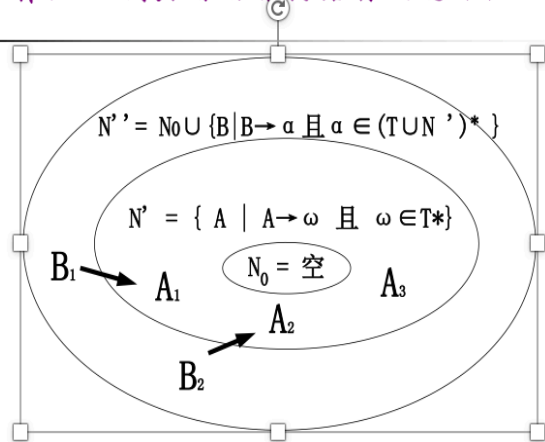




复习

- 上下文无关文法变换
 - 消无用符号
 - 消 ε 产生式
 - 消单产生式
 - 消除左递归

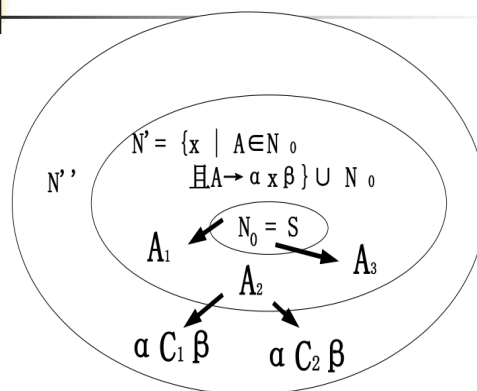
算法1: 找出有用非终结符 (图示)



一层层向外扩展, 直至最外两层相等为止。所得集合
即是算法1的有用符号。

College of Computer Science & Technology, BUPT

算法2: 找出从S可达的符号 (图示)



一层层外扩, 找出从S可达的所有符号(含非终结符和
终结符)

College of Computer Science & Technology, BUPT

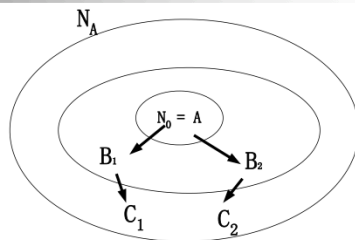
算法3: 生成无ε文法

算法步骤:

- (1) 利用算法1, 找出 $N' = \{ A \mid A \in N \text{ 且 } A \Rightarrow^+ \epsilon \}$
(找出能推导出ε的所有非终结符A)
- (2) 用以下两步组成 P_1
 - ① 如果生成式 $A \rightarrow \beta_0 C_1 \beta_1 C_2 \dots C_n \beta_n \in P$ $n \geq 0$
且每个 C_k ($1 \leq k \leq n$) 均在 N' 内
而对于 $0 \leq j \leq n$, 没有 β_j 在 N' 内 (β_j 也可能是终结符)
则 P_1 应加入 $A \rightarrow \beta_0 Y_1 \beta_1 Y_2 \beta_2 \dots Y_n \beta_n$
其中 Y_k 是 C_k 或 ϵ (即所有的可能组合)
但是 $A \rightarrow \epsilon$ 不加入 P_1

消去单产生式

算法步骤 (续):



(2) 构造 P_1 :

如果 $B \rightarrow \alpha \in P$ 且不是单生成式, 则对于 $B \in N_A$ 的所有 A , 把
 $A \rightarrow \alpha$ 加入到 P_1 中

(即对每个 $B \in N_A$ (意味着 $A \Rightarrow^+ B$) 的非单生成式 $B \rightarrow \alpha \in P$,
直接将 α 与 N_A 的 A 连接, 构成新产生式 $A \rightarrow \alpha$ 加入到 P_1 中)

(3) 得到 $G_1 = (N_1, T_1, P_1, S)$

消除直接左递归

引理2: 消除直接左递归

设 $G = (N, T, P, S)$, P 中有 A 生成式

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \dots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$$

