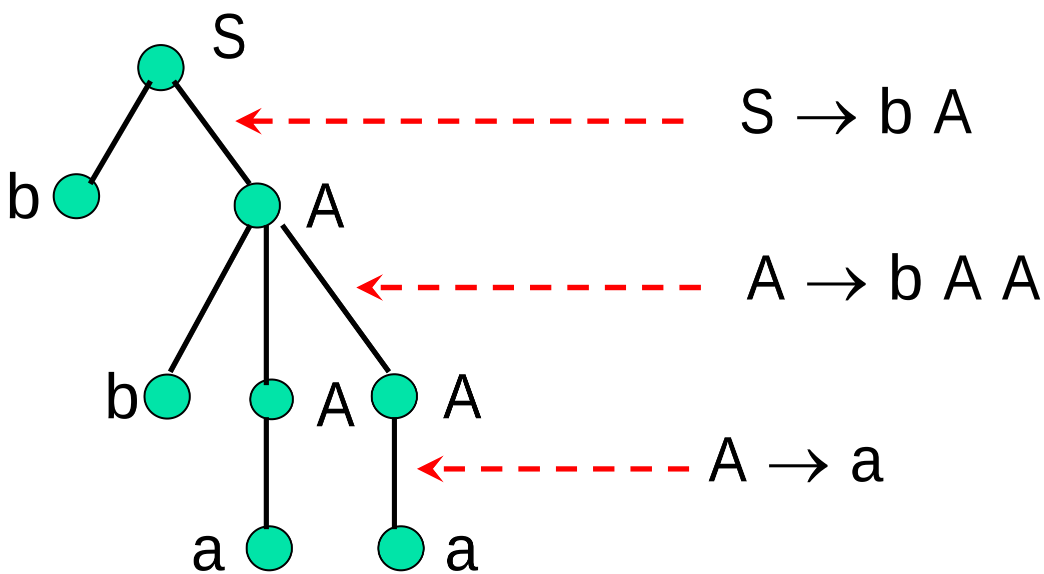
(3) 如果一个节点的标记为 A，它的 k ($k > 0$) 个直接后裔节点按从左到右的次序依次标记为 $A_1, A_2, \ldots, A_k$，则 $A \rightarrow A_1, A_2, \ldots, A_k$ 一定是 P 中的一个产生式。

