

自然语言处理

赵云蒙

华东理工大学 信息科学与工程学院

能源化工过程智能制造教育部重点实验室

2022-2023 第一学期

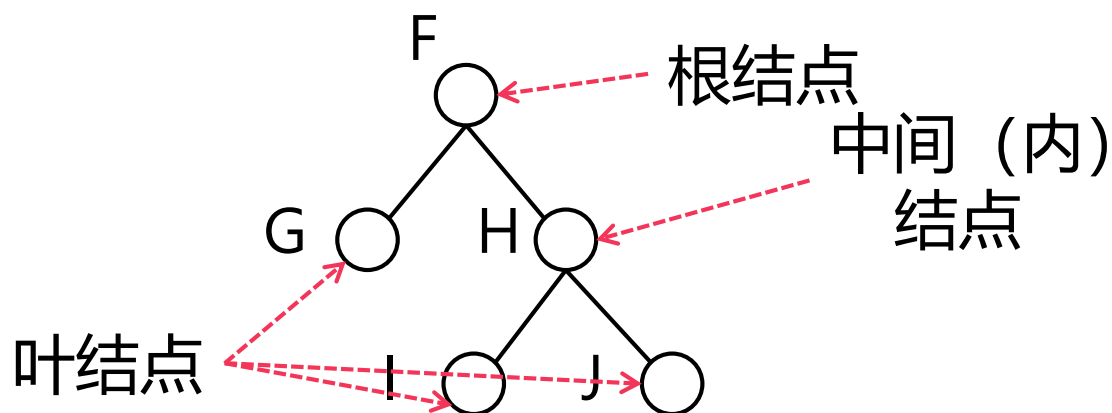
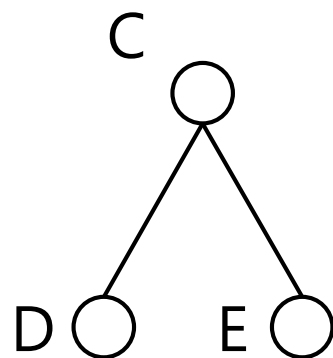
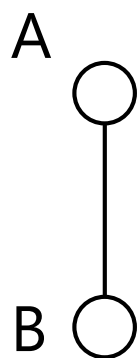
第三章 形式语言与自动机 及其在NLP中的应用

3.1 基本概念

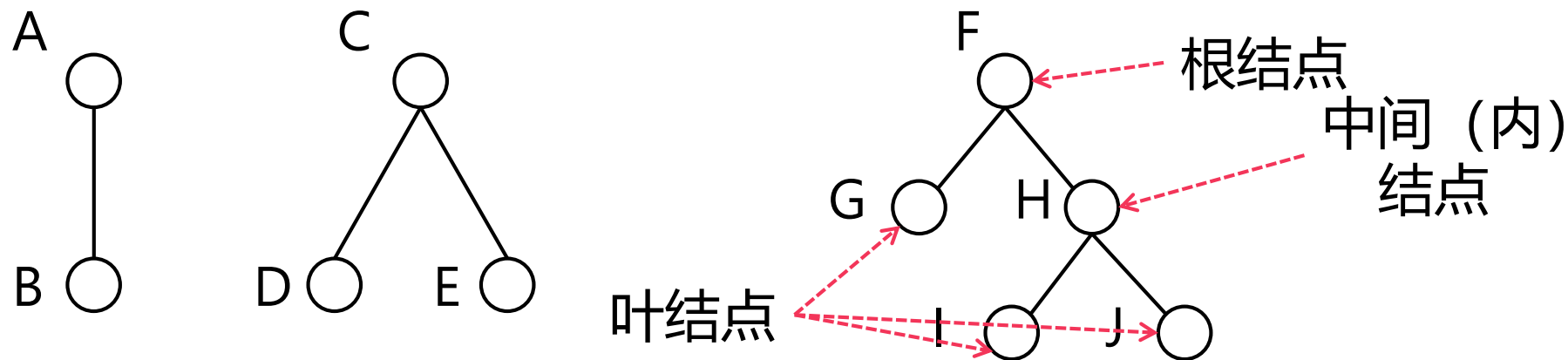
3.1 基本概念

★ 树(tree):

- ✦ 树是一种数据结构。
- ✦ 一个连通的无回路的无向图成为**树**（或**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**，这个特别标记的结点被称之为**根结点**。



3.1 基本概念



- ✦ **父结点:** A是B的父结点。
- ✦ **子结点:** D、E是C的子结点。
- ✦ **兄弟结点:** D和E互为兄弟结点。

3.1 基本概念

★ 字符串(string)

- ✦ **定义**：假设 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成为**空串**，记为 ε 。
- ✦ **字符串的长度**：字符串中符号的个数。字符串 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 基本概念

★ 字符串的操作:

① **字符串连接**: 假定 Σ 是字符串的有限集合, x, y 是 Σ 上的符号串, 则把 y 的各个符号依次写在 x 字符串之后得到的字符串成为 x 与 y 的连接, 记作 xy 。

✦ 例如, $\Sigma = \{a, b, c\}, x = ab, y = cba$, 那么, $xy = abcba$

✦ 如果 x 是字符串, 把 x 自身连接 $n(n \geq 0)$ 次得到字符串 $z = \overbrace{xx \cdots x}^n$, 称为 x 的 n 次方幂, 记作 x^n 。

✦ 当 $n=0$ 时, $x^0 = \{\varepsilon\}$

✦ 当 $n \geq 1$ 时, $x^n = xx^{n-1} = x^{n-1}x$

3.1 基本概念

- ② **字符串集合的乘积**：设 A, B 是字符表 Σ 上字符串的集合，则 A 和 B 的乘积定义为：

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

其中， $A^0 = \{\varepsilon\}$ 。当 $n \geq 1$ 时， $A^n = A^{n-1}A = AA^{n-1}$

- ★ 例：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

③ **闭包**：字符表 Σ 上字符串集合 V 的闭包定义为：

$$V^* = V^0 \cup V^1 \cup V^2 \dots$$

$$V^+ = V^1 \cup V^2 \dots$$

V 的正闭包

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

★ 例： $V = \{a, b\}$, 则

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

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

3.1 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 \emptyset 和空字符串 ε 是正则式，它们的正则集分别为 \emptyset 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 是上 Σ 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 $U|V, U \cdot V$ 和 U^* 。

✦ 假设 $\Sigma = \{0,1\}$ ，如果 $x = 0, y = 1$ ，那么：

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

3.1 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：

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

✦ $xy^* = 01^*$ 也是正则式，且对应的正则集为：

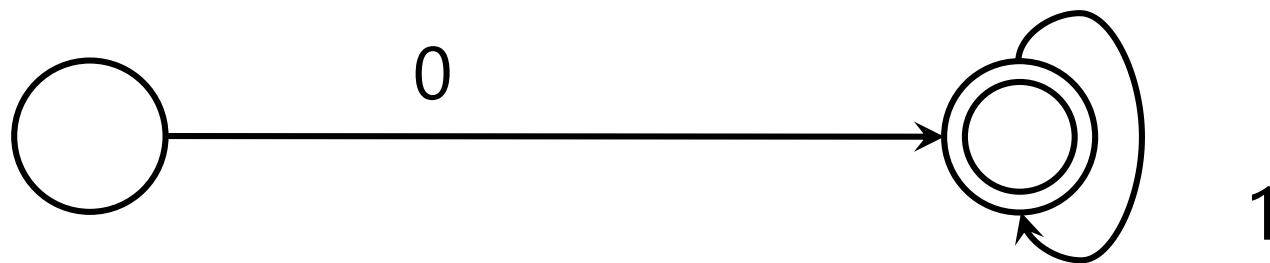
$$V = \{0, 01, 011, 0111, \dots\}$$

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

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限状态图表示，图的结点时状态，有一个起始节点和一个终止结点。起始节点只有出边，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：