其中 β_1 的第一个字符不是非终结符 A (可以是其它非终结符)

可构成 $G_1 = (N \cup \{A'\}, T, P_1, S)$, A' 为新引入的非终结符

G_1 中的 P_1 为, 将 P 中的 A 生成式用以下生成式取代

$$A \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n \mid \beta_1 A' \mid \beta_2 A' \mid \dots \mid \beta_n A'$$

$$A' \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_m \mid \alpha_1 A' \mid \alpha_2 A' \mid \dots \mid \alpha_m A'$$

证明思路:

G 和 G_1 二者的正则式都是 $(\beta_1 + \beta_2 + \dots + \beta_n)(\alpha_1 + \dots + \alpha_m)^*$

College of Computer Science & Technology, BUPT

§ 4.3 Chomsky范式和Greibach范式

■ Chomsky范式定义:

- 2型文法 $G = (N, T, P, S)$ ，若生成式形式都是 $A \rightarrow BC$ 和 $A \rightarrow a$ ， $A, B, C \in N$ ， $a \in T$ ，则 G 是Chomsky范式。若 $\epsilon \in L(G)$ ，则 $S \rightarrow \epsilon$ 是 P 的一个生成式，但 S 不能在任何其它生成式的右边。
- 每个上下文无关文法都具有等效的CNF（定理4.3.1）

CNF 的构成步骤

1. 用算法1、2、3、4消除 ϵ 生成式、单生成式、无用符号
2. 对生成式 $A \rightarrow D_1 D_2 \dots D_n \quad n \geq 2$

若 $D_i \in T$ ，则引入新生成式 $B_i \rightarrow D_i$ ， B_i 是新非终结符

若 $D_i \in N$ ，则令 $B_i = D_i$ ，从而将原生成式变化为

$$A \rightarrow B_1 B_2 \dots B_n \quad n \geq 2$$

当 $n > 2$ 时，再将其变为

$$A \rightarrow B_1 C_1, \quad C_1 \rightarrow B_2 C_2, \quad C_2 \rightarrow B_3 C_3, \quad \dots, \quad C_{n-1} \rightarrow B_{n-1} B_n$$

C_i 是新引入的非终结符。

定理证明——自学

CNF 的构成例

例：设 $G = (\{A, B, S\}, \{a, b\}, P, S)$ 是无 ϵ 、无循环、无无用符号、无单生成式的文法。

$P : S \rightarrow aAB \mid BA \quad A \rightarrow BBB \mid a \quad B \rightarrow AS \mid b$

求等效的CNF G_1

解： $\because S \rightarrow BA, A \rightarrow a, B \rightarrow AS, B \rightarrow b$ 已是CNF

\therefore 加入到 P_1 中

对 $S \rightarrow aAB$ ，将其变换为

$S \rightarrow C_a C_1, C_a \rightarrow a, C_1 \rightarrow AB$

将 $A \rightarrow BBB$ 变换为

$A \rightarrow BC_2, C_2 \rightarrow BB.$

CNF 的构成例

例：2型文法 $G = (\{A, B, S\}, \{a, b\}, P, S)$

$P: S \rightarrow bA \mid aB \quad A \rightarrow bAA \mid aS \mid a \quad B \rightarrow aBB \mid bS \mid b$

求等效的CNF

解：

$S \rightarrow C_b A \mid C_a B$, 增加 $C_b \rightarrow b$, $C_a \rightarrow a$

$A \rightarrow C_b D \mid C_a S \mid a$, 增加 $D \rightarrow AA$

$B \rightarrow C_a E \mid C_b S \mid b$, 增加 $E \rightarrow BB$



Greibach范式

- Greibach范式 (GNF)定义:

- 2型文法 $G = (N, T, P, S)$, 若生成式的形式都是 $A \rightarrow a \beta$, $A \in N$, $a \in T$, $\beta \in N^*$, 且 G 不含 ε 生成式, 则称 G 为Greibach范式, 记为GNF。

- 任何2型文法都具有等效的GNF (定理4.3.2)