3.2 形式语言

例如，$G = (\{S, A\}, \{a, b\}, P, S)$

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

G 所产生的一个句子 bbaa 可以由下面的 生树表示：



3.2 形式语言

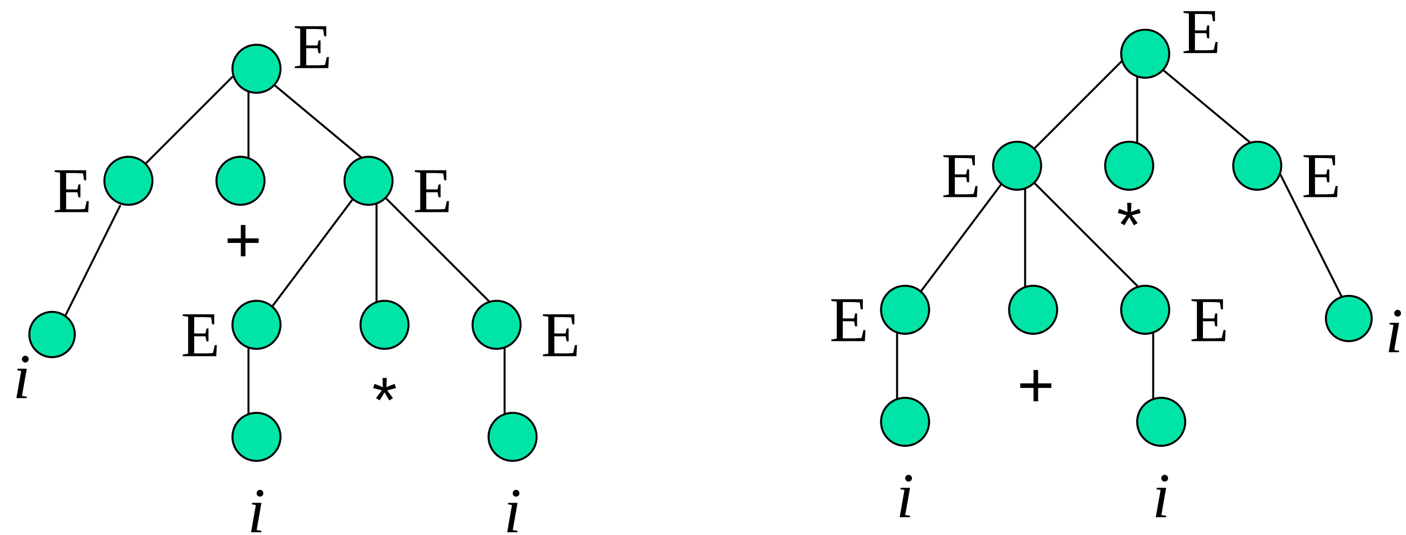
- 上下文无关文法的二义性

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。

3.2 形式语言

例： G(E): $E \rightarrow E + E \mid E * E \mid (E) \mid E - E \mid i$

对于句子 $i + i * i$ 有两棵对应的分析树。



3.2 形式语言

- 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 有限自动机与正则文法

❑ 确定的有限自动机 (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，Σ 是输入符号的有穷集合；

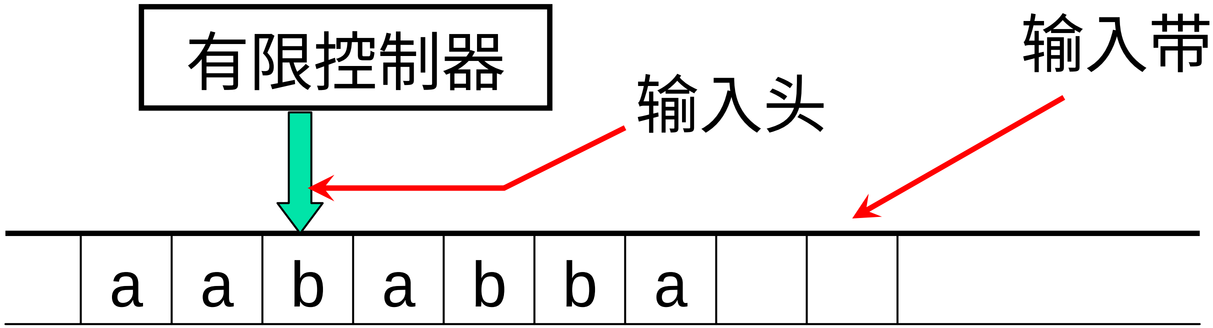
Q 是状态的有限集合；　$q_0 \in Q$ 是初始状态；

F 是终止状态集合，$F \subseteq Q$；

δ 是 Q 与 Σ 的直积 $Q \times \Sigma$ 到 Q (下一个状态) 的映射。它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 有限自动机与正则文法

- DFA 示意图

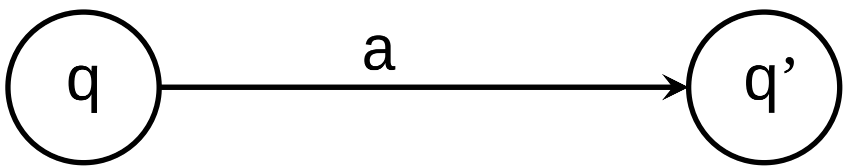


处在状态 $q \in Q$ 中的有限控制器从左到右依次从输入带上读入字符。开始时有限控制器处在状态 q_0，并注视 Σ^* 中一个链的最左符号。映射 $\delta(q, a) = q'$ $(q, q' \in Q, a \in \Sigma)$ 表示在状态 q 时，若输入符号为 a，则自动机进入状态 q' 并且将输入头向右移动一个字符。

3.3 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。

3.3 有限自动机与正则文法

❑ DFA 定义的语言

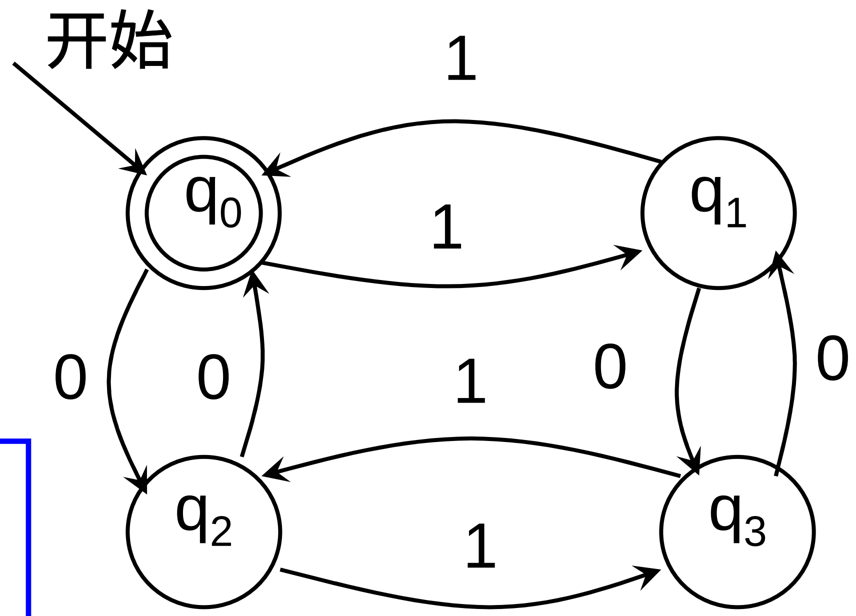
如果一个句子 x 使得有限自动机 M 有 $\delta(q_0, x) = p$, $p \in F$，那么，称句子 x 被 M 接受。由 M 定义的语言 T(M) 就是被 M 接受的句子的全集。即：