3.2 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。

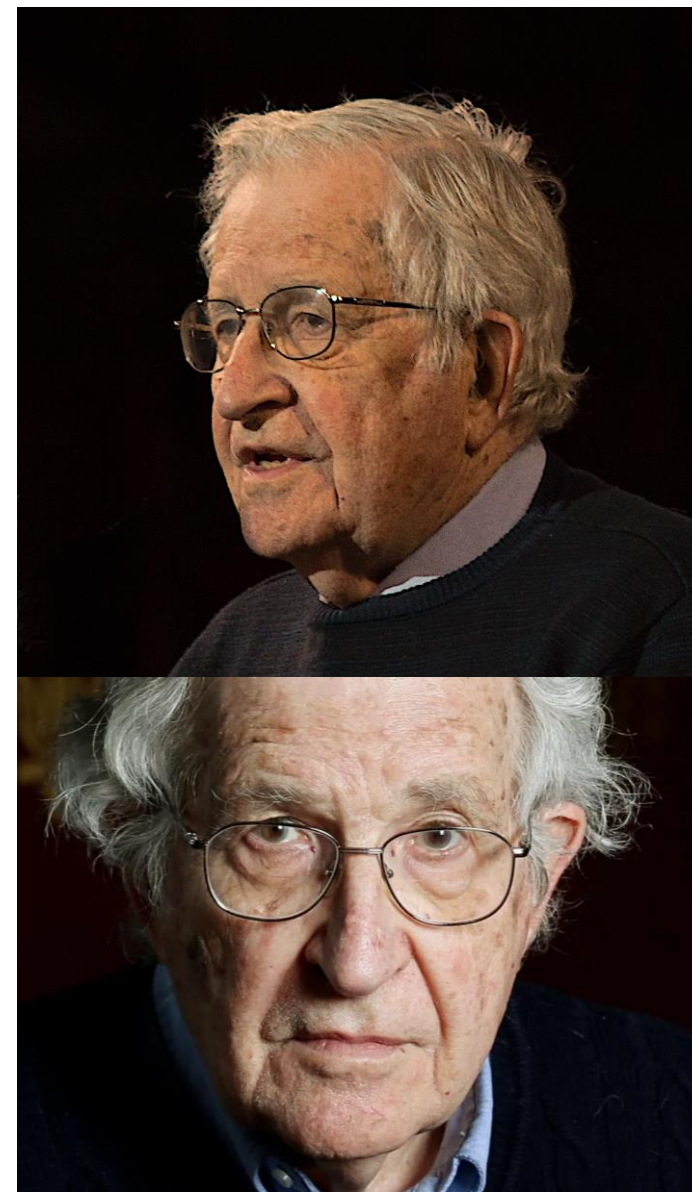
✦ —吴蔚天，1994

✦ 按照一定规律构成的句子和符号串的有限或无限的集合。

✦ —N. Chomsky

★ 诺姆·乔姆斯基(Noam Chomsky)

- ✦ 1928年12月生于美国费城
- ✦ 1944年(16岁) 进入宾州大学学习哲学、语言学和数学
- ✦ 1949年获学士学位、1951年获硕士学位
- ✦ 1952 起在哈佛大学认知研究中心研究员, 后来获宾州大学博士学位
- ✦ 1957年(29岁) 麻省理工大学副教授, 32岁成为现代语言学教授、47岁终生教授



★ 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。

★ 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。

★ 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, $N \cap \Sigma = \emptyset$; $V = N \cup \Sigma$ 称总词汇表; P 是一组重写规则的有限集合: $P = \{\alpha \rightarrow \beta\}$, 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: $G = (\{A, S\}, \{0, 1\}, P, S)$

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

★ 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 $(N \cup \Sigma)^*$ 中的字符串, 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

- ✦ 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 ξ_i 到 ξ_{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 ξ_i 到 ξ_{i+1} 经过 $n(n \geq 0)$ 步推导或派生。
- ✦ 如果清楚某个推导文法 G 所产生的, 则符号 $\xRightarrow{+}_G$ 或中 $\xRightarrow{*}_G$ 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:** $G = (\{E, T, F\}, \{a, +, *, (,)\}, P, E)$

✦ $P: E \rightarrow E + T \mid T \quad T \rightarrow T * F \mid F \quad F \rightarrow (E) \mid a$

✦ 字符串 $a + a * a$ 的两种推导过程:

★ E

★ 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成为 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:
- ✦
$$L(G) = \{x | x \in \Sigma, S \xRightarrow[G]{+} x\}$$

★ 正则文法(Regular grammar)

- ★ 如果文法 $G = (N, \Sigma, P, S)$ 的 P 中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 $A \rightarrow xB$, 则该文法称为**右线性正则文法**。

3.2 形式语言

★ 例3-2: $G = (N, \Sigma, P, S)$, $N = \{S, A, B\}$, $\Sigma = \{a, b\}$,
★ $P: (a) S \rightarrow aA, (b) A \rightarrow aA, (c) A \rightarrow bbB, (d) B \rightarrow$
 $bB, (e) B \rightarrow b$

★ $L(G) = ?$

$A \rightarrow bB',$
 $B' \rightarrow bbB$

★ 解:

★ $S \Rightarrow aA \Rightarrow abbB \Rightarrow abbb$

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

★ 上下文无关文法(context-free grammar, CFG)

✦ 如果 P 中的规则满足如下形式: $A \rightarrow \alpha$, 其中 $A \in N$, $\alpha \in (N \cup \Sigma)^*$, 则该文法为**上下文无关文法**或称**2型文法**。

★ **例3-3**: $G = (N, \Sigma, P, S)$, $N = \{S, A, B, C\}$, $\Sigma = \{a, b, c\}$,

✦ P : (a) $S \rightarrow ABC$, (b) $A \rightarrow aA|a$, (c) $B \rightarrow bB|b$, (d) $C \rightarrow BA|c$

✦ $L(G) = ?$

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

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

★ 上下文有关文法(context-sensitive grammar, CSG)

- ✦ 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, 则称该文发为**上下文有关文法(CFG)**或称**1型文法**。
- 另一种定义: *if* $x \rightarrow y, x \in (N \cup E)^+, y \in (N \cup \Sigma)^*$, 并且 $|y| \geq |x|$

3.2 形式语言

- ★ **例3-4:** $G = (N, \Sigma, P, S)$, $N = \{S, A, B, C\}$, $\Sigma = \{a, b, c\}$,
- ✦ $P: (a) S \rightarrow ABC, (b) A \rightarrow aA|a, (c) B \rightarrow bB|b, (d) BC \rightarrow Bcc$
 - ✦ $L(G) = ?$
 - ✦ **解:** $L(G) = \{a^n b^m c^2\}, n \geq 1, m \geq 1$

