

第6章 上下文无关语言



6.1 上下文无关文法

6.2 语法分析树

6.3 文法和语言的歧义性

6.4 下推自动机

6.5 PDA与CFG的等价性

6.6 上下文无关文法的应用

乔姆斯基文法体系 (4型文法)



0型文法或短语结构文法 $\alpha \rightarrow \beta$

1型文法或上下文相关文法 $|\beta| \geq |\alpha|$

2型文法或
上下文无关文法 $\alpha \in V$

3型文法
或正则文法
 $A \rightarrow w \mid wB$



- 上下文无关文法和它所描述的上下文无关语言，在定义程序设计语言、语法分析、简化程序设计语言的翻译等方面有重要意义。
- 高级程序设计语言的绝大多数语法结构都可以用上下文无关文法 (CFG) 描述。**RL只适用于词法。**
- 近年来，上下文无关文法也被用来描述文档格式：XML 中使用的 DTD(文档类型定义)就是用来描述 Web 上的信息交换格式的。

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 $L = \{0^n 1^n \mid n \in \mathbb{N}\}$
 $G = (\{S\}, \{0, 1\}, P, S)$
 $P: S \rightarrow \varepsilon, S \rightarrow 0S1$

例2 $L = \{0^n 1^{2n+1} \mid n \in \mathbb{N}\}$
 $G = (\{S\}, \{0, 1\}, P, S)$
 $P: S \rightarrow 1, S \rightarrow 0S11$

6.1 上下文无关文法



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

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

6.1 上下文无关文法



例 5. $L = \{0^n 1^m \mid n, m \in \mathbb{N}\}$

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 $RL \subset CFL$ 。

思路: 令任意 $L \in RL$, 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 $|E| \leq 1$ 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 $E = E_1 E_2$: $S \rightarrow S_1 S_2$
- 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 $x_1 \dots x_k$, 当且仅当存在产生式规则 $A \rightarrow x_1 \dots x_k$ 。

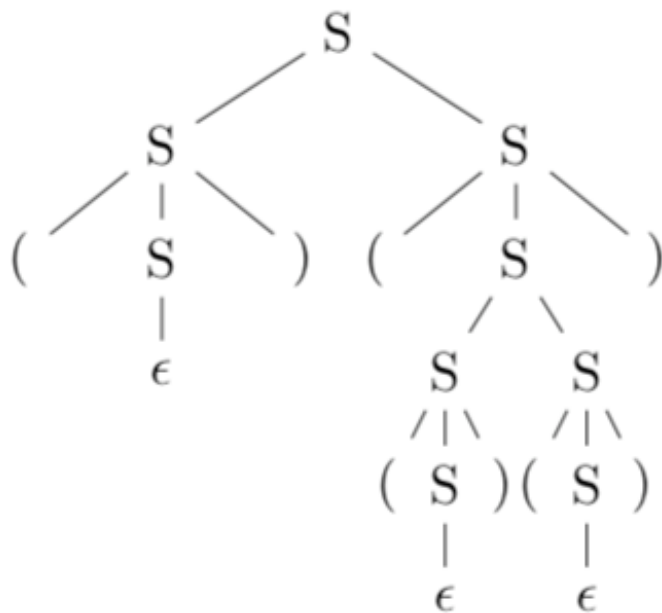
□ 语法分析树(parse tree)又称派生树(derivation tree)、语法树(syntax tree)。

6.2 语法分析树



例5. $S \rightarrow \varepsilon \mid (S) \mid SS$ 推出 $()((())())$

$$S \xRightarrow{lm} SS \xRightarrow{lm} (S)S \xRightarrow{lm} ()S \xRightarrow{lm} ()(S) \xRightarrow{lm} ()(SS) \xRightarrow{lm} ()((S)S) \xRightarrow{lm} ()(()S) \xRightarrow{lm} ()(()(S)) \xRightarrow{lm} ()((())())$$



6.2 语法分析树



例6.