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

3.3 有限自动机与正则文法

❑ DFA 定义的语言

例 3.6：



链 x = 110101 被 M 接受.

T(M) = {含偶数个0和偶数个1的链}

3.3 有限自动机与正则文法

不确定的有限自动机 (Non-definite Automata, NFA)

不确定的有限自动机 M 是一个五元组

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

其中，Σ 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，$F \subseteq Q$；

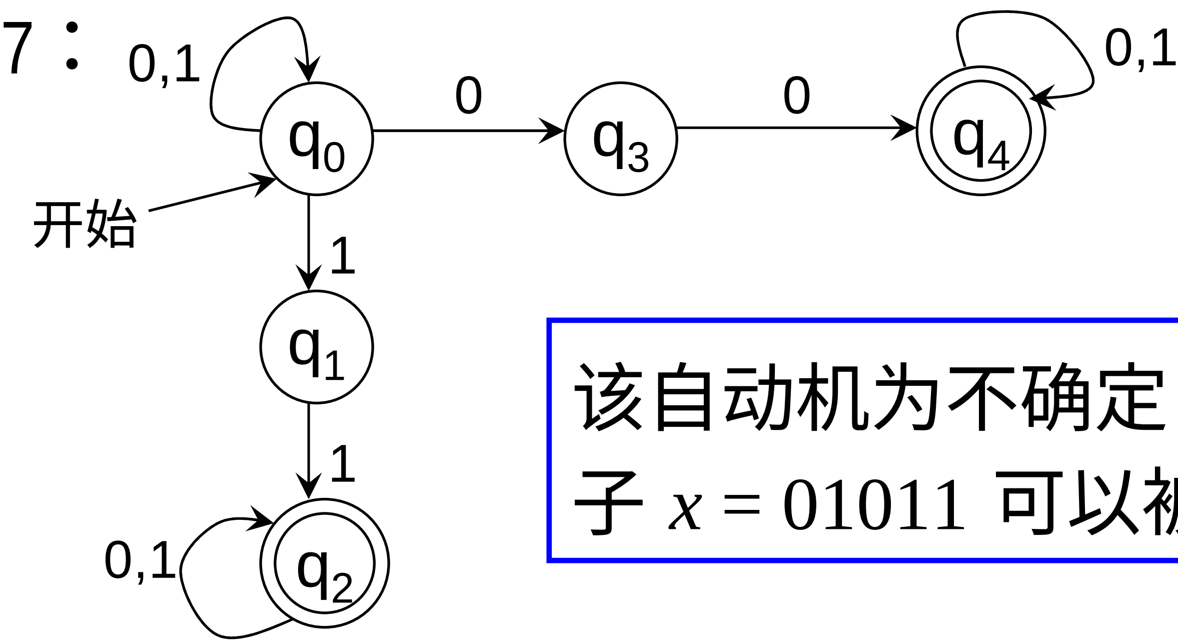
δ 是Q与Σ的直积 $Q \times \Sigma$ 到Q的幂集 2^Q 的映射。

3.3 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 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 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 有限自动机与正则文法

- 由 G 构造 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 $T \in \delta(B, a)$。

3.3 有限自动机与正则文法

(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 $\delta(T, a) = \phi$。

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。

3.3 有限自动机与正则文法

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

$\Sigma = V_T = \{a, b\}$，$Q = V_N \cup \{ T \} = \{S, B, T\}$，$q_0 = S$，$F = \{ T \}$

(2) 映射 δ为：$\delta(S, a) = \{B\}$ (因为有规则 $S \rightarrow aB$)