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： $\alpha \rightarrow \beta$ ， α ， β 是字符串，则称 G 为**无约束文法**，或称**0型文法**。

显然，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：

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

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

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

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

G 为上下文无关文法。

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

★ CFG 产生的语言句子的派生树表示

CFG $G = (N, \Sigma, P, S)$ 产生一个句子的派生树由如下步骤构成:

- (1) 对于 $\forall x \in N \cup \Sigma$ 给一个标记作为结点, 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, \dots, A_k , 则 $A \rightarrow A_1, A_2, \dots, A_k$ 一定是 P 中的一个产生式。

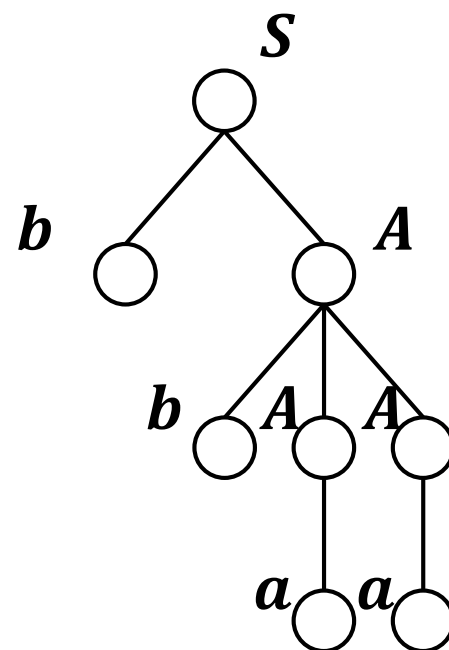
3.2 形式语言

★ 例如: $G = (\{S, A\}, \{a, b\}, P, S)$

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

G 所产生的句子 $bbaa$ 可以由下面的派生树表示:

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

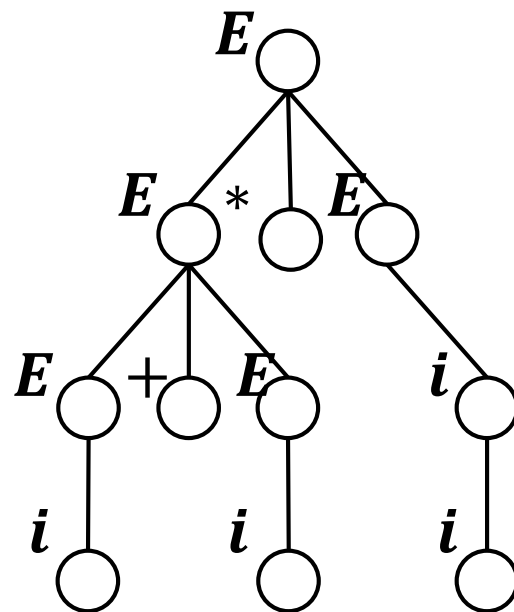
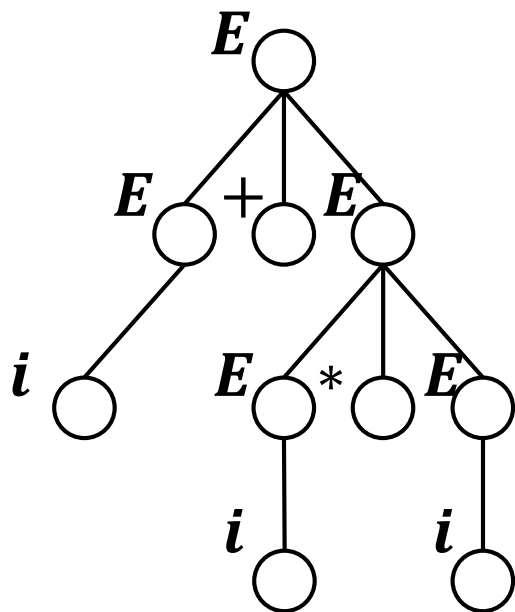


★ 上下文无关文法的二义性

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

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 $PP \rightarrow P NP$

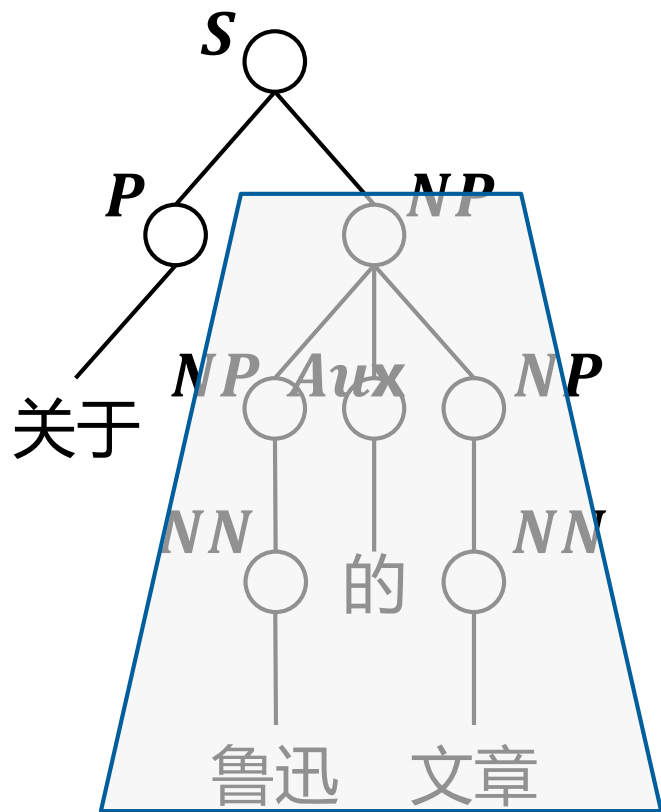
③ $NP \rightarrow NN | NP Aux NP$ ④ $P \rightarrow \text{关于}$

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

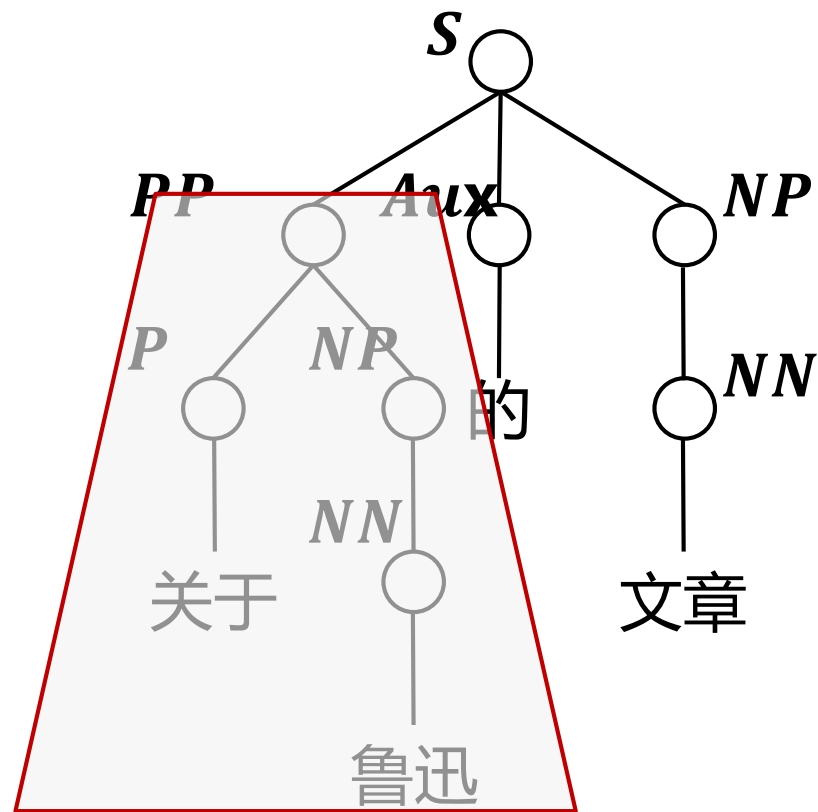
短语 “关于鲁迅的文章” 的推导。

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

3.2 形式语言



短语的派生树 (1)



