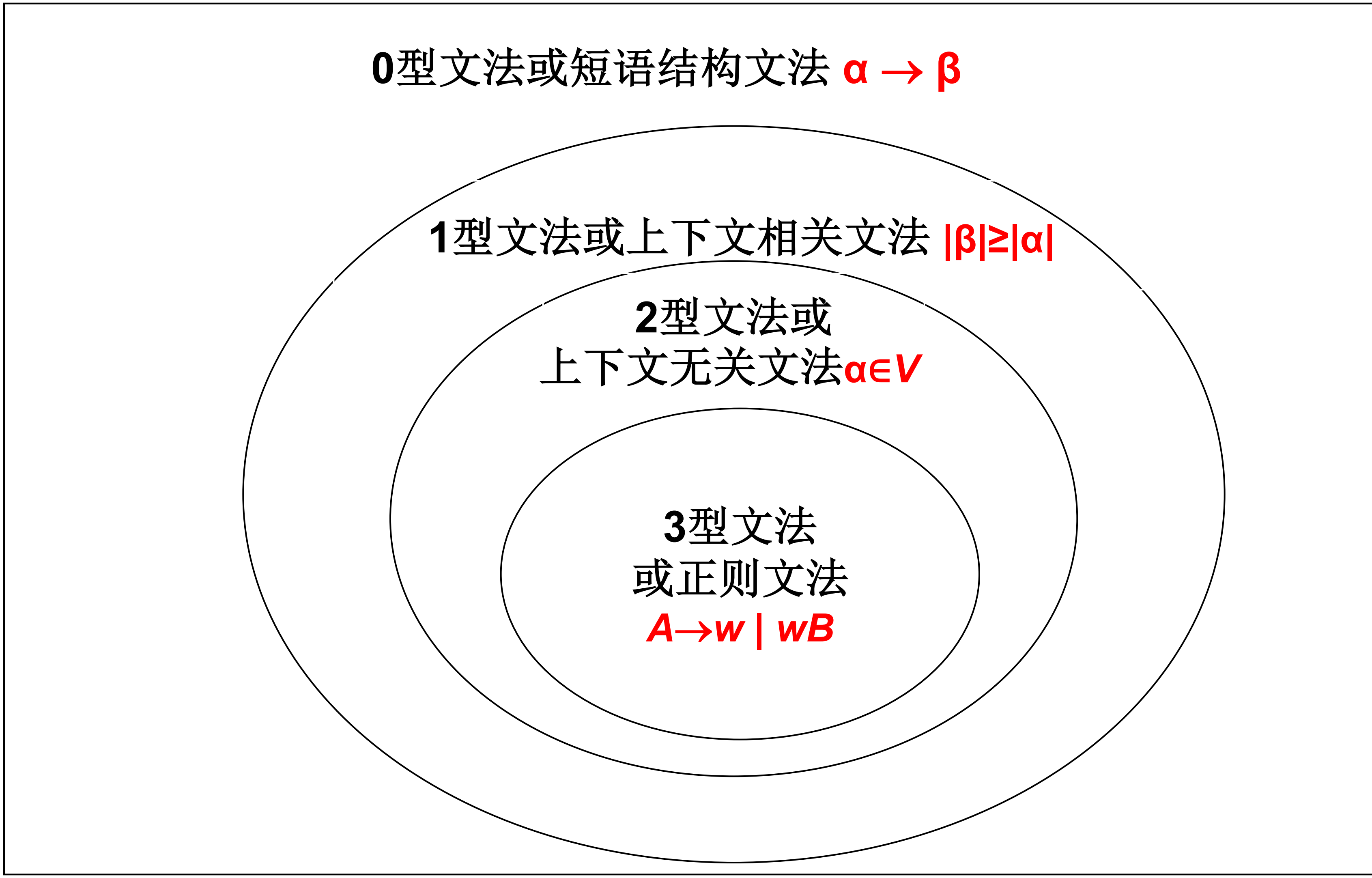
第6章 上下文无关语言

乔姆斯基文法体系（4型文法）



引言

CFG/CFL的主要应用

1. **语法分析器**：生成描述语言结构特征的语法树（parse tree）。
2. **描述文档格式**：如XML中的DTD（描述Web上的信息交换格式）。

注意：CFG描述语法结构，RL适用于词法。

6.1 上下文无关文法

> **定义 6.1** CFG(Context Free Grammar)上下文无关文法：$G = (V, T, P, S)$。其中：V 是变元集，变元也称为非终结符或**语法范畴**， T 是终结符集，P 是产生式规则，S 是开始字符。

- **特别注意**：CFG 的 P 中的规则都是如下形式：$V \rightarrow (V \cup T)^*$（产生式规则与上下文无 关）。

- 上下文无关语言定义为：$L(G) = \{\omega \in T^* \mid S \Rightarrow^* \omega\}$

例1 $L=\{0^n 1^n | n \in N\}$

$G = (\{S\},\{0,1\},P,S)$

$P: S \rightarrow \varepsilon, S \rightarrow 0S1$

例2 $L = \{0^n 1^{2n+1} | n \in N\}$

$G = (\{S\},\{0,1\},P,S)$

$P: S \rightarrow 1, S \rightarrow 0S11$

6.1 上下文无关文法

例3. $L=\{w \in (0, 1)^* \mid w$至少包括三个$1\}$

例4. $L=\{w \in (0, 1)^* \mid w$中0和1 的个数相等$\}$

6.1 上下文无关文法

例5. $L=\{0^n1^m \mid n , m \in N \}$

$$0^n 1^m = 0^p 0^k 1^k 1^q$$

A B C

S->ABC

A->0A | ε

B->0B1 | ε

C-> 1C| ε

6.1 上下文无关文法

定义 6.2 CSG(Context Sensitive Grammar)上下文有关文法：G = (V, T, P, S)。其中：V 是变元集，T 是终结符集，P 是产生式规则，S 是开始字符。

特别注意： CSG 的 P 的规则都是如下形式：

$\omega_1 V \omega_2 \rightarrow \omega_1 (V \cup T)^* \omega_2$ ， $(\omega_1, \omega_2 \in T^*)$

(产生式规则和上下文有关，并且规定了在什么情况下变量能够推导)。

定义 6.3 CFL(Context Free Language) 上下文无关语言L:存在一个 CFG G，使得 L(G) = L，其中，
$L(G) = \{\omega \in T^* \mid S \Rightarrow^* \omega\}$ 。

6.1 上下文无关文法

定理 6.1 RL ⊂ CFL。

思路：令∀ L ⊆ RL，考虑 L 的正则表达式 E，都可以找到一个 CFG G，使得 L= L(E) =L(G) 。

证明

1. 当 |E| ≤ 1 时是容易的。
2. 假设对任意长度比 |E| 小的正则表达式定理都成立。
3. 那么考虑由正则表达式的定义推导出 E 的最后一步运算。由归纳假设，$|E_1|$，$|E_2|$ < |E|，设 $L(S_1) = L(E_1)$，$L(S_2) = L(E_2)$。