□ 算术表达式的文法

$G_1:$

$$\begin{aligned} E &\rightarrow E+T \mid E-T \mid T \\ T &\rightarrow T * F \mid T / F \mid F \\ F &\rightarrow F \uparrow P \mid P \\ P &\rightarrow (E) \mid N(L) \mid \text{id} \\ N &\rightarrow \sin \mid \cos \mid \exp \mid \text{abs} \mid \log \mid \text{int} \\ L &\rightarrow L, E \mid E \end{aligned}$$

□ 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

$E \Rightarrow E + T$
 $\Rightarrow T + T$
 $\Rightarrow F + T$
 $\Rightarrow P + T$
 $\Rightarrow x + T$
 $\Rightarrow x + T / F$
 $\Rightarrow x + F / F$
 $\Rightarrow x + P / F$
 $\Rightarrow x + x / F$
 $\Rightarrow x + x / F \uparrow P$
 $\Rightarrow x + x / P \uparrow P$
 $\Rightarrow x + x / y \uparrow P$
 $\Rightarrow x + x / y \uparrow 2$

$E \Rightarrow E + T$
 $\Rightarrow E + T / F$
 $\Rightarrow E + T / F \uparrow P$
 $\Rightarrow E + T / F \uparrow 2$
 $\Rightarrow E + T / P \uparrow 2$
 $\Rightarrow E + T / y \uparrow 2$
 $\Rightarrow E + F / y \uparrow 2$
 $\Rightarrow E + P / y \uparrow 2$
 $\Rightarrow E + x / y \uparrow 2$
 $\Rightarrow T + x / y \uparrow 2$
 $\Rightarrow F + x / y \uparrow 2$
 $\Rightarrow P + x / y \uparrow 2$
 $\Rightarrow x + x / y \uparrow 2$

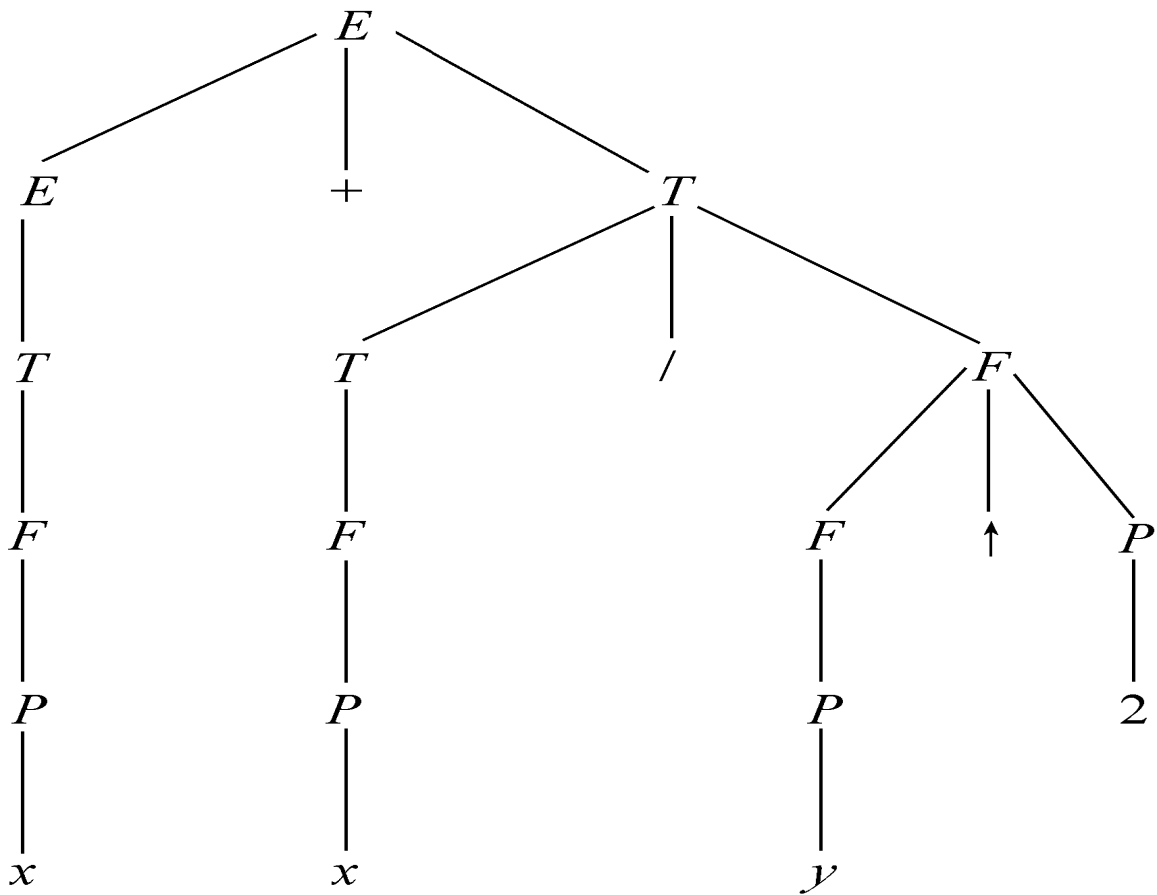
$E \Rightarrow E + T$
 $\Rightarrow T + T$
 $\Rightarrow T + T / F$
 $\Rightarrow F + T / F$
 $\Rightarrow F + T / F \uparrow P$
 $\Rightarrow P + T / F \uparrow P$
 $\Rightarrow x + T / F \uparrow P$
 $\Rightarrow x + F / F \uparrow P$
 $\Rightarrow x + F / F \uparrow 2$
 $\Rightarrow x + F / P \uparrow 2$
 $\Rightarrow x + P / P \uparrow 2$
 $\Rightarrow x + P / y \uparrow 2$
 $\Rightarrow x + x / y \uparrow 2$