短语的派生树 (2)

3.3 自动机

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	识别机类型
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

3.3.1 有限自动机与正则文法

★ 确定的有限自动机(definite automata, DFA)

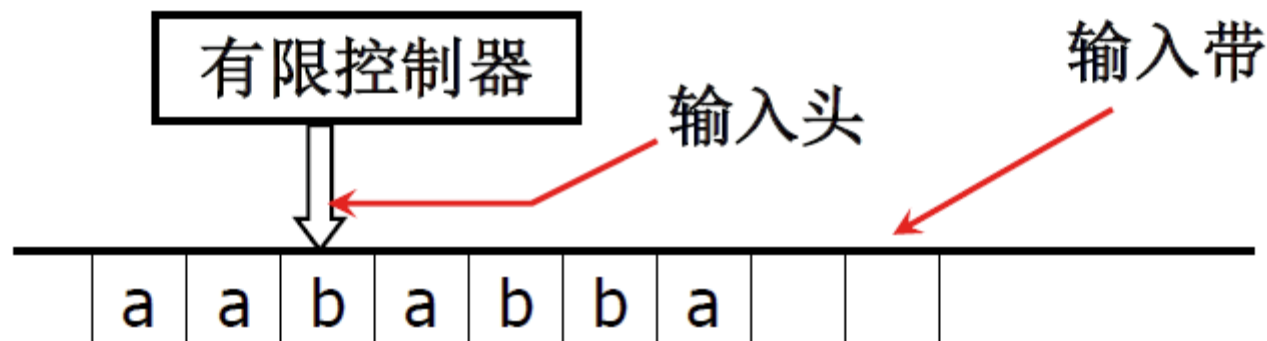
确定的有限自动机 M 是一个五元组:

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

- ✦ 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, 有时也成为**状态转移函数**。

3.3.1 有限自动机与正则文法

★ DFA示意图

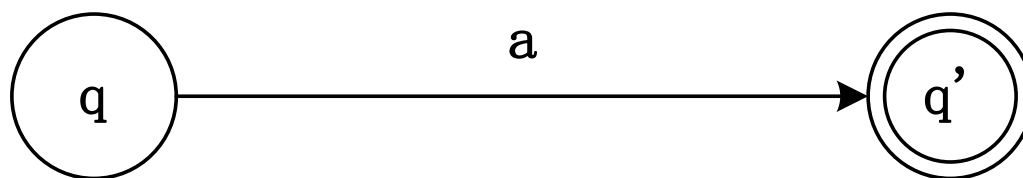


- ✦ 处在状态 $q \in Q$ 中的有限控制器从左到右依次从输入带上读入字符。开始时有限控制器处在状态 q_0 ，并注视 Σ^* 中一个链的最左符号。映射 $\delta(q, a) = q' (q, q' \in Q, a \in \Sigma)$ 表示在状态 q 时，若输入符号为 a ，则自动机进入状态 q' 并且将输入头向右移动一个字符。

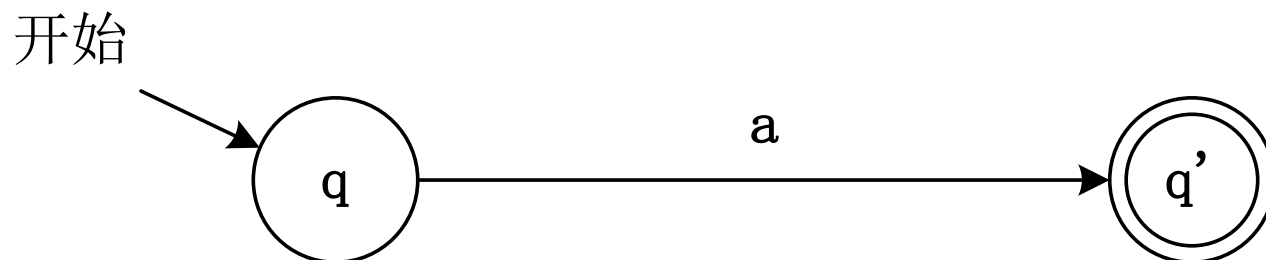
3.3.1 有限自动机与正则文法

★ 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

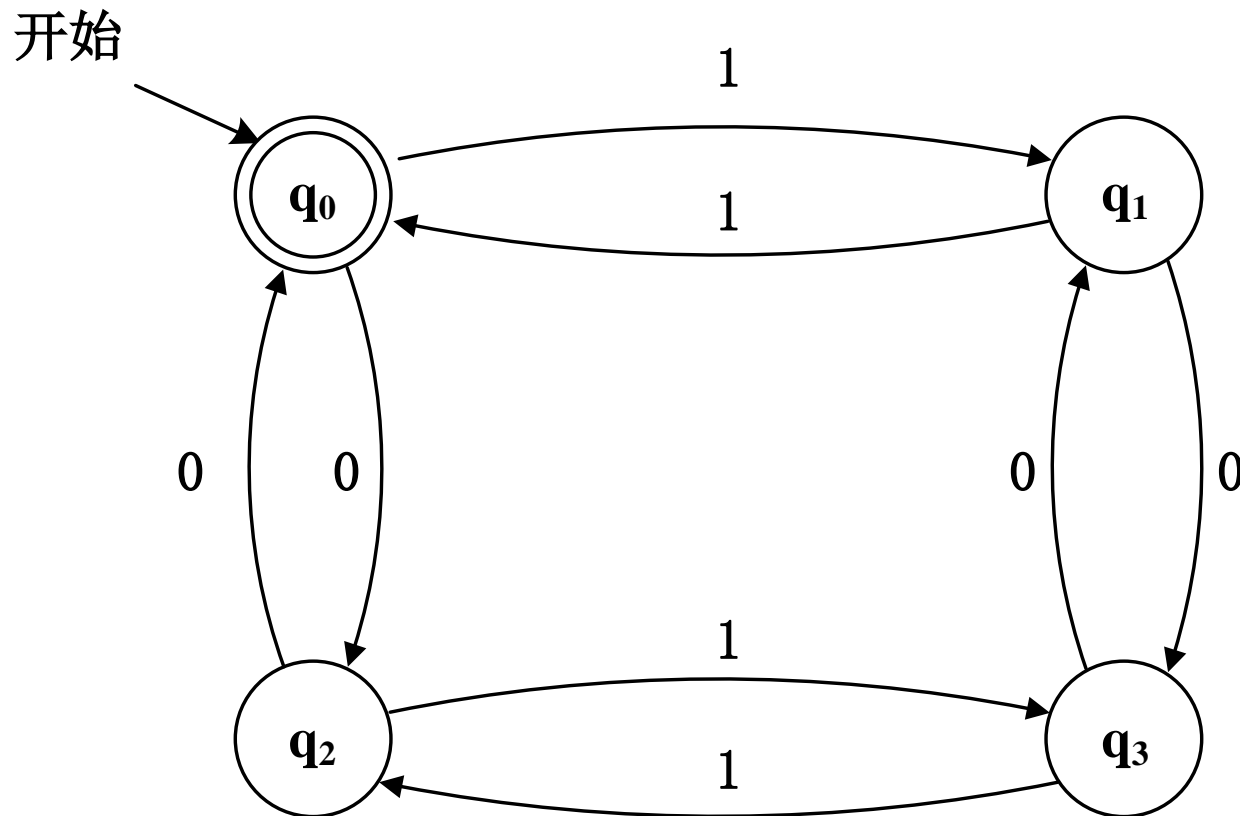
★ 如果一个句子 x 使得有限自动机 M 有 $\delta(q_0, x) = p, p \in F$, 那么, 称句子 x 被 M 接受。