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

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

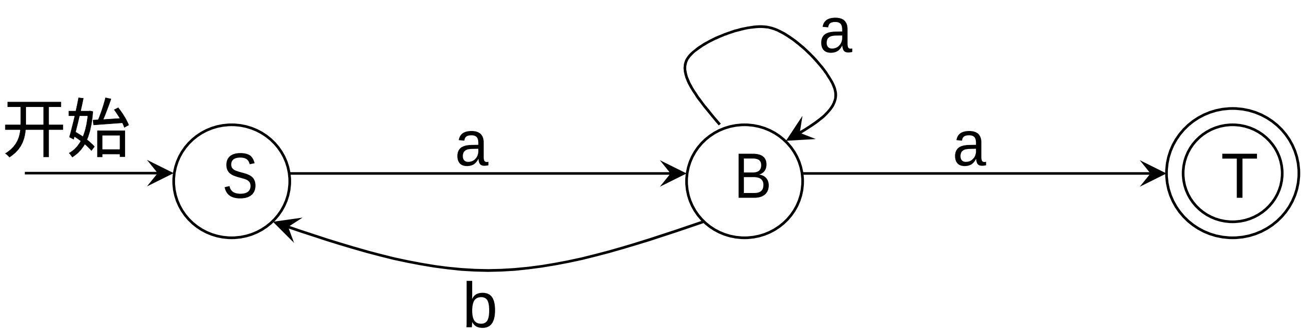
$\delta(B, b) = \{S\}$ (因为有 $B \rightarrow bS$)

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

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

3.3 有限自动机与正则文法

等价的 NFA 的状态变换图为：



3.3 有限自动机与正则文法

定理 3.2：若 $M=(\Sigma, Q, \delta, q_0, F)$ 是一个有限自动机，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 $B \to aC$；

(3) 如果 $C \in \delta(B, a)$，$C \in F$，则在 P 中有产生式 $B \to a$。

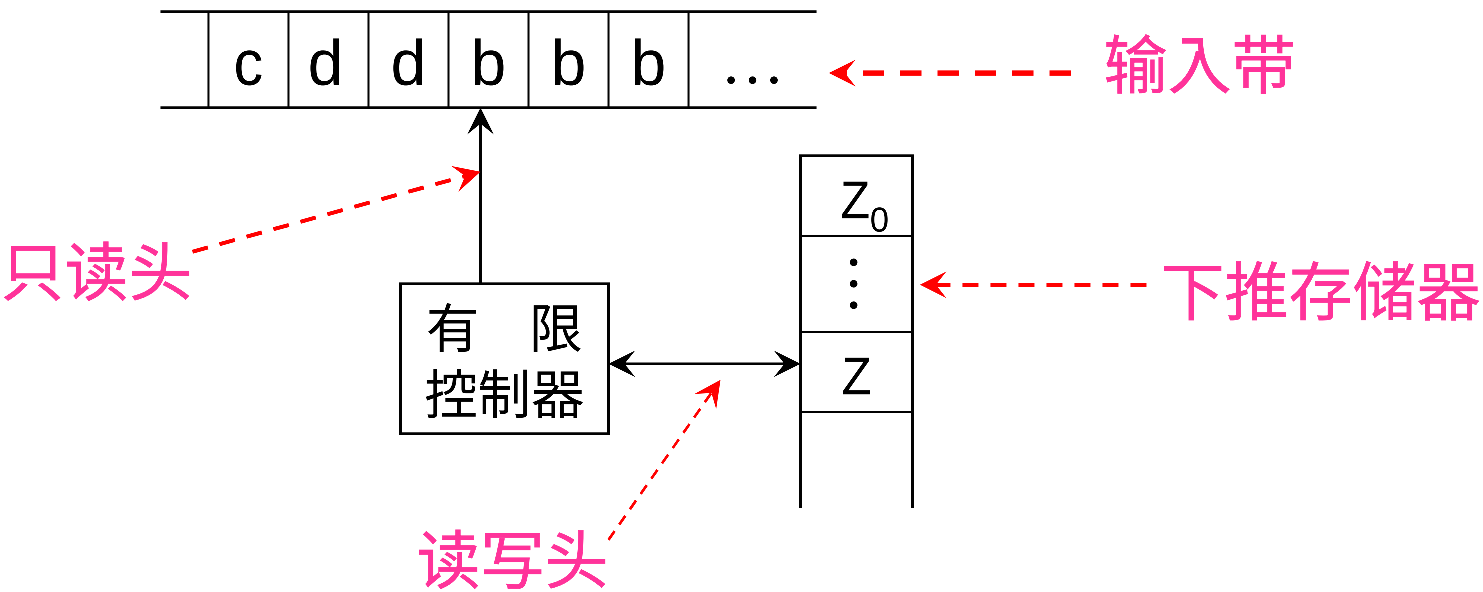
3.3 有限自动机与正则文法

结论：对于任意一正则文法，总可以构造一个识别器 —— DFA。

3.4 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：



3.4 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，Σ 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；