分情况讨论：

① $E = E_1 + E_2$ ： $S \rightarrow S_1 \mid S_2$

② $E = E_1E_2$ ： $S \rightarrow S_1S_2$

③ $E = E_1^*$ ： $S \rightarrow S S_1 \mid \varepsilon$

所以，有 L(S) = L(E)。

6.2 语法分析树

定义 6.4　一个 CFG G = (V, T, P, S) 的语法分析树（Parse Tree）定义如下：

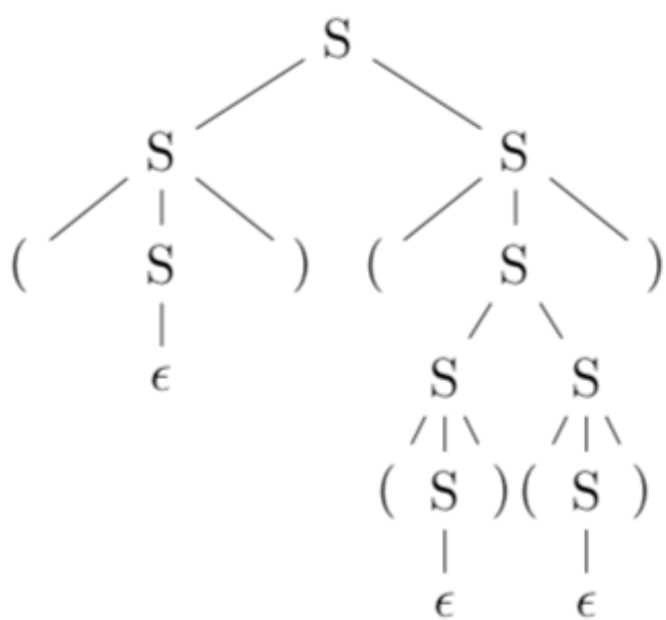
1. 根节点被标记为 S。
2. 每一个内部节点被标记为一个变量。
3. 每一个叶子节点被标记为一个终结符，特别的，假如一个节点被标记为 ε，它必须是父节点唯一的子节点。
4. 一个标记为 A 的内部节点的子节点从左到右被标记为 $X_1 . . . X_k$，当且仅当存在产生式 $A \rightarrow X_1 ... X_k$。

- 语法分析树(parse tree)又称派生树(derivation tree)、语法树(syntax tree)。

6.2 语法分析树

例5.　　S → ε | (S) | SS　　　推出 ()(()())

$$S \underset{lm}{\Rightarrow} SS \underset{lm}{\Rightarrow} (S)S \underset{lm}{\Rightarrow} ()S \underset{lm}{\Rightarrow} ()(S) \underset{lm}{\Rightarrow} ()(SS) \underset{lm}{\Rightarrow} ()((S)S) \underset{lm}{\Rightarrow} ()(()S) \underset{lm}{\Rightarrow} ()(()(S)) \underset{lm}{\Rightarrow} ()(()())$$



6.2 语法分析树

例6.

- 算术表达式的文法

G_1:

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

- 语法变量的含义

E—表达式(expression)

T—项(term)

F—因子(factor)

P—初等量(primary)

N—函数名(name of function)

L—列表(list)

id—标识符(identifier)

↑—幂运算

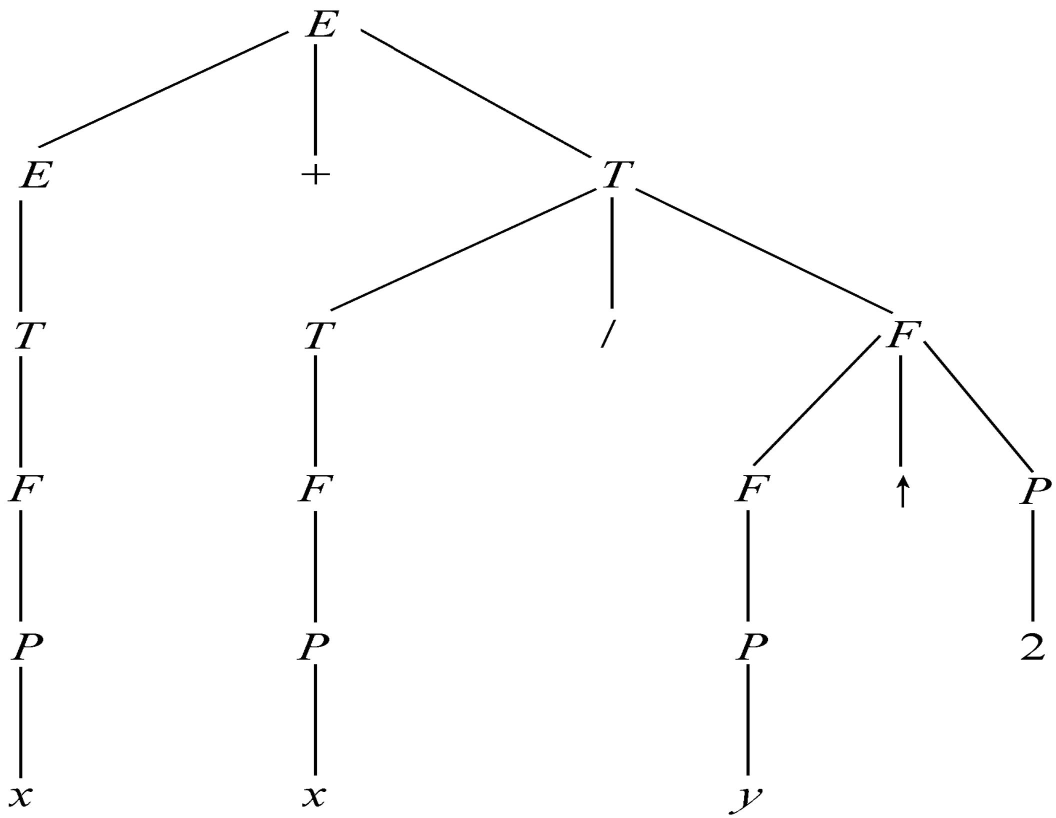
6.2 语法分析树

算术表达式 $x + x / y\uparrow 2$ 的三种不同派生/推导

$E \Rightarrow E+T$	$E \Rightarrow E+T$	$E \Rightarrow E+T$
$\Rightarrow T+T$	$\Rightarrow E+T/F$	$\Rightarrow T+T$
$\Rightarrow F+T$	$\Rightarrow E+T/F\uparrow P$	$\Rightarrow T+T/F$
$\Rightarrow P+T$	$\Rightarrow E+T/F\uparrow 2$	$\Rightarrow F+T/F$
$\Rightarrow x+T$	$\Rightarrow E+T/P\uparrow 2$	$\Rightarrow F+T/F\uparrow P$
$\Rightarrow x+T/F$	$\Rightarrow E+T/y\uparrow 2$	$\Rightarrow P+T/F\uparrow P$
$\Rightarrow x+F/F$	$\Rightarrow E+F/y\uparrow 2$	$\Rightarrow x+T/F\uparrow P$
$\Rightarrow x+P/F$	$\Rightarrow E+P/y\uparrow 2$	$\Rightarrow x+F/F\uparrow P$
$\Rightarrow x+x/F$	$\Rightarrow E+x/y\uparrow 2$	$\Rightarrow x+F/F\uparrow 2$
$\Rightarrow x+x/F\uparrow P$	$\Rightarrow T+x/y\uparrow 2$	$\Rightarrow x+F/P\uparrow 2$
$\Rightarrow x+x/P\uparrow P$	$\Rightarrow F+x/y\uparrow 2$	$\Rightarrow x+P/P\uparrow 2$
$\Rightarrow x+x/y\uparrow P$	$\Rightarrow P+x/y\uparrow 2$	$\Rightarrow x+P/y\uparrow 2$
$\Rightarrow x+x/y\uparrow 2$	$\Rightarrow x+x/y\uparrow 2$	$\Rightarrow x+x/y\uparrow 2$
最左派生	最右派生	混合派生

6.2 语法分析树

句子 $x + x / y\uparrow 2$ 的同一棵语法分析树---对应三种不同的派生/推导（最左、最右、混合）



6.2 语法分析树

定义6.5　产物（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 $X_1X_2\cdots X_n$ 是 T 的**产物**。