★ 由 M 定义的语言 $T(M)$ 就是被 M 接受的句子的全集。即:

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

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:

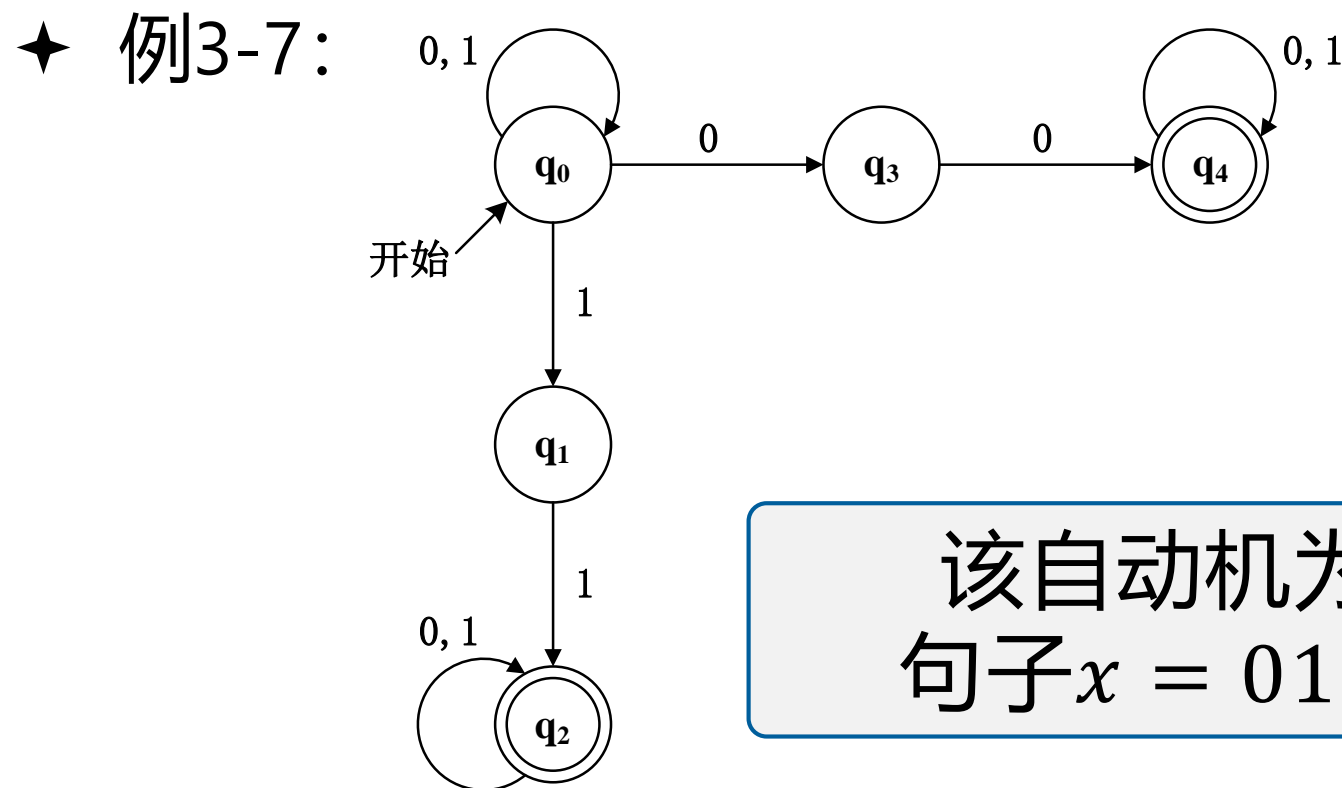
$$M = (\Sigma, Q, \delta, q_0, F)$$

- ✦ 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。

3.3.1 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。



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的区别

- ✦ **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 。
- ✦ 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)**。
- ✦ **定理3.2**: 若 $G = (VN, VT, P, S)$ 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 $G = (V_N, V_T, P, S)$ 构造 $M = (\Sigma, Q, \delta, q_0, F)$ 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

★ 解:

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

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

3.3.1 有限自动机与正则文法

(2) 映射 δ 为:

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

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

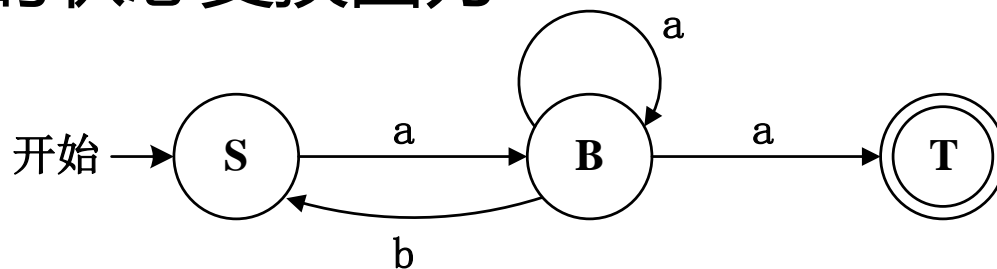
$$\delta(B, a) = \{B, T\} \quad (B \rightarrow aB | a)$$

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

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

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

等价的NFA的状态变换图为:



3.3.1 有限自动机与正则文法

✦ **定理3.3**: 若 $M = (\Sigma, Q, \delta, q_0, F)$ 是一个有限自动机, 则存在一个正则文法 $G = (V_N, V_T, P, S)$, 使得: $L(G) = T(M)$