Γ 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开始符号;

F 是终止状态集合，$F \subseteq Q$；

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

3.4 CFG 与下推自动机

❑ 映射关系 δ 的解释

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

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

该映射的意思是：当PDA处于状态 q，面临输入符号 a时，自动机将进入到 q_i, $i = 1, 2, \ldots, m$ 状态，并以 γ_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 γ_i 取代时，γ_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。

3.4 CFG 与下推自动机

特殊情况下，$\delta(q, \varepsilon, Z)=\{(q_1, \gamma_1), (q_2, \gamma_2), \ldots, (q_m, \gamma_m)\}$ 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 “ε移动”。

3.4 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 $a\in(\sum\cup\{\varepsilon\}), \beta\in\Gamma^*, Z\in\Gamma$

如果 $(q',\beta)\in\delta(q,a,Z), q',q\in Q$，则表达式

$$a\text{: } (q, Z\gamma) \vdash_{M} (q', \beta\gamma)$$

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

$a\text{: } (q, Z\gamma) \vdash_{M} (q', \beta\gamma)$ 为合法转移。零次或多次合法转移记为： $a\text{: } (q, Z\gamma) \vdash_{M}^{*} (q', \beta\gamma)$ M 可以省略不写。

3.4 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：

$$T(M) = \{x | x: (q_0, Z_0) \overset{*}{\underset{M}{\vdash}} (q, \gamma), \gamma \in \Gamma^*, q \in F\}$$

3.4 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 $L=\{wcw^R | w\in\{a, b\}^*\}$，其中，$Q=\{0, 1\}$, $\Sigma=\{a, b, c\}$, $\Gamma = \{A, B\}$，$q_0 = 0$, $Z_0 = \#$, $F = \{1\}$, δ 定义如下：

1) $\delta(0, a, \varepsilon) \vdash_M \{(0, A)\}$　　2) $\delta(0, b, \varepsilon) \vdash_M \{(0, B)\}$

3) $\delta(0, c, \varepsilon) \vdash_M \{(1, \varepsilon)\}$　　4) $\delta(1, a, A) \vdash_M \{(1, \varepsilon)\}$

5) $\delta(1, b, B) \vdash_M \{(1, \varepsilon)\}$

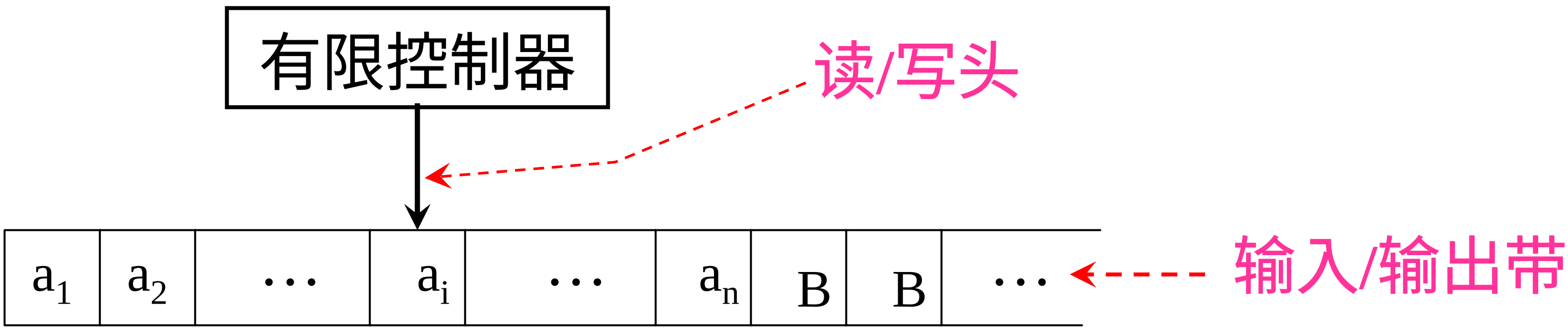
3.4 CFG 与下推自动机

对于输入 abbcbba 下推自动机 M 的处理步骤为：

状态	输入	栈	运用的规则
0	abbcbba	#	-
0	bbcbba	A#	1
0	bcbba	BA#	2
0	cbba	BBA#	2
1	bba	BBA#	3
1	ba	BA#	5
1	a	A#	5
1	ε	#	4

3.5 图灵机和线性带限自动机

❑ 图灵机的理解



给定字符串 α 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 α 的最左字符。根据转移函数 δ 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。

3.5 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。

3.5 图灵机和线性带限自动机

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，Σ 是输入/输出带上字符的有穷集合；

Q是状态的有限集合；$\Gamma = \Sigma \cup \{B\}$, B为空白字符;