定义6.6　最左派生(leftmost derivation)：派生过程中，每一步都是对当前句型的最左变量进行替换。最右派生(rightmost derivation)：派生过程中，每一步都是对当前句型的最右变量进行替换。

- 最右归约(rightmost reduction)**与最左派生对应**，最左归约(leftmost reduction)**与最右派生对应**；
- 派生也称为**推导**；

6.2 语法分析树

语法分析树（Parse Tree）和推导/派生（Derivation）的关系:

1. 任意一个 Parse Tree，将它的叶子从左到右写下来，称作为 Parse Tree 生成的串，或产物。

2. Parse Tree 反映了推导这个串应用的语法规则，它的层次结构反映了语法信息(比如运算顺序)。

3. 一个 Parse Tree 对应的串的最左(或最右)推导是唯一的。

6.2 语法分析树

定理6-1 设 CFG G=(V, T, P, S)，$S \Rightarrow^* \alpha$ 的充分必要条件为 G 有一棵产物为 α 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。

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。

算术表达式的二义性文法

G_2: 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

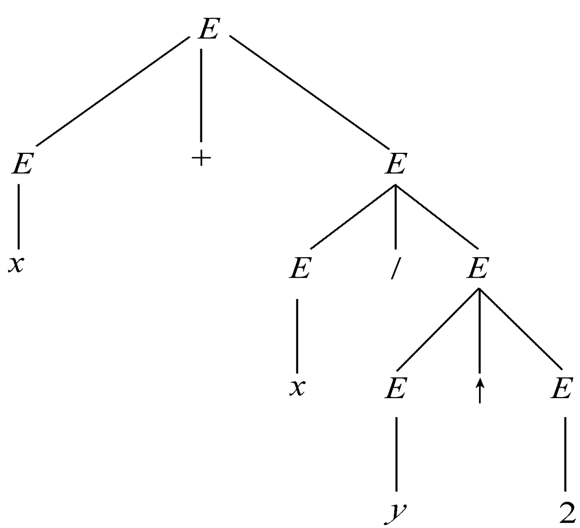
6.3 文法和语言的歧义性

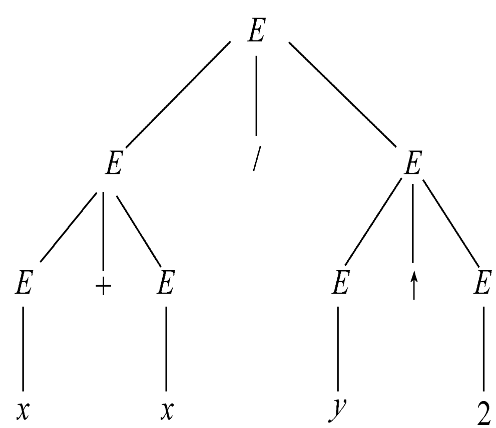
句子 $x+x/y\uparrow 2$ 在文法中的三个不同的最左派生:

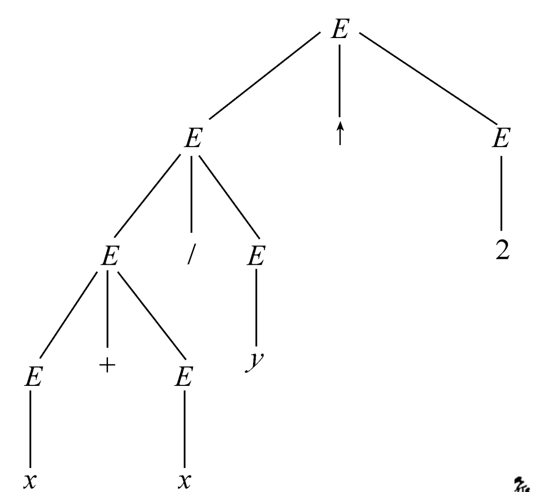
$$E \Rightarrow E+E \Rightarrow x+E \Rightarrow x+E/E \Rightarrow x+x/E \Rightarrow x+x/E\uparrow E \Rightarrow x+x/y\uparrow E \Rightarrow x+x/y\uparrow 2$$

$$E \Rightarrow E/E \Rightarrow E+E/E \Rightarrow x+E/E \Rightarrow x+x/E \Rightarrow x+x/E\uparrow E \Rightarrow x+x/y\uparrow E \Rightarrow x+x/y\uparrow 2$$

$$E \Rightarrow E\uparrow E \Rightarrow E/E\uparrow E \Rightarrow E+E/E\uparrow E \Rightarrow x+E/E\uparrow E \Rightarrow x+x/E\uparrow E \Rightarrow x+x/y\uparrow E \Rightarrow x+x/y\uparrow 2$$







6.3 文法和语言的歧义性

定义6.7 歧义性/二义性(Ambiguity)

CFG G = (V, T, P, S)，如果存在 w ∈ L(G)，w 至少有两棵不同的语法分析树，则称 G 是歧义性的或二义的。否则 G 为非歧义性的。

- G_2 是歧义性的，例1中的G_1 是非歧义性的，但二者是等价的，即$L(G_1)=L(G_2)$
- 判定 CFG G 是否为歧义性的问题是一个不可解的(unsolvable)问题。

6.3 文法和语言的歧义性

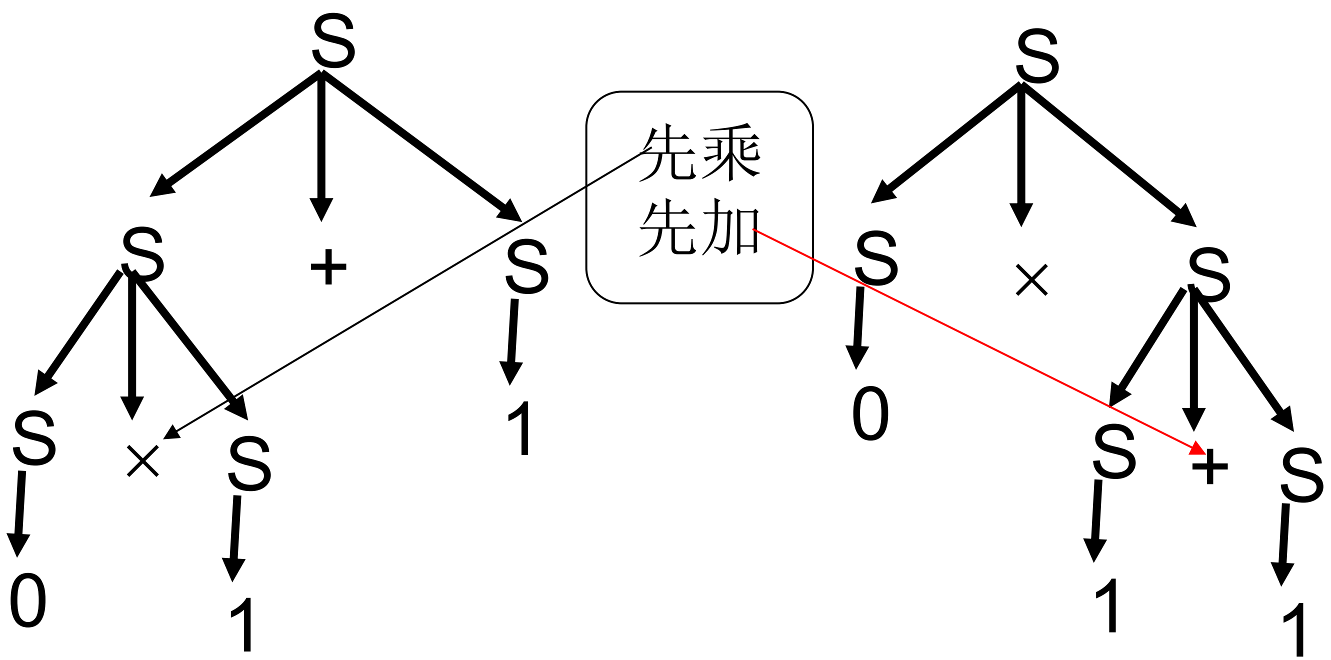
例7. **歧义性的原因**

G: $S \rightarrow 0 \mid 1 \mid S+S \mid S\times S$

多棵语法分析树导致歧义性，
而不是多种推导导致了歧义性

①$S \Rightarrow S+S \Rightarrow S\times S+S \Rightarrow 0\times S+S \Rightarrow 0\times 1+S \Rightarrow 0\times 1+1$