✦ **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 $B \rightarrow a$

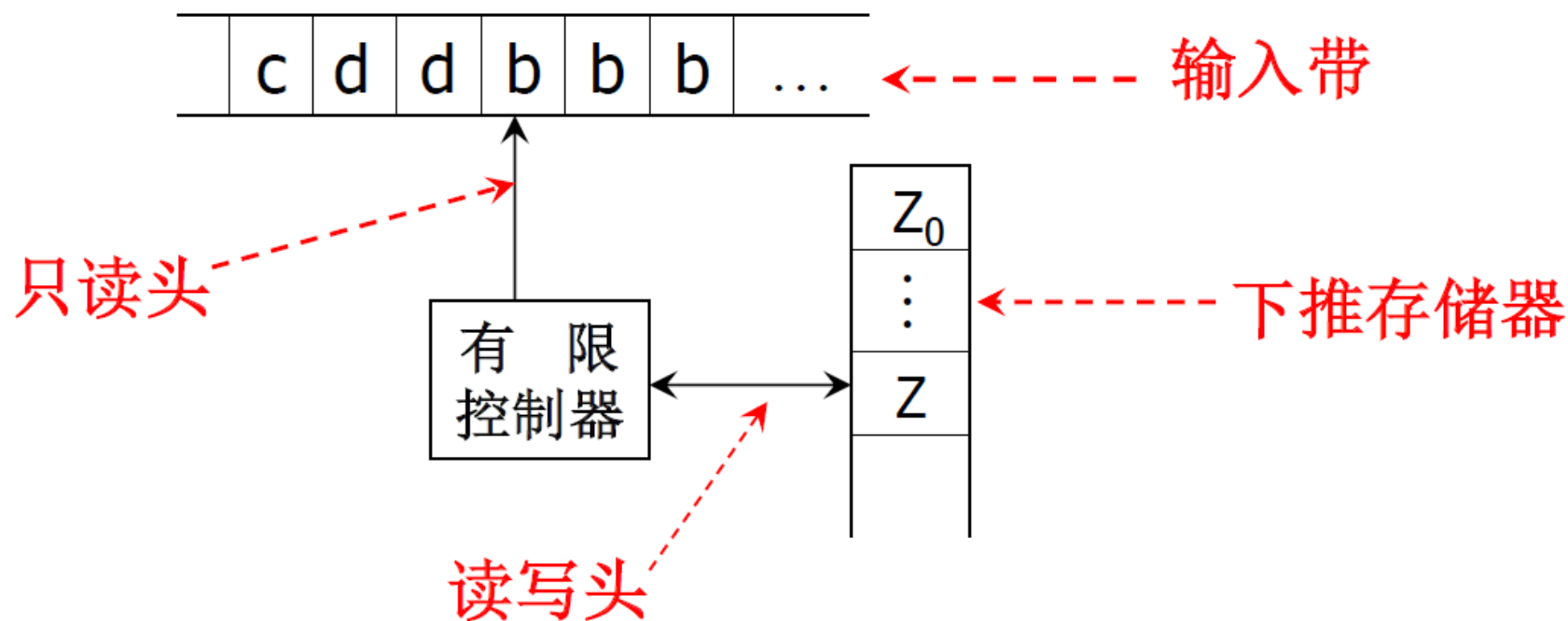
3.3.2

上下文无关文法与下推自动机

3.3.2 上下文无关文法与下推自动机

★ 下推自动机(pushdown 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 上下文无关文法与下推自动机

★ **PDA的定义**：一个不确定的**PDA**可以表达成一个**7**元组

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

- ✦ 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。

3.3.2 上下文无关文法与下推自动机

★ 映射关系 δ 的理解

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

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

★ **含义**: 当PDA处于状态 q , 面临输入符号 a 时, 自动机将进入 q_i , $i = 1, 2, \dots, m$ 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。

3.3.2 上下文无关文法与下推自动机

✦ 特殊情况下，当：

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

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”。

3.3.2 上下文无关文法与下推自动机

★ 符号约定:

- ✦ 设有序对 (q, γ) , $q \in Q, \gamma \in \Gamma^*$, 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 $a: (q, Z\gamma) \vdash_M (q', \beta\gamma)$ 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 上下文无关文法与下推自动机

★ 下推自动机接受的语言

下推自动机 M 接受的语言定义为:

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

- ✦ 终止状态接受标准: 对于输入句子 x , 如果PDA从初始状态 q_0 开始到终止状态 q 时, x 正好被读完, 则认为句子 x 被PDA M 接受, 而不管这时下推储存器里的内容如何。
- ✦ 空储存器接受标准: 对于输入句子 x , 当输入头指向 x 末端时, 如果下推储存器变为空, 则认为句子 x 被PDA M 接受, 而不管这时PDA的状态 q 是否在终止状态集 F 中。

3.3.2 上下文无关文法与下推自动机

★ 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, $Q = \{0, 1\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{A, B\}$, $q_0 = 0$, $Z_0 = \#$, $F = \{1\}$

δ 定义如下:

$$(1) \delta(0, a, \varepsilon) \underset{M}{\vdash} \{(0, A)\} \qquad (2) \delta(0, b, A) \underset{M}{\vdash} \{(0, AB)\}$$

$$(3) \delta(0, b, B) \underset{M}{\vdash} \{(0, BB)\} \qquad (4) \delta(0, c, B) \underset{M}{\vdash} \{(1, B)\}$$

$$(5) \delta(1, b, B) \underset{M}{\vdash} \{(1, \varepsilon)\} \qquad (6) \delta(1, a, A) \underset{M}{\vdash} \{(1, \varepsilon)\}$$

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

$$(2) \delta(0, b, A) \vdash_M \{(0, AB)\}$$

$$(3) \delta(0, b, B) \vdash_M \{(0, BB)\}$$

$$(4) \delta(0, c, B) \vdash_M \{(1, B)\}$$

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

$$(6) \delta(1, a, A) \vdash_M \{(1, \varepsilon)\}$$

状态	输入	栈	运用的规则
0	abbcbbba	#	-

★ 图灵机

- ✦ 图灵机与0型文法等价
- ✦ **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

★ 线性带限自动机

- ✦ 线性带限自动机与1型文法等价
- ✦ 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字符串的长度所限制。

★ 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。

3.4

有限自动机在NLP中的应用

3.4 有限自动机在NLP中的应用

★ 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：

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

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

3.4 有限自动机在NLP中的应用

- ★ 假设 $Z = z_1 z_2 \dots z_p$ 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 有限自动机在NLP中的应用

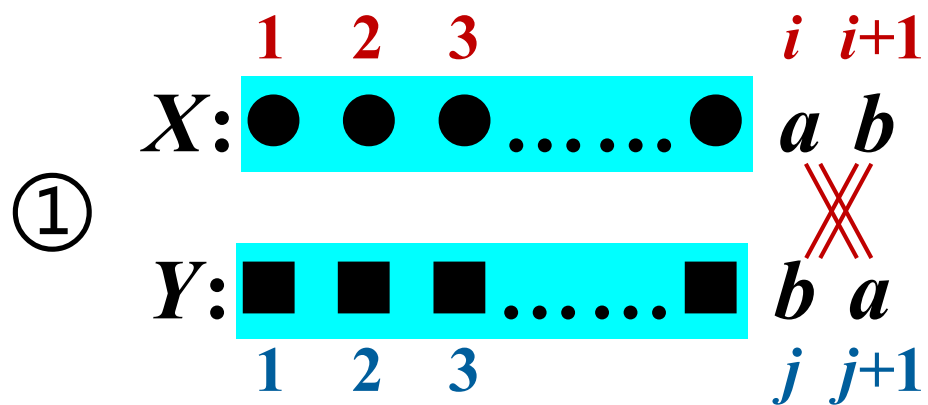
1. 如果 $x_{i+1}=y_{j+1}$ （两个串的最后字母相同），则
 $ed(X[i+1], Y[j+1]) = ed(X[i], Y[j]);$
2. 如果 $x_i=y_{j+1}$ ，并且 $x_{i+1}=y_j$ （最后两个字符需要交换位置），则