$q_0 \in Q$ 是初始状态；　F 是终止状态集，$F \subseteq Q$；

δ 是从 $Q \times \Sigma$ 到 $Q \times \Sigma \times \{R, L, S\}$ 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。

3.5 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, α, i)，其中，$q \in Q$, α 是字符串，且 $\alpha \in \Sigma$， i 是整数，表示 T 的读/写头到 α 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 $(q, A_1A_2\ldots A_n, i)$, $1 \le i \le n+1$ 是当前 T 的格局，

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

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

即 T 的读/写头在 i 位置写入符号 X，并将读/写头向右移动一个位置。

3.5 图灵机和线性带限自动机

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

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

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

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

而如果 $\delta(q, B) = (p, X, L)$, 则：

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

3.5 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：$X \overset{*}{\underset{T}{\vdash}} Y$

由图灵机T 所接受的语言定义为：

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

3.5 图灵机和线性带限自动机

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

定理 3.3 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 3.4 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。

3.5 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。

3.5 图灵机和线性带限自动机

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，Σ 是输入/输出带上字符的有穷集合；

Q 是状态的有限集合；$q_0 \in Q$ 是初始状态；

Γ 是输入带上符号的有穷集；

F 是终止状态集合，$F \subseteq Q$；

δ 是从 $Q \times \Gamma$ 到 $Q \times \Gamma \times \{R, L\}$ 子集的映射。Σ 包括两个特殊符号 # 和 \$，分别表示输入链的左端和右端结束标志。

3.5 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 $\vdash_M$ 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：

$$L(M) = \{\alpha | \alpha \in (\Sigma\text{-}\{\#, \$\})^*, (q_0, \#\alpha\$, 1) \vdash_M^* (q, \beta, i), q \in F, \beta \in \Gamma^* \}$$

3.5 图灵机和线性带限自动机

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。

- <u>定理 3.5</u>：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。

3.6 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的差异：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。

3.6 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，图灵机等基于 0 型文法。

3.7 有限自动机的应用

- 有限自动机用于英语单词拼写检查

[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 $ed(X[m], Y[n])$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 (recoginze, recognize) = 1