②$S \Rightarrow S\times S \Rightarrow 0\times S \Rightarrow 0\times S+S \Rightarrow 0\times 1+S \Rightarrow 0\times 1+1$



6.3 文法和语言的歧义性

Chomsky Normal Form(乔姆斯基范式)

1. CFG = (V,Σ,R,S) is in CNF if every rule is of the form
   - $A \to BC$ 一分为二
   - $A \to x$ 或终极化
   - $S \to \varepsilon$

其中，变元$A \in V$ and $B,C \in V \setminus \{S\}$, and $x \in \Sigma$

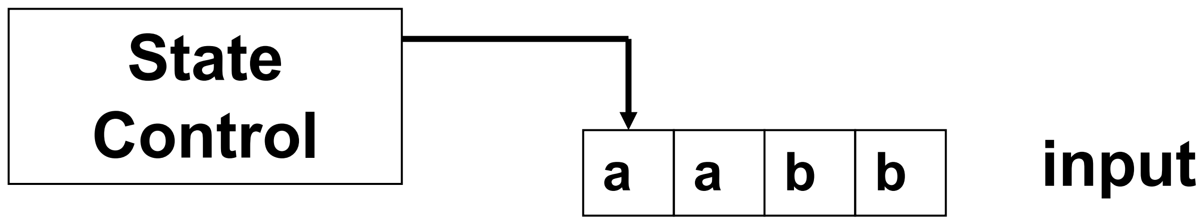
2. Every context-free language can be described by a grammar in Chomsky normal form。

6.4 下推自动机

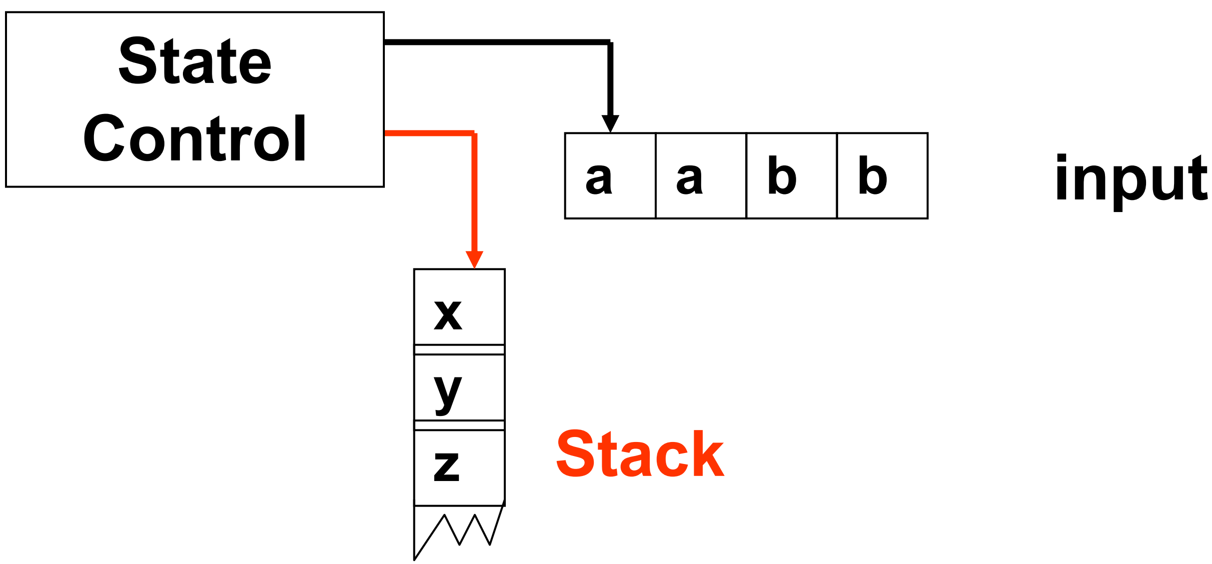
关于PDA (Pushdown Automata)

1. NFA+Stack can recognizes some nonregular languages（CFL）
2. PDA ⇔ CFG 有两钟方式可以证明一个语言是CFL
3. Stack Last in first Out
4. PDA is a nondeterministic Automata

6.4 下推自动机



Schematic of a Finite Automaton



Schematic of a PDA

Formal Definition of PDA

A Pushdown Automata M is defined by a six tuple $(Q,\Sigma,\Gamma,\delta,q_0,F)$, with

- Q finite set of states 有限个状态（寄存器）
- Σ finite input alphabet 字母表
- Γ finite stack alphabet 可压栈字符表
- q_0 start state $\in Q$
- F set of accepting states $\subseteq Q$ 接受态集合
- δ transition function 状态转移函数 ~ 相当于3种语句goto, push ,pop

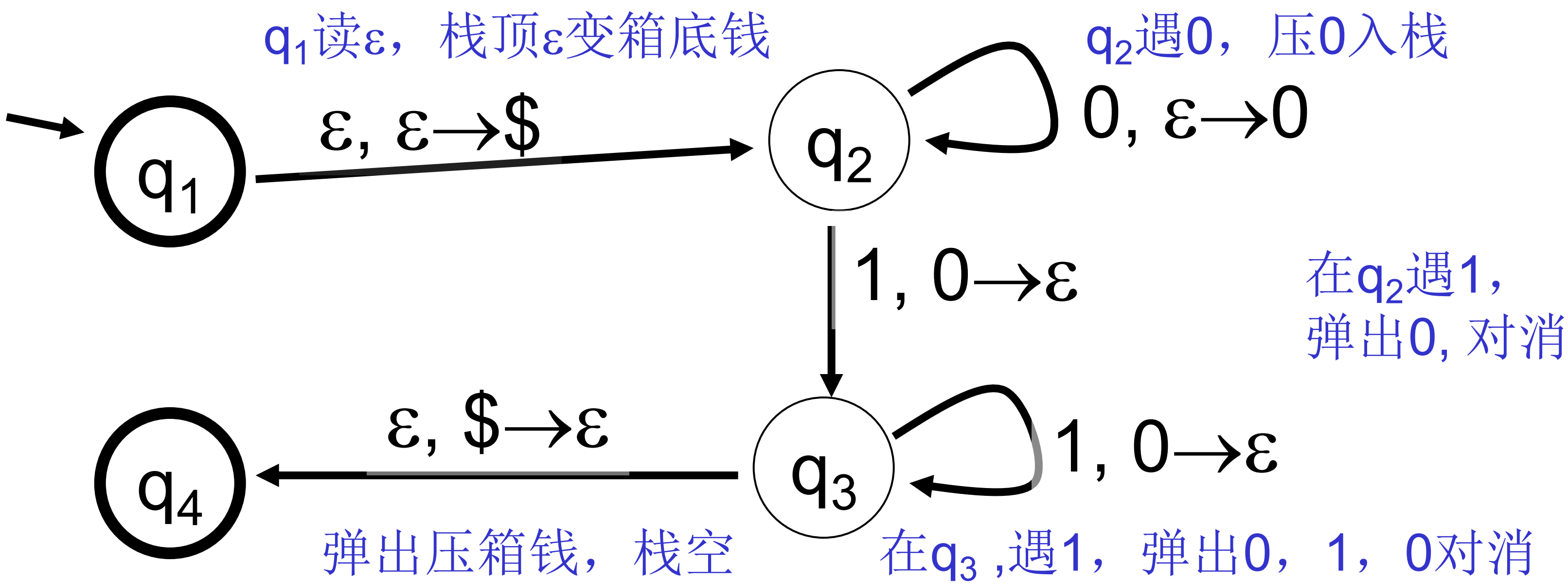
$$\delta: Q \times \Sigma\varepsilon \times \Gamma\varepsilon \rightarrow \mathcal{P}(Q \times \Gamma\varepsilon)$$ 是一个超集