6.2 语法分析树



句子 $x + x / y \uparrow 2$ 的派生树---对应三种不同的派生



6.2 语法分析树



定义6.5 边缘（或结果）

派生树 T 的所有叶子顶点从左到右依次标记为 X_1, X_2, \dots, X_n ，则称符号串 $X_1X_2\dots X_n$ 是 T 的边缘或结果。

- 一个文法可以有多棵派生树，它们可以有不同的边缘。

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，或边缘。
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(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 有一棵结果为 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。

6.3 文法和语言的歧义性



算术表达式的二义性文法

$G_2: \quad E \rightarrow E + E \mid E - E \mid E / E \mid E * E$

$E \rightarrow E \uparrow E \mid (E) \mid N(L) \mid id$

$N \rightarrow \sin \mid \cos \mid \exp \mid \text{abs} \mid \log \mid \text{int}$

$L \rightarrow L, E \mid E$

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:

$$E \Rightarrow E+E$$

$$\Rightarrow x+E$$

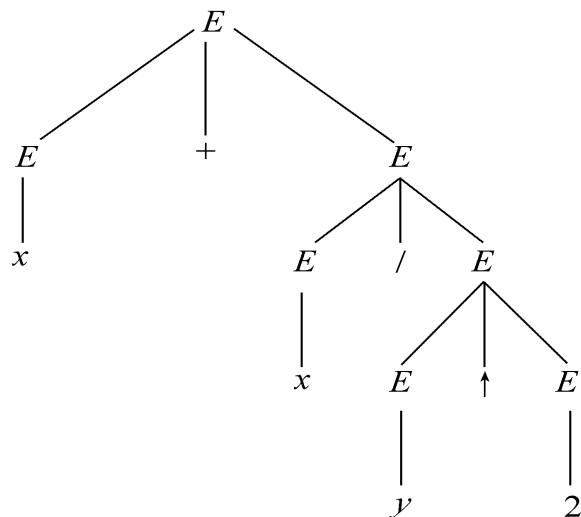
$$\Rightarrow x+E/E$$

$$\Rightarrow x+x/E$$

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

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

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



$$E \Rightarrow E/E$$

$$\Rightarrow E+E/E$$

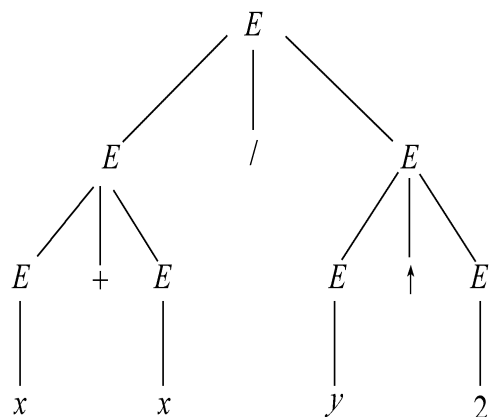
$$\Rightarrow x+E/E$$

$$\Rightarrow x+x/E$$

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

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

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



$$E \Rightarrow E \uparrow E$$

$$\Rightarrow E/E \uparrow E$$

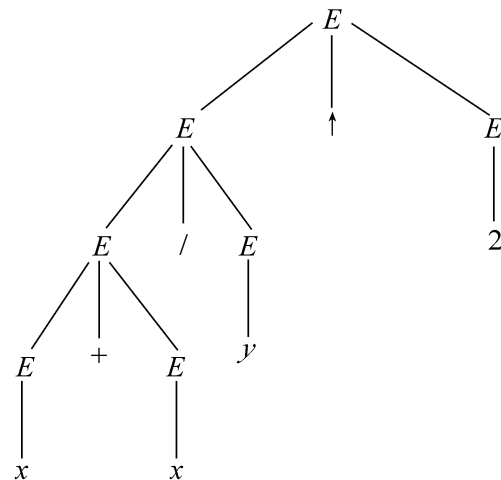
$$\Rightarrow E+E/E \uparrow E$$

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

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

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

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



6.3 文法和语言的歧义性



定义6.7 二义性/歧义性(ambiguity)