ed (sailn, failing) = 3

3.7 有限自动机的应用

一个确定的有限状态机 R 定义为：

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

其中，Q 表示状态集；A 表示输入字符集；

$$\delta : Q \times A \to Q$$

$q_0 \in Q$ 为起始状态；

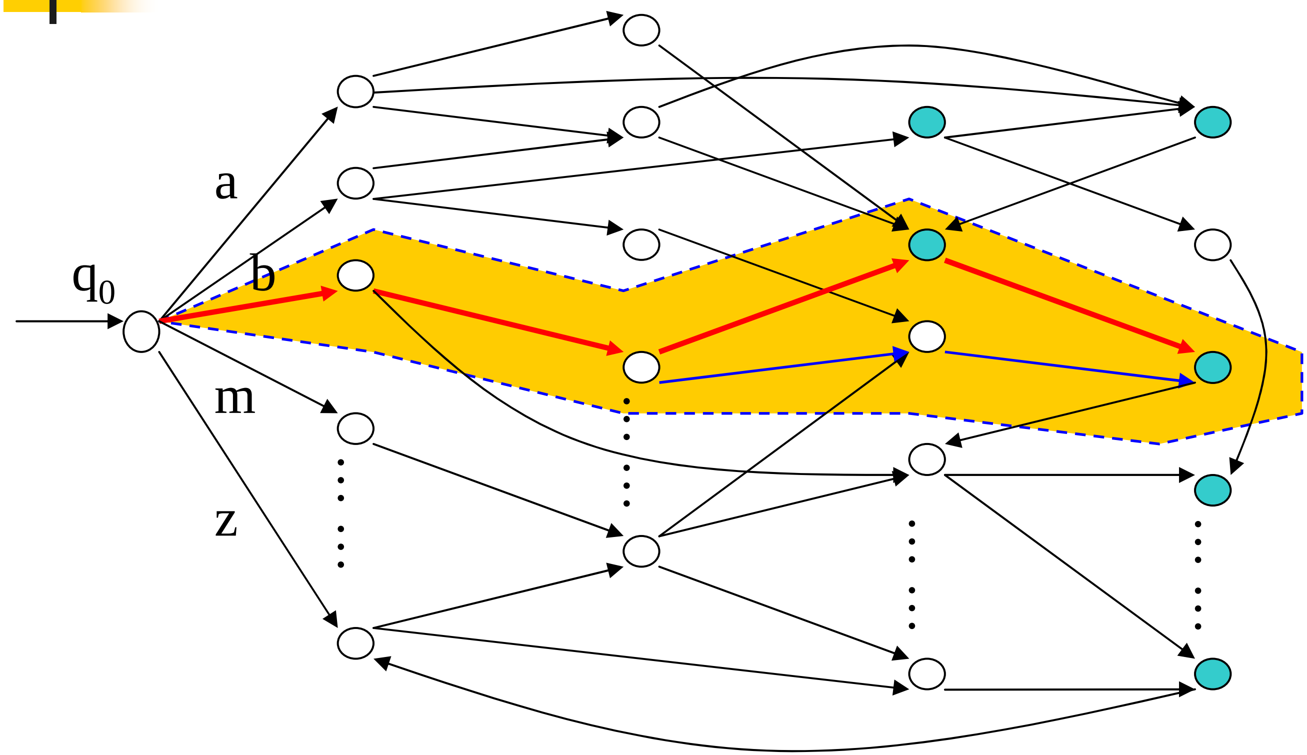
$F \subseteq Q$ 为终止状态集；

3.7 有限自动机的应用

如果$L \subseteq A^*$表示有限状态机 R 接受的语言，$t > 0$ 为编辑距离的阈值，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：

$$C = \{Y[n] \mid Y[n] \in L \quad and \quad ed(X[m], Y[n]) \le t\}$$

3.7 有限自动机的应用



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)。

3.7 有限自动机的应用

即： $cuted(X[m],Y[n]) = \min\limits_{l \le i \le u}\{ed(X[i],Y[n])\}$

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

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

3.7 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

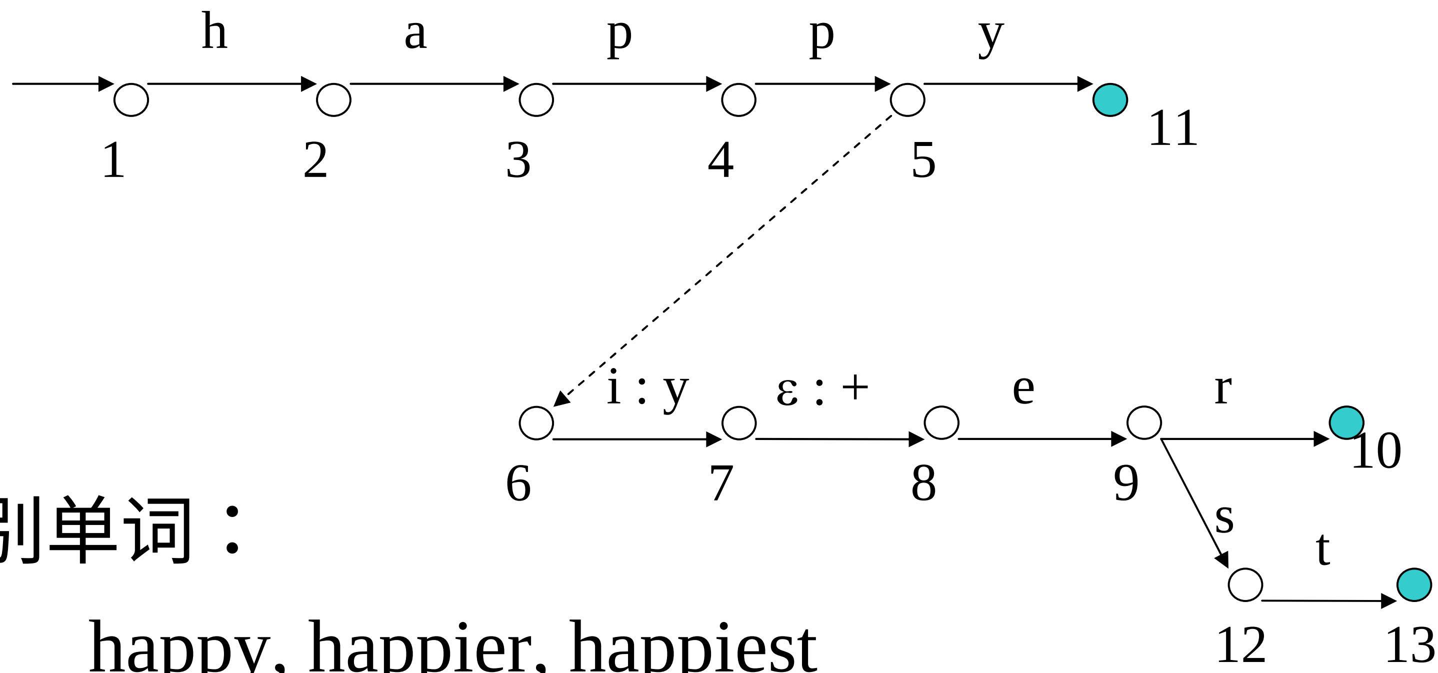
happy: happier, happiest

seed ?