6.4 下推自动机

Exp8: A PDA recognizes language $\{0^n1^n|n>=0\}$。

思路: 先将0^n 压栈；读1^n，0^n出栈；比较0与1的个数。

问题：PDA 无法判断何时为空，解决办法：先将$压栈。



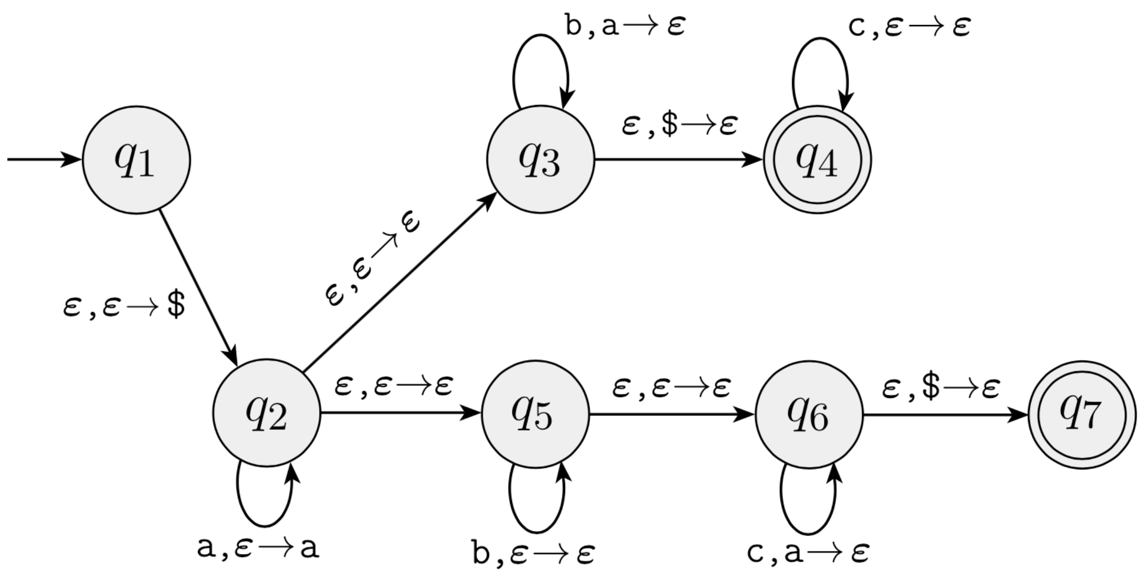
a, b→c 的含义： read ‘a’, pop ‘b’, push ‘c’

a=ε means read nothing; b= ε means no pop ; c= ε means push nothing

例9. 识别语言{$a^i b^j c^k$| i, j, k>=0 and i=j or i=k}的PDA

问题是：究竟是**a**与**b**匹配呢，还是**a**与**c**匹配呢？

不知道→都有可能→不确定→采用不确定的**PDA**



6.4 下推自动机

定义 6.8 Pushdown automaton(PDA) 下推自动机: $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$。其中 Q 是状态集， Σ 是输入字符集，Γ 是栈字符集，$q_0 \in Q$ 是初始状态，$F \subset Q$ 是接受状态集合，Z_0 是初始栈底符号， $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$ 是转移函数。

- 转移函数的一般形式如下：

$\delta(q, a, Z) = \{(p_1, \gamma_1), (p_2, \gamma_2),\ldots,(p_m, \gamma_m)\}$， 表示当自动机处于状态 q， 读到输入字符 a，发现栈顶的符号为 Z时， 可以转移到状态 p_i，同时弹出栈顶的 Z，压入 γ_i。

- 压入 γ 时，从右向左压入。即压入完成后 γ 左侧的元素对应栈顶。
- 符号约定：用字母表靠后的大写字母表示栈中的字符，如 X, Y ;用希腊字母表示栈中的字符串， 如 α,γ。

6.4 下推自动机

例10. 构造一个PDA按终结状态方式接受语言{ 0^n1^n | n≥1}。

形式化定义：

M =（{q_0，q_1，q_2，q_3}，{0，1}，{Z，A}，δ，q_0，Z，{q_3}），　其中，

δ（q_0，0，Z）={（q_1，AZ）}　在栈中加入A

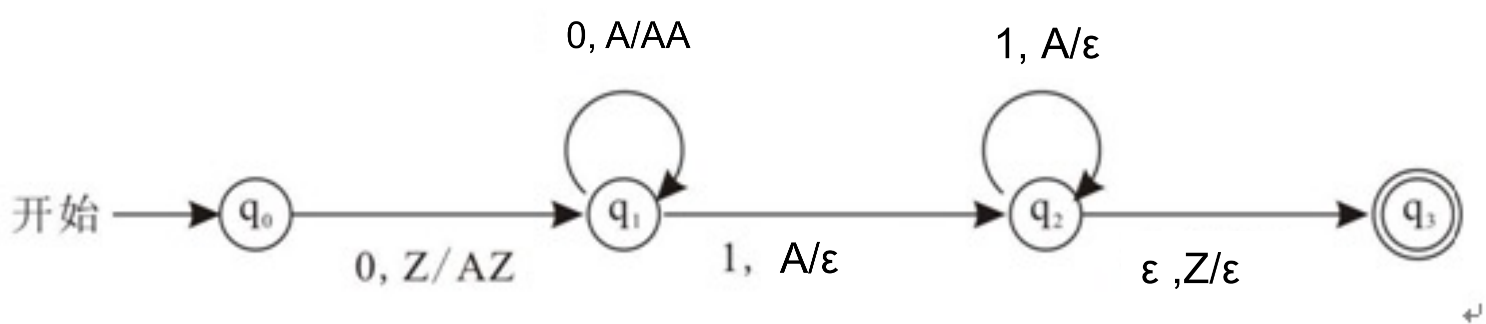
δ（q_1，0，A）={（q_1，AA）}　在栈中加入A

δ（q_1，1，A）={（q_2，ε）}　遇1消去栈顶符号

δ（q_2，1，A）={（q_2，ε）}　遇1消去栈顶符号

δ（q_2，ε，Z）={（q_3，Z）}　接受

状态转移图



6.4 下推自动机

定义 6.9 (终态接受). 考虑 PDA $P = (Q,\Sigma,q_0,\delta,F,\Gamma,Z_0)$。定义语言 L(P) 如下:$w \in L(P)$ 当且 仅当存在接受态 $q \in F$ 和栈字符串 $\alpha \in \Gamma^*$， 满足 $(q_0,w,Z_0)\vdash^*(q,\varepsilon,\alpha)$。 //栈不一定为空

定义 6.10 (空栈接受). 考虑 PDA $P = (Q,\Sigma,q_0,\delta,F,\Gamma,Z_0)$。定义语言 N(P) 如下:$w \in N(P)$ 当且仅当存在状态 $q \in Q$， 满足 $(q_0, w, Z_0) \vdash^*(q, \varepsilon, \varepsilon)$。 //栈一定为空