GNF 的构成步骤

1. 将2型文法变换为CNF。 ($A \rightarrow a$, $A \rightarrow BC$ 形式)

2. 将非终结符排序, 再进行代换。

对形如 $A_i \rightarrow A_j \beta$ ($j < i$) 的生成式进行代换, 直至使

$A_i \rightarrow A_1 \beta$ ($1 > i$)。

3. 消左递归。

对最高的 $A_n \rightarrow A_n \gamma$ 进行变换, 使 A_n 生成式变为终结符开头。

4. 回代。

将 A_n 生成式回代入 A_{n-1} 生成式, 使其右部首符为终结符,

将 A_{n-1} 生成式回代入 A_{n-2} 生成式, 使其右部首符为终结符 ...

5. 最后将消直接左递归时引入的 A_1' 、 A_2' 、... A_n' 生成式右部进行代换。使其首符变为终结符。

GNF 的构成例

例：(书 P111 例2)

设已有CNF： $A \rightarrow BC$, ①

$B \rightarrow CA \mid b$, ②

$C \rightarrow AB \mid a$, ③ 将其变换为GNF。

解：(1) 按其非终结符排列为A、B、C，A是低位，C是高位。

(2) \because ①、②中，右部首符序号高于左部的非终结符

\therefore 无需变换。

对③，需要变换，

将①代入③得 $C \rightarrow BCB \mid a$ ④， 仍需变换，

将②代入④得 $C \rightarrow CACB \mid bCB \mid a$ ⑤

GNF 的构成例

(3) 对上述变换后所得结果消直接左递归

对 $C \rightarrow \underline{C} \underline{A} \underline{C} \underline{B} \mid \underline{b} \underline{C} \underline{B} \mid \underline{a}$ 变换为

$$\begin{array}{l} \alpha_1 \quad \beta_1 \quad \beta_2 \\ C \rightarrow \beta_1 \mid \beta_2 \mid \beta_1 C' \mid \beta_2 C' \\ C' \rightarrow \alpha_1 \mid \alpha_1 C' \end{array}$$

$$\text{即 } C \rightarrow bCB \mid a \mid bCBC' \mid aC' \quad \textcircled{6}$$

$$C' \rightarrow ACB \mid ACBC' \quad \textcircled{7}$$

GNF 的构成例

(4) 回代

将C的生成式⑥回代入B的生成式②

$B \rightarrow \underline{C}A \mid b$ 被变换为

$B \rightarrow bCBA \mid aA \mid bCBC'A \mid aC'A \mid b$ ⑧

将新的B生成式⑧回代入A的生成式①

$A \rightarrow \underline{B}C$ 被变换为

$A \rightarrow bCBAC \mid aAC \mid bCBC'AC \mid aC'AC \mid bC$ ⑨

再将新的A生成式⑨代入新引入的C'生成式⑦

$C' \rightarrow \underline{A}CB \mid \underline{A}CBC'$ 被变换为

... .. (略)

注意：新引入的 A_i' 相当于排在最低位。



§ 4.4 下推自动机 (PDA)

- 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是上下文无关语言的接收器。
- 重点
 - PDA的基本定义及其构造
 - PDA与上下文无关语言等价
- 难点
 - 根据PDA构造上下文无关文法。