3.7 有限自动机的应用

<u>**说明：**</u>在实际应用中，除了有限状态机以外，我们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。

3.7 有限自动机的应用



识别单词：

happy, happier, happiest

可转换的形式：

happier → happy + er　　happiest →happy + est

3.7 有限自动机的应用

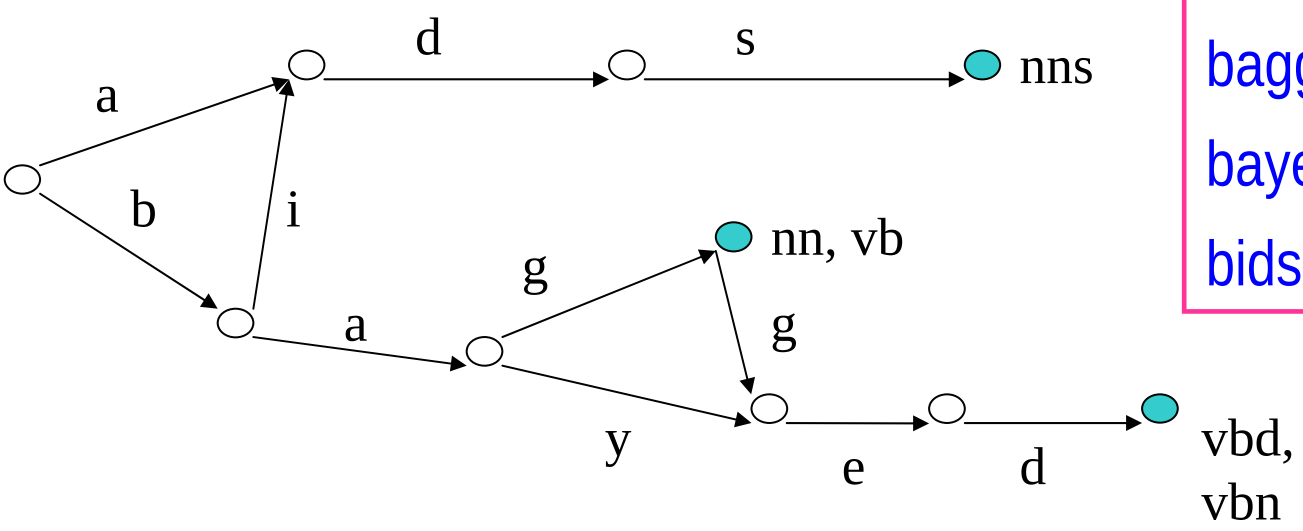
一般地，具有相同的前缀或词根，词缀不同的单词可以共用一个有限状态转移机，共享其中的某些状态节点。如：tie, ties, trap, traps, try, tries, to, torch, torches, toss, tosses 等。

3.7 有限自动机的应用

- 有限自动机用于词性标注 [Roche, 1995]

(1) 词汇标注 (lexical tagger)

—DAG: directed acyclic graph



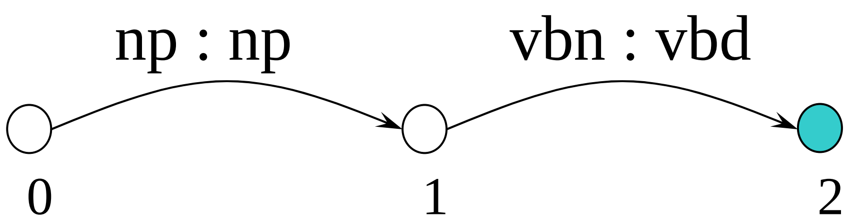
ads	nns
bag	nn, vb
bagged	vbn, vbd
bayed	vbn, vbd
bids	nns

3.7 有限自动机的应用

(2) FSTs: 词性上下文约束规则

A B *PRETAG* C

如果前一个词性标注是C，那么，将A 转换成 B。



本章小结

- 几个基本概念
  - 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) $\{a_1a_2\ldots a_na_n\ldots a_2a_1 | a_i \in \{0,1\},\ 1\le i \le n\}$。

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

P: S → aSBC | abC　　　CB → BC

bB → bb　　　bC → bc

cC → cc

求 L(G)=?

习题

3-3. 设文法 G 由如下规则定义：

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

$$B \rightarrow a|Sb$$

给出下列句子形式的派生树:

(1) baabaab　　(2) bBABb

3-4. 写一个程序模拟一个确定性的 PDA。

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。

Thanks

谢谢！