定理6.4 如果对于某个按终结状态方式接受语言的PDA M_1，有$L(M_1)=L$，则存在一个按空栈方式接受语言的PDA M_2，使得$N(M_2)=L$。

定理6.5 如果对于某个按空栈方式接受语言的PDA M_1，有$N(M_1)=L$，则存在一个按终结状态方式接受语言的PDA M_2，使得$L(M_2)=L$。

6.5 PDA与CFG的等价性

Theorem 6.6 一个语言是上下文无关的，当且仅当存在一台下推自动机识别它。
CFL⇔PDA

Lemma 6.7 如果一个语言是上下文无关的，则存在一台下推自动机识别它。
CFL→PDA

Lemma 6.8 如果一个语言被一台下推自动机识别，则它是上下文无关的。
PDA→ CFL

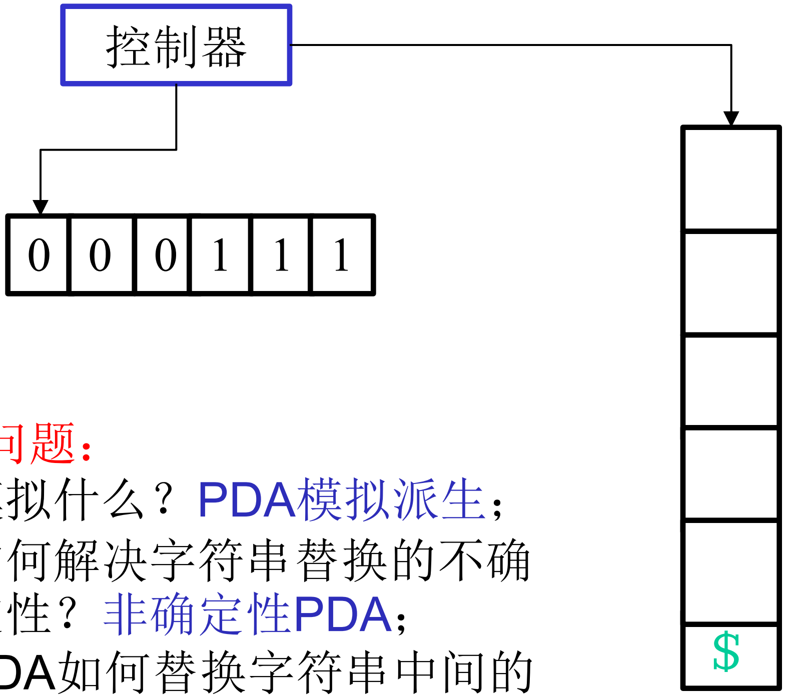
6.5 PDA与CFG的等价性

Lemma 6.7 如果一个语言是上下文无关的，则存在一台下推自动机识别它。CFL L → CFG G → PDA P

Proof Idea：如何用PDA P模拟CFG G？

L= { 0^n1^n | n>=0 }

G：S →0S1 | ε



文法G产生w=000111的过程如下：

S →0S1 ①
→ 00S11 ②
→ 000S111 ③
→ 000111 ④

关键问题：

1. 模拟什么？PDA模拟派生；
2. 如何解决字符串替换的不确定性？非确定性PDA；
3. PDA如何替换字符串中间的变元？终结符提前匹配；

6.5 PDA与CFG的等价性

给定CFL L，如何构造一个等价的PDA P

1. 把标记符号$和起始变元放入栈中；
2. 重复下列步骤：

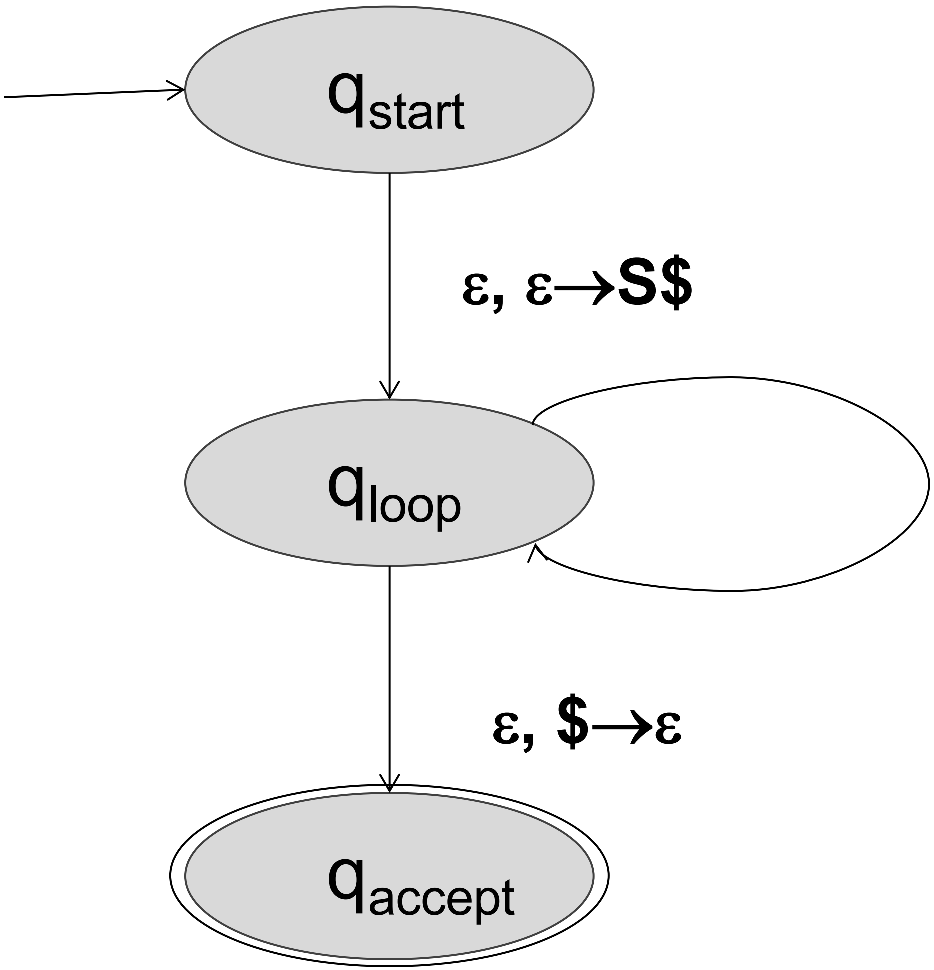
① 如果栈顶符号是变元 A，则非确定性第选择一个A的规则，并且把A替换成这条规则右边的字符串。