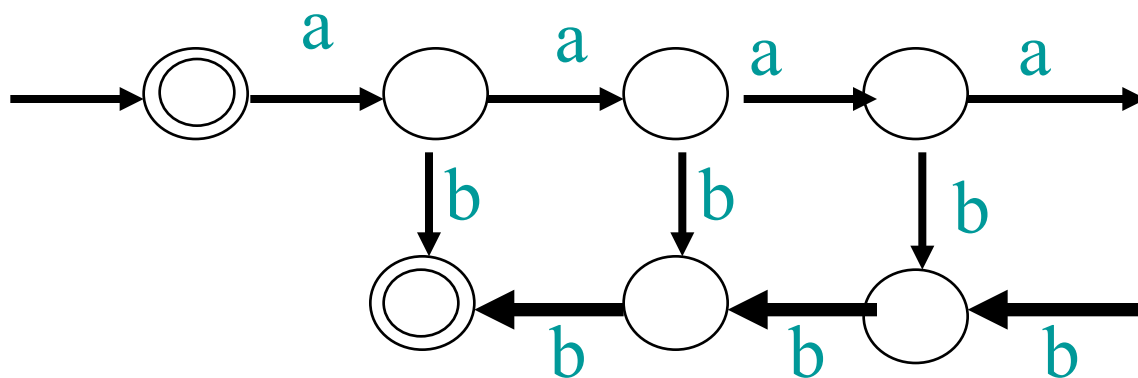


问题

- 写出识别 $a^n b^n$ 的文法，并指出是文法类型。
- 是否是正则文法，为什么？

问题的引出

✧ 类似于 $a^n b^n$ 的语言无法由一般的有限自动机识别。



识别 $a^n b^n$ 的无限状态自动机

➔ 有限状态识别器中必须有无限个状态 (不允许!)
∴ 需要扩充机器的能力。



下推自动机的结构

- 扩充办法：引入一个下推栈

- ① 足够简单

- ② 可解决许多有意义的问题，如识别有效的程序

- 下推自动机PDA (Push Down Automaton)

由一条输入带,一个有限状态控制器和一个下推栈组成

- PDA的动作

在有限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。有时，不需要考虑输入符号（空转移）。

- 栈：后进先出表

对栈的操作（压入、弹出）均在栈顶进行。

下推自动机的定义

■ NPDA的形式定义:

七元组 $M = (Q, T, \Gamma, \delta, q_0, z_0, F)$

其中: Q : 有限控制器的状态集合

T : 有限输入字母表

Γ : 有限下推栈字母表

$\delta: Q \times (T \cup \varepsilon) \times \Gamma \rightarrow Q \times \Gamma^*$

当前状态 当前输入 当前栈顶符号 有限子集

q_0 : 初始状态, $q_0 \in Q$

z_0 : 下推栈的起始符号, $z_0 \in \Gamma$

F : 终态集合, $F \subseteq Q$

下推自动机的转换函数

■ 转换函数 $\delta(q, a, Z) = \{(p, \alpha)\}$

$q, p \in Q, a \in T, Z \in \Gamma, \alpha \in \Gamma^*$

表示在状态 q ，输入字符 a ，且栈顶符 Z 时，转入状态 p ，栈顶符 Z 由 α 代替，同时读头右移一格。

■ 约定：

α 的最左符号放在栈顶。

$\alpha = \varepsilon$ 表示下推栈的顶符被弹出

如 $\delta(q, a, Z) = \{(p, \varepsilon)\}$

$\delta(q, \varepsilon, Z) = \{(p, \alpha)\}$ 称为 ε 转换。

即不考虑当前输入字符，读头不移动，但控制器状态可以改变且栈顶符可以调整。

下推自动机的格局

- 格局：用于描述PDA的瞬时工作状态

PDA格局 (q, ω, α) 其中 $\omega \in \Gamma^*$, $\alpha \in \Gamma^*$

q — 当前状态

ω — 待输入串 ($\omega = \varepsilon$ 时, 表示输入字符已读完)

α — 下推栈中的内容 ($\alpha = \varepsilon$ 时表示栈已弹空)

- $\delta(q, a, Z) = \{(p, r)\}$ 用格局可表示为
 $(q, a\omega, Z\alpha) \vdash (p, \omega, r\alpha)$

对PDA而言,

初始格局为 (q_0, ω, z_0)

终止格局为 (q, ε, α) $q \in F, \alpha \in \Gamma^*$

下推自动机接受的语言

■ 两种接受方式

■ 终态接受:

$$L(M) = \{ \omega \mid (q_0, \omega, z_0) \vdash^* (q, \varepsilon, \alpha), q \in F, \alpha \in \Gamma^* \}$$