CFG $G = (V, T, P, S)$, 如果存在 $w \in L(G)$, w 至少有**两棵不同的语法分析树**, 则称 G 是**二义性的或歧义的**。否则 G 为**非二义性的**。

- G_2 是二义性的, 但 G_1 是非二义性的, 但二者是等价的, 即 $L(G_1) = L(G_2)$
- 判定任给 CFG G 是否为二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

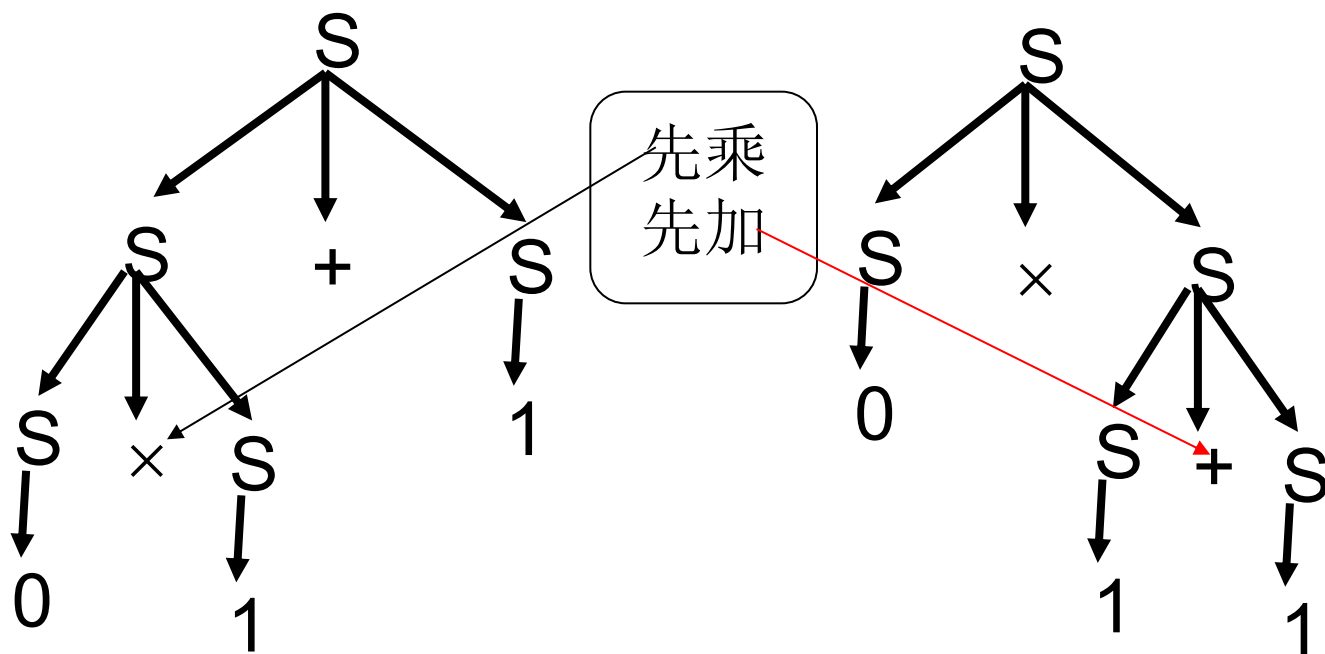


例7. 二义性

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

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

➤ $A \rightarrow BC$ 一分为二

➤ $A \rightarrow x$ 或终极化

➤ $S \rightarrow \varepsilon$

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.