② 如果栈顶是终结符 a，则读取下一个输入符号，并且把它与a进行比较。如果它们匹配，则重复，如果它们不匹配，则拒绝这个非确定性分支。

③ 如果栈顶符号是 $，则进入接受状态，如果此刻输入已全部读完，则接受这个输入串。

6.5 PDA与CFG的等价性

如何把文法G转换成PDA



ε, A→ω　　for rule A → ω

a, a→ε　　for terminal a

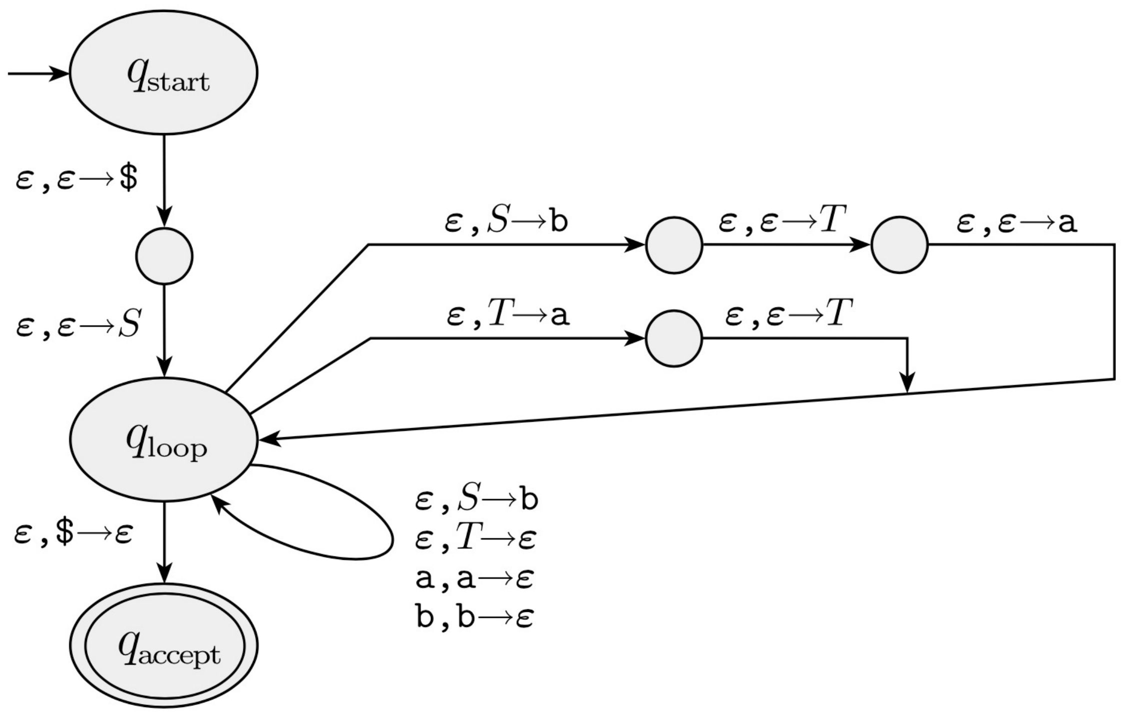
6.5 PDA与CFG的等价性

EXP：把CFG G转换成PDA P，其中

G：

$S \rightarrow aTb \mid b$

$T \rightarrow Ta \mid \varepsilon$



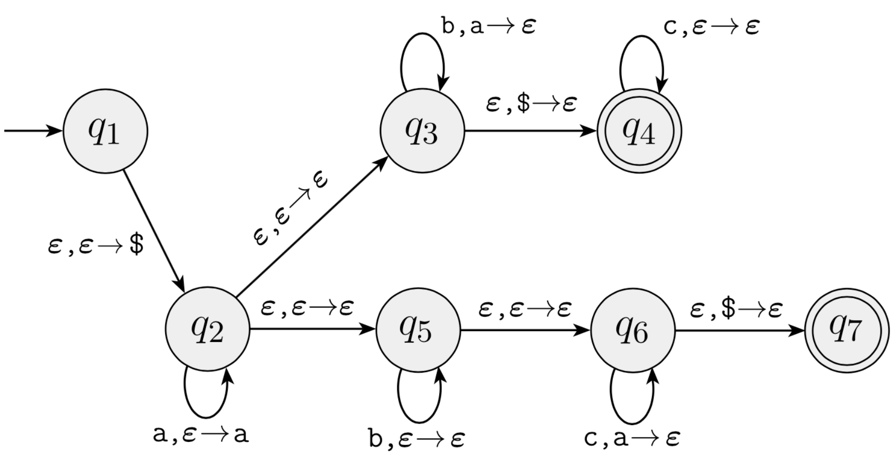
6.5 PDA与CFG的等价性

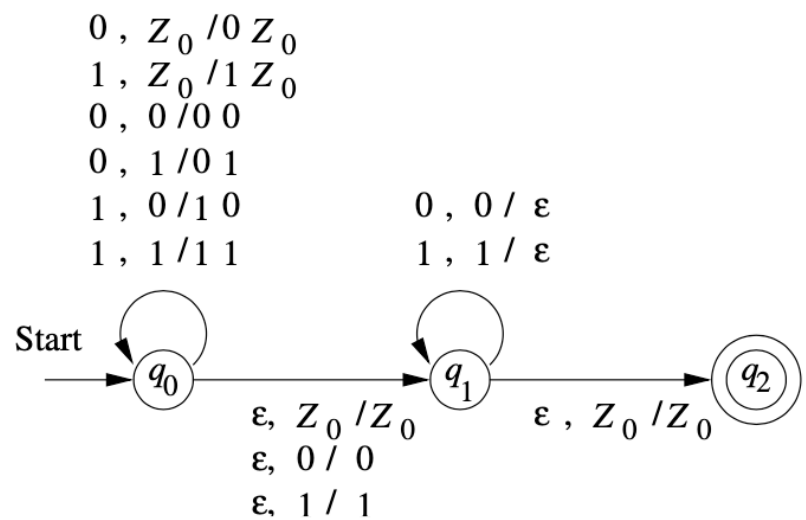
例11. 设计一个PDA P，识别语言$L=\{0^i1^j \mid i \geq j \geq 1\}$。

6.6 确定型下推自动机

例9. 识别语言$\{a^ib^jc^k \mid i, j, k>=0$ and $i=j$ or $i=k\}$的PDA。

例12. 设计一个PDA，接受语言$L_{wwr} = \{ww^R \mid w\in \{0, 1\}^* \}$。





- PDA的主要特点：**非确定性**

① 对于同一个δ(q, a, A)（q∈Q，a∈∑∪{ε}，A∈Γ），可以有多个或零个转移；

② 对于同样的q和A，δ(q,a,A)（a∈∑）和δ(q,ε,A)可以都有定义。

$\delta: Q \times \Sigma\varepsilon \times \Gamma\varepsilon \rightarrow \mathcal{P}(Q \times \Gamma\varepsilon)$ 是一个超集，$\delta(q, a, X) = \{(p_1, \gamma_1),\ldots,(p_m, \gamma_m)\}$

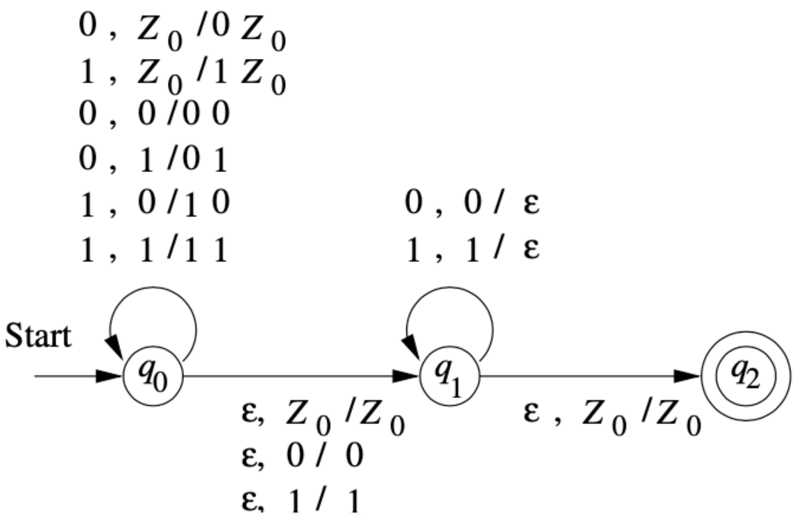
6.6 确定型下推自动机

定义6.11 一个下推自动机M=（Q, Σ, Γ, δ, q_0, Z_0, F），如果满足下列条件：

1. 对于∀q∈Q, ∀ a∈Σ_ε, ∀ A∈Γ, δ(q,a,A)至多有一个转移。
2. 对于∀ a∈Σ，若δ(q,a,A)非空，则δ(q, ε,A)为空。

则称M为确定的下推自动机（Deterministic PushDown Automaton），简记为DPDA。确定的下推自动机接受的语言称为确定的上下文无关语言，简记为DCFL。