■ 空栈接受:

$$L(M) = \{ \omega \mid (q_0, \omega, z_0) \vdash^* (q, \varepsilon, \varepsilon), q \in Q \}$$

(当空栈接受时, 终止状态可为 Q 中任意状态, 换言之, 终止状态集是与状态无关的。此时, 取 $F = \varnothing$)

下推自动机例

- 例：构造PDA M ，接受语言 $L(M) = \{ a^n b^n \mid n \geq 1 \}$
- 思路：把输入的字符 a 入栈，当开始输入 b 时，从栈中弹出 a ，若 a 、 b 个数相同，则到达终态，且栈中空。
- 解：设PDA $M = (Q, T, \Gamma, \delta, q_0, z_0, F)$ ，

$$Q = \{q_0, q_1, q_2\}$$

q_0 — 初态；接受 a — q_1 状态；接受 b — q_2 状态；输入 ε — 回到 q_0

$$T = \{a, b\}, \quad \Gamma = \{z_0, a\}, \quad F = \{q_0\}$$

δ 定义为：

$$\delta(q_0, a, z_0) = \{(q_1, a z_0)\}$$

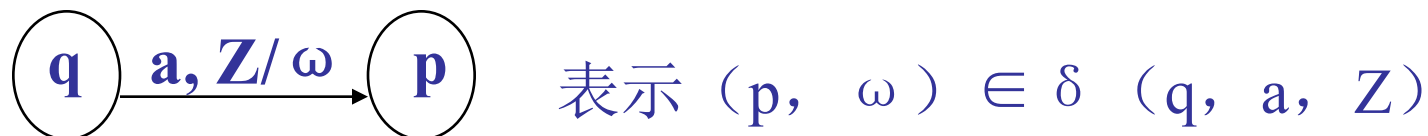
$$\delta(q_1, a, a) = \{(q_1, aa)\}$$

$$\delta(q_1, b, a) = \{(q_2, \varepsilon)\}$$

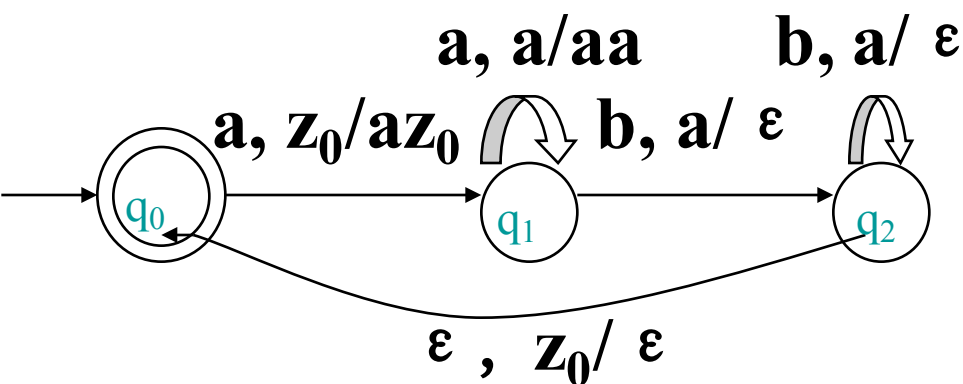
$$\delta(q_2, \varepsilon, z_0) = \{(q_0, \varepsilon)\}$$

$$\delta(q_2, b, a) = \{(q_2, \varepsilon)\}$$

下推自动机的图形表示



■ 上例的图形表示:



注：栈空就不能再移动了

用格局表示aabb的识别过程:

$(q_0, aabb, z_0)$

$\vdash (q_1, abb, az_0)$

$\vdash (q_1, bb, aaz_0)$

$\vdash (q_2, b, az_0)$

$\vdash (q_2, \epsilon, z_0)$

$\vdash (q_0, \epsilon, \epsilon) \quad \#$

终态接受



确定的下推自动机 (DPDA)