3. 符合乔姆斯基范式的CFG不存在二义性。

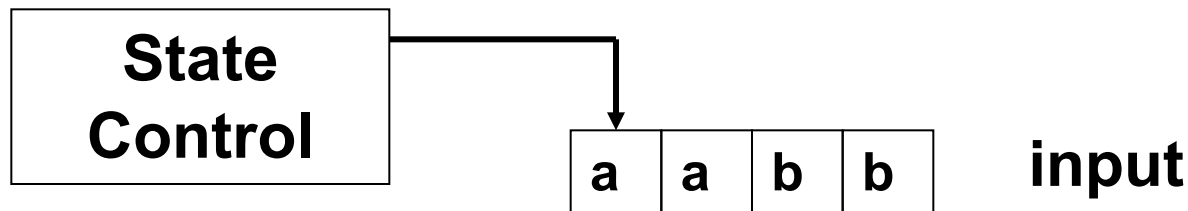
6.4 下推自动机



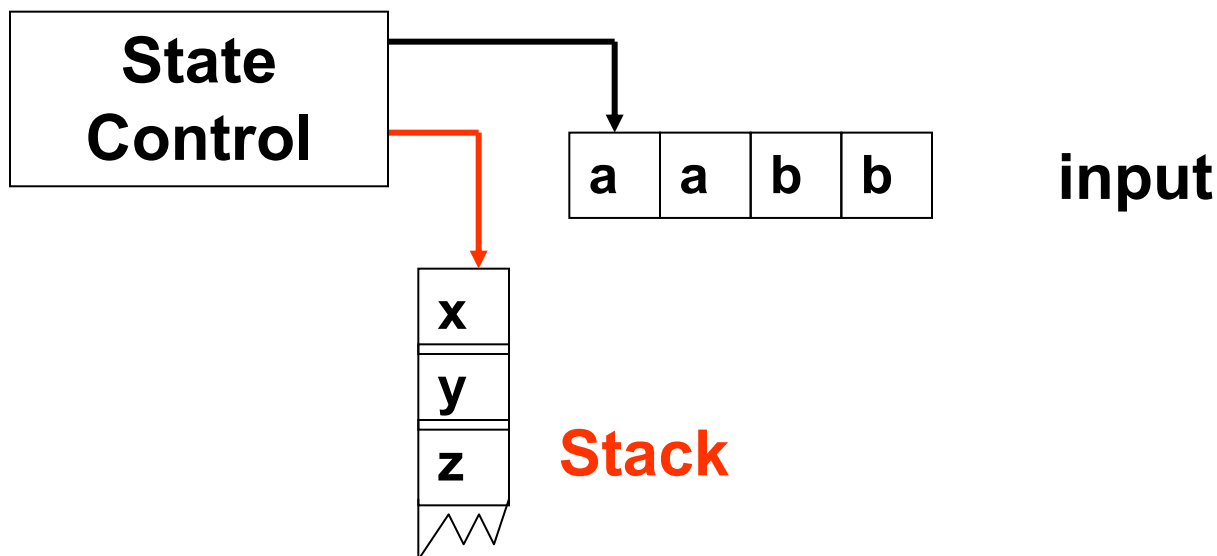
关于PDA (pushdown automata)

1. NFA+Stack can recognizes some nonregular languages (CFL)
2. $PDA \leftrightarrow CFG$ 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