6.6 确定型下推自动机

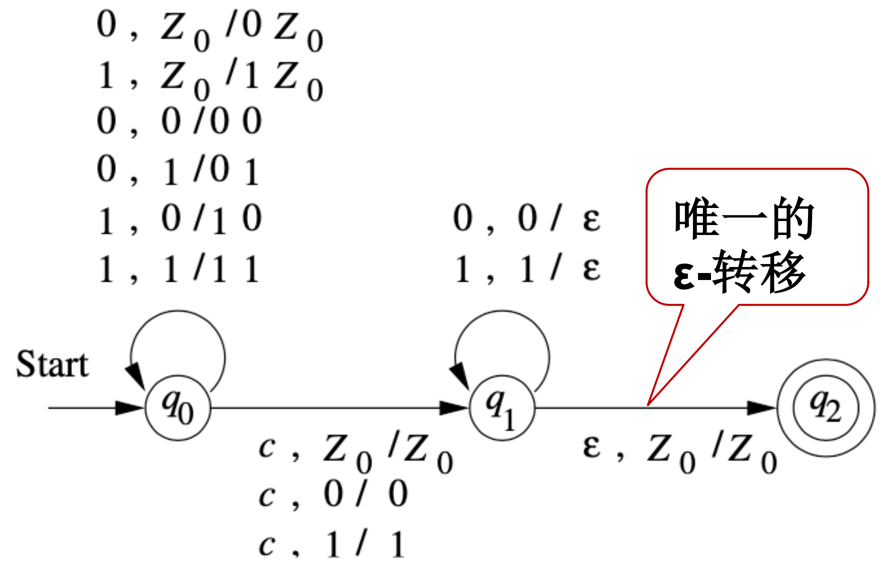


接受L_{wwr} = $\{ww^R \mid w\in \{0,1\}^*\}$的PDA

问题：

在q_0状态时，把下一字符压栈，还是 ε -转移到q_1?

GUESS



例13. 接受L_{wcwr} = $\{wcw^R \mid w\in\{0,1\}^*\}$的DPDA

6.6 确定型下推自动机

例14. 构造一个DPDA M，使其接受语言$\{0^n1^{n+2}|n>=0\}$。

分析：

① 1的个数比0的个数多2

② 当n=0时，字符串以1开头

$\delta(q_0, 0, Z) = (q_1, AZ)$

$\delta(q_1, 0, A) = (q_1, AA)$

$\delta(q_1, 1, A) = (q_2, \varepsilon)$

$\delta(q_2, 1, A) = (q_2, \varepsilon)$

$\delta(q_2, 1, Z) = (q_3, Z)$

$\delta(q_3, 1, Z) = (q_4, Z)$ //q_4是接受状态

$\delta(q_0, 1, Z) = (q_3, Z)$ //处理n=0的情形

6.6 确定型下推自动机

DCFL的重要应用

- 非固有歧义语言的真子集
- 程序设计语言的语法分析器
- LR(k)文法，Yacc的基础

6.6 确定型下推自动机

定理6.7 如果L是正则语言，则存在某个DPDA P 有 L=L(P)。

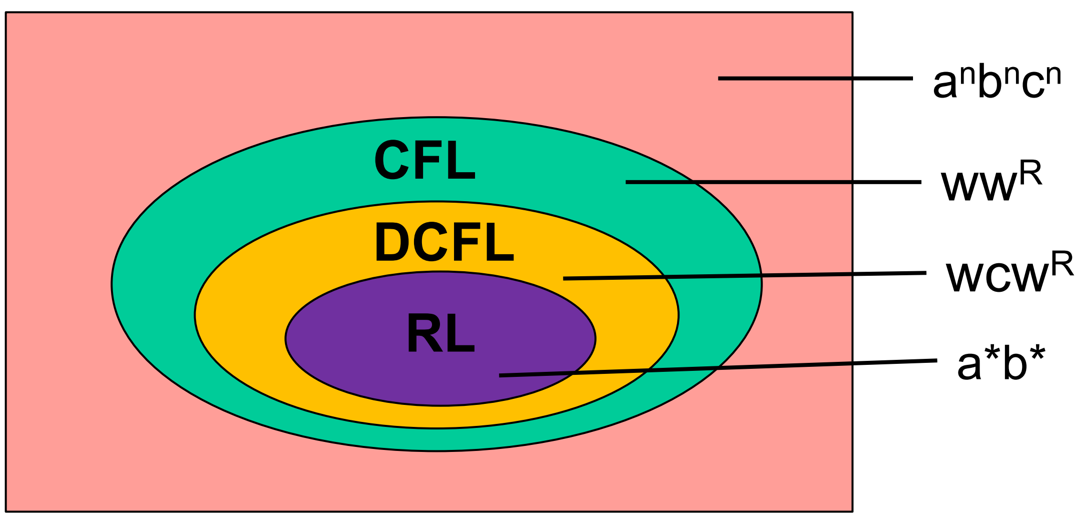
证明思路：用DPDA P可以模拟任一DFA A，也就是，已知DFA A，如何构造一个DPDA P的问题。

1. 设$A=(Q, \Sigma, \delta_A, q_0, F)$是一个DFA，构造DPDA P
   $P=(Q,\Sigma, \{Z_0\}, \delta_P, q_0, Z_0, F)$
2. 对于所有的Q中满足$\delta_A(q,a)=p$的状态对p和q，定义$\delta_P(q,a, Z_0)=\{(p, Z_0)\}$。 //栈顶恒等于$Z_0$

3. 通过对|w|进行归纳来证明：
$(q_0, w, Z_0) \overset{*}{\underset{P}{\vdash}} (p, \varepsilon, Z_0)$当且仅当 $\delta_A(q_0,w)=p$ 。

6.6 确定型下推自动机

- DPDA识别正则语言；
- DPDA识别上下文无关语言L_{wcwr}，所以DCFL语言类真包含正则语言；
- DPDA无法识别上下文无关语言L_{wwr}，所以DCFL真包含于CFL；



6.6 确定型下推自动机

定理6.8 DPDA P，语言L=L(P)，那么L有无歧义的CFG。

定理6.9 DPDA P，语言L=N(P)，那么L有无歧义的CFG。

- DPDA P接受的语言都是无歧义的，因此DPDA在语法分析中占重要地位；
- 但是，并非所有的非固有歧义的CFL都能被DPDA识别，如L_{wwr}有无歧义文法$S \rightarrow 0S0|1S1|\varepsilon$，但是，它不是DPDA能识别的。

小　结

1. 上下文无关文法CFG的定义，与上下文有关文法CSG的主要区别。
2. 语法分析树（Parse Tree）充分体现了CFL的结构特征（递归）。
3. **最左派生（leftmost derivation）与**最右归约(rightmost reduction)对应，**最右派生（rightmost derivation）与**最左归约(leftmost reduction) **对应。**
4. 文法的歧义性由语法分析树决定，与推导无关。有些文法是固有歧义的。
5. PDA = NFA + Stack，PDA是非确定性的。PDA可以按终态接受方式定义，也可以按空栈接受方式定义，两者是等价的。
6. 一个语言是上下文无关的，当且仅当存在一台下推自动机识别它。
7. DPDA接受的语言DCFL是无歧义的，广泛应用于语法分析器。