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

$$ed(X[i], Y[j+1]), \quad \textcircled{2}$$

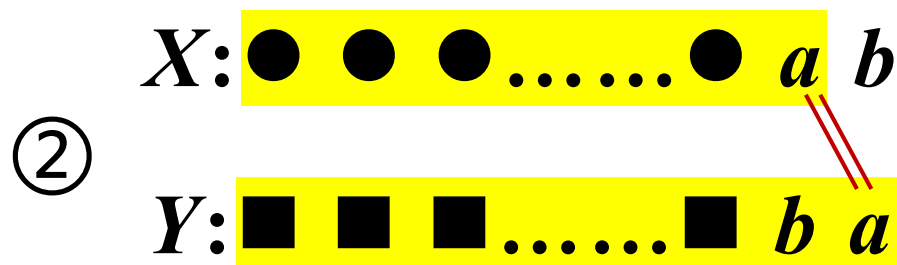
$$ed(X[i+1], Y[j])\} \quad \textcircled{3}$$

3.4 有限自动机在NLP中的应用



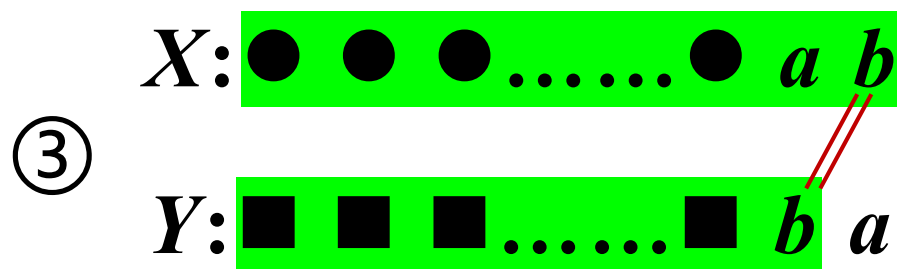
$ed(X[i-1], Y[j-1])$

(交换 X 中最末端的两个字符)



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

(删除 x_{i+1})



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

(在 x_{i+1} 插入 y_{j+1})

3.4 有限自动机在NLP中的应用

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

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

其中,

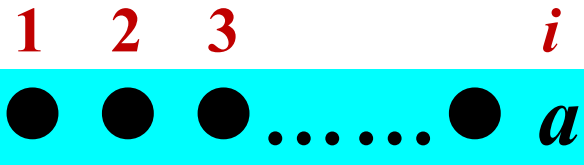
$$ed(X[0], Y[j]) = j \quad (0 \leq j \leq n) \quad (X \text{ 长度为 } 0)$$

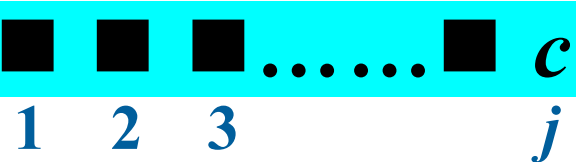
$$ed(X[i], Y[0]) = i \quad (0 \leq i \leq m) \quad (Y \text{ 长度为 } 0)$$

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

3.4 有限自动机在NLP中的应用

①


X :  a b


Y :  c d

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

$(x_{i+1} \neq y_{j+1}, \text{修改 } x_{i+1})$

②

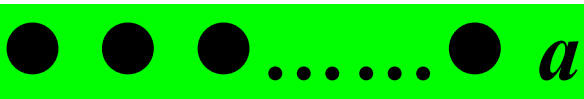
X :  a b

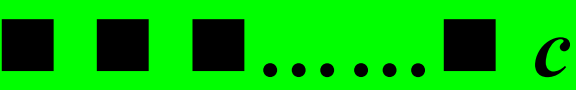
Y :  c d

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

$(x_{i+1} \neq y_{j+1}, \text{且 } x_{i+1} \neq y_j, \text{删除 } x_{i+1})$

③

X :  a b

Y :  c d

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

$(x_{i+1} \neq y_{j+1}, \text{且 } x_i \neq y_{j+1}, \text{在 } x_{i+1} \text{ 之后插入字符 } y_{j+1})$

3.4 有限自动机在NLP中的应用

★ 构造一个确定的有限状态机 R :

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

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;

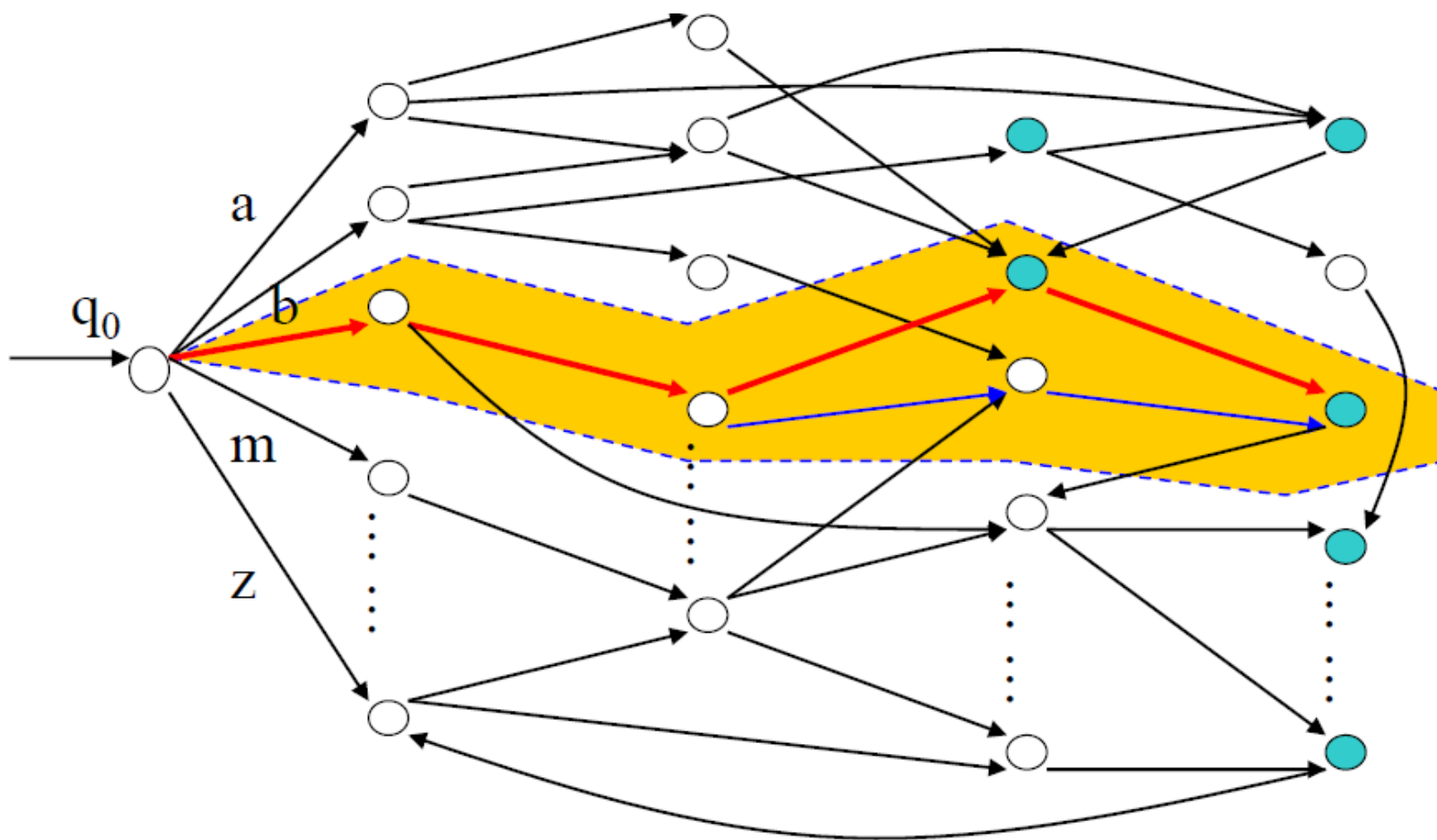
$F \subseteq Q$ 为终止状态集;