若对于每个输入字符，其后续状态都是确定的，就是DPDA（如前例）。

■ DPDA必须满足下述二个条件之一：

对 $\forall q \in Q, \forall z \in \Gamma, \forall a \in T$ 有

- (1) $\delta(q, a, z)$ 最多含一个后续选择且 $\delta(q, \varepsilon, Z) = \phi$
或者
- (2) $\delta(q, a, z) = \phi$ 且 $\delta(q, \varepsilon, z)$ 最多含一个元素。

这两个限制防止了在 ε 动作和包含一个输入符号的动作之间做选择的可能性（即在同样状态，同样栈顶符号下最多只能有一个选择。）

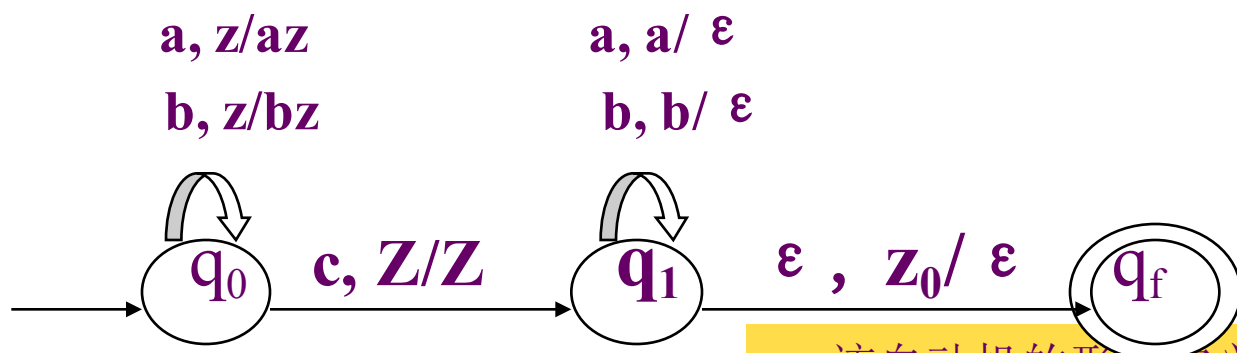
确定的下推自动机 (DPDA)

例：构造PDA M ，接受语言 $L(M) = \{\omega c \omega^T \mid \omega \in \{a, b\}^*\}$.

解题思路：

- ① 从状态 q_0 接受句子 ω ，将输入存到栈中，状态不变，直到看到中心标记 c 。
- ② 当达到 c 时，将状态变为 q_1 ，栈不变。
- ③ 将输入与下推栈匹配，状态不变，退栈，直至栈空。

采用格局写出 $abcba$ 的识别过程



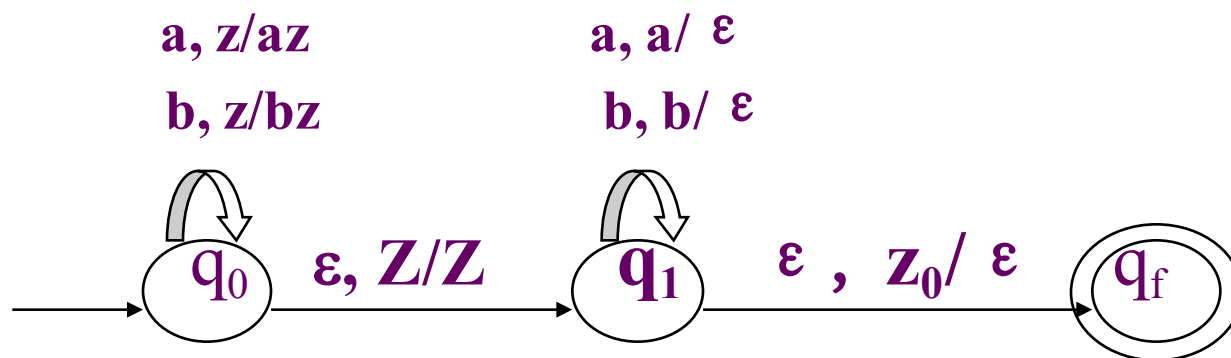
该自动机的形式定义：见书P116

非确定的下推自动机 (NPDA)

例：构造PDA M ，接受语言 $L(M) = \{\omega\omega^T \mid \omega \in \{a, b\}^*\}$.

(与前面的例子类似，区别在于中间没有标志”c”)

解：



把 “c, z/z” 改为 “ ϵ , z/z” 就引进了非确定性。因为机器可在任何时刻进行这种 ϵ 转换。

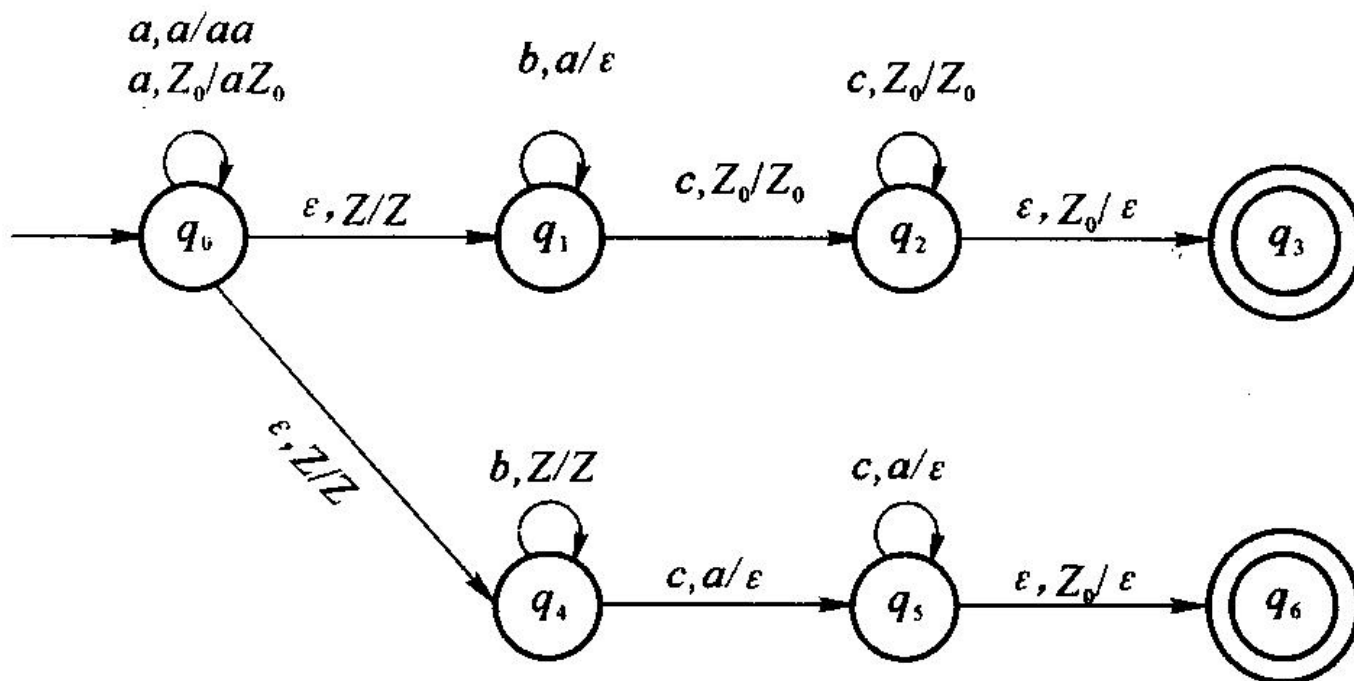
非确定的下推自动机 (NPDA)

例：构造PDA M，接受语言 $L(M) = \{ a^i b^j c^k \mid i = j \text{ 或 } i = k \}$ 。

解题思路：

与前例类似，利用不确定性，PDA可以猜想a应与b匹配还是与c匹配。
所构造的NPDA M利用两个不确定的分支实现不同的猜想。

解：



空栈接受与终态接受的等价

定理4.4.1 如果 L_f 是PDA M_f 以终态接受的语言, 必存在一个PDA M_ϕ 以空栈接受语言 L_ϕ , 使 $L_\phi = L_f$

证明: 设 $M_f = (Q, T, \Gamma, \delta, q_0, z_0, F)$

构造PDA $M_\phi = (Q \cup \{q_e, q_1\}, T, \Gamma \cup \{z_1\}, \delta_1, q_1, z_1, \phi)$