3.4 有限自动机在NLP中的应用

- ★ 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：

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

3.4 有限自动机在NLP中的应用



英文单词可用**键树**
(又称为**数字查找树**,
digital search
trees)存储。

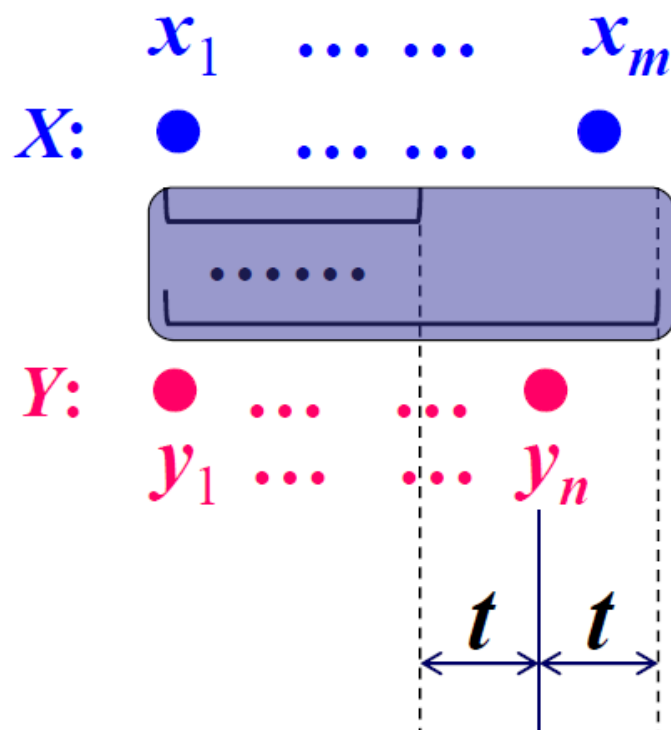
说明：蓝色节点表
示终结节点

★ 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 有限自动机在NLP中的应用

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

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



3.4 有限自动机在NLP中的应用

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

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

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

结论: rep 和repr为最接近的两个候选。

$ed(rep, repo) = 1,$

$ed(repr, repo) = 1,$

$ed(reprt, repo) = 2,$

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

$= 1$

3.4 有限自动机在NLP中的应用

★ 有限自动机用于英语单词形态分析 [Allen, 1995]

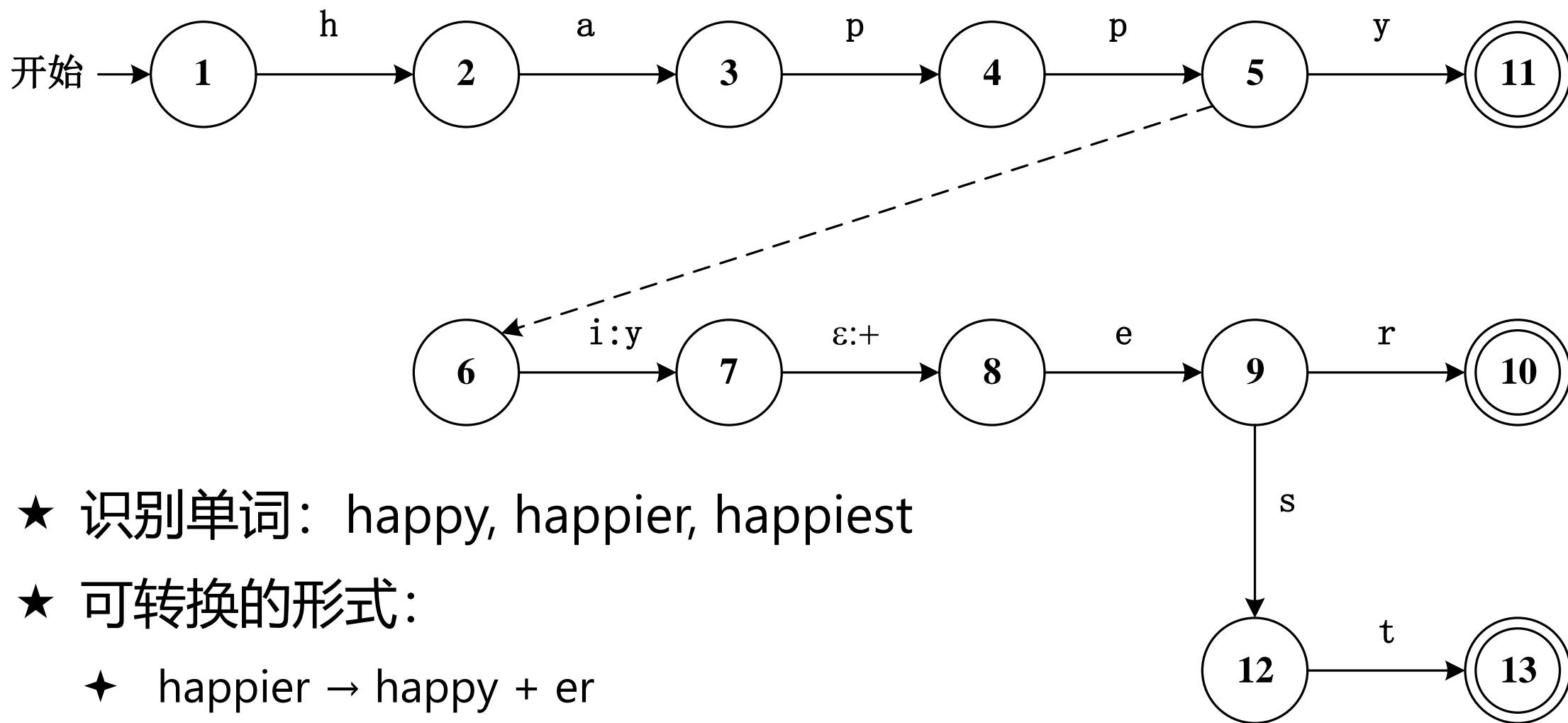
- ✦ 英语单词形态变化非常普遍，例如

- ▲ eat: eats, eating, ate, eaten

- ▲ happy: happier, happiest

- ✦ 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 有限自动机在NLP中的应用



★ 识别单词: happy, happier, happiest

★ 可转换的形式:

✦ happier → happy + er

✦ happiest → happy + est

3.4 有限自动机在NLP中的应用

- ★ 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。

除了单词拼写检查、形态分析以外，有限状态自动机还广泛应用于词性标注、句法分析、短语识别、机器翻译和语音识别等很多方面。

谢 谢

3.4 有限自动机在NLP中的应用

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1~3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。