用 M_ϕ 模拟 M_f



δ_1 定义:

① $\delta_1(q_1, \epsilon, z_1) = \{(q_0, z_0z_1)\}$

(将 z_1 作为栈底符, 进入 M_f 的起始状态)

$\{(q_e, \epsilon)\}$

, 用 ϵ 转换进入 q_e 状态, 弹出栈顶)

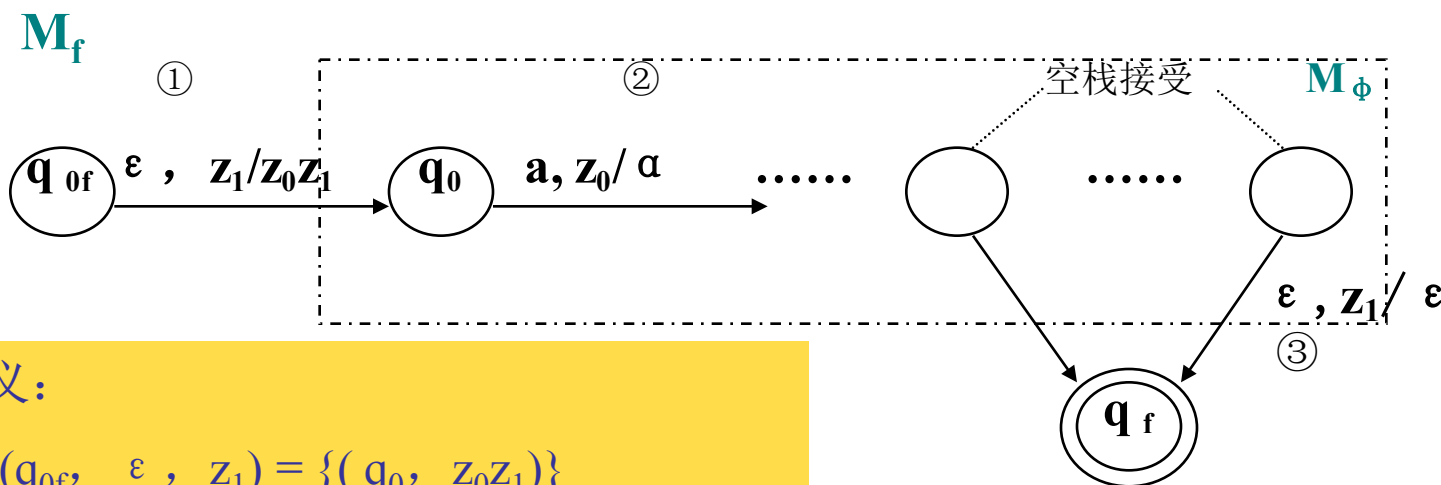
若栈不为 ϵ , 则不断弹出栈顶符, 直至栈

空栈接受与终态接受的等价

定理4.4.2 如果 L_ϕ 是PDA M_ϕ 以空栈接受的语言, 必存在一个PDA M_f 以终态接受语言 L_ϕ , 使 $L_f = L_\phi$

证明: 设 $M_\phi = (Q, T, \Gamma, \delta_\phi, q_0, z_0, \phi)$

构造PDA $M_f = (Q \cup \{q_{0f}, q_f\}, T, \Gamma \cup \{z_1\}, \delta_f, q_{0f}, z_1, \{q_f\})$




δ_f 定义:

① $\delta_f(q_{0f}, \epsilon, z_1) = \{(q_0, z_0z_1)\}$

② $\delta_f(q, a, z) = \delta_\phi(q, a, z)$

③ $\delta_f(q, \epsilon, z_1) = \{(q_f, \epsilon)\}$



作业：ch4习题.

25 : 1-3