6.4 下推自动机



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

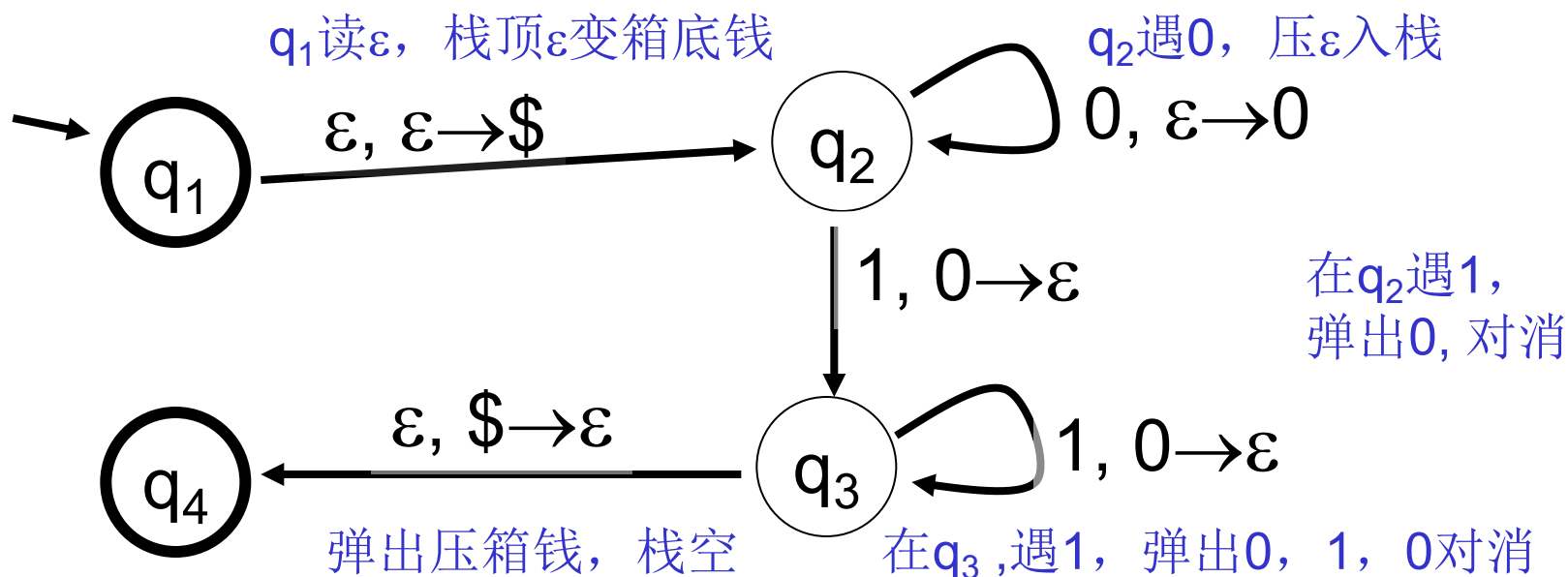
6.4 下推自动机



Exp8: A PDA recognizes language $\{0^n 1^n | n \geq 0\}$.

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: The PDA 无法判断何时为空. 办法: 先将\$压栈。



a, $b \rightarrow c$ 的含义: read 'a', pop 'b', push 'c'

$a = \epsilon$ means read nothing; $b = \epsilon$ means no pop; $c = \epsilon$ means push nothing

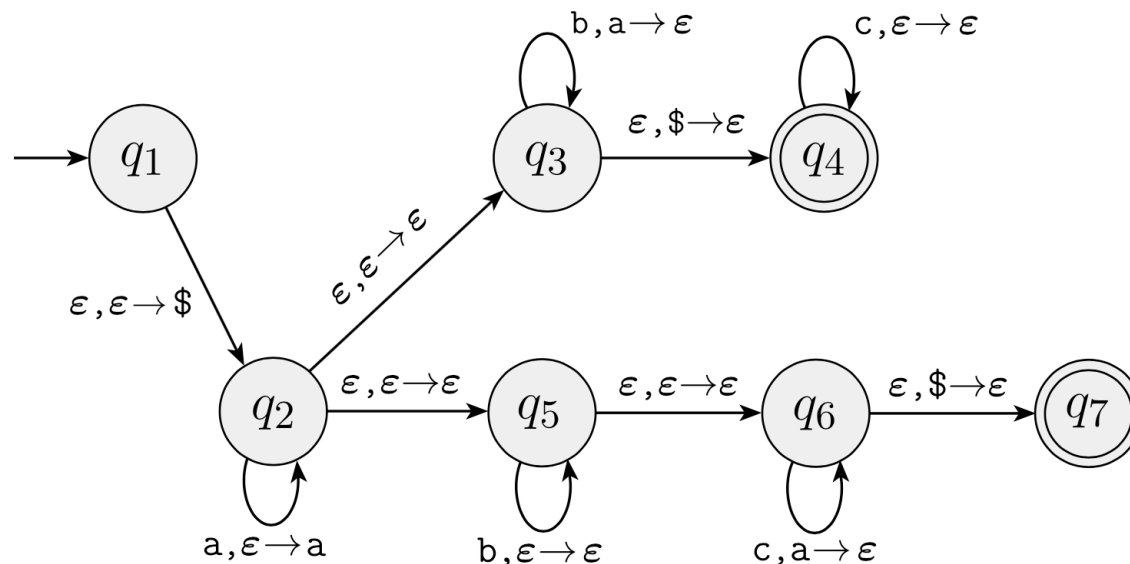
6.4 下推自动机



例9. 识别语言 $\{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i=j \text{ or } i=k\}$ 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: $P = (Q, \Sigma, q_0, \delta, F, \Gamma, Z_0)$ 。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, 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:

$\delta(q, a, Z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \dots, (p_m, \gamma_m)\}$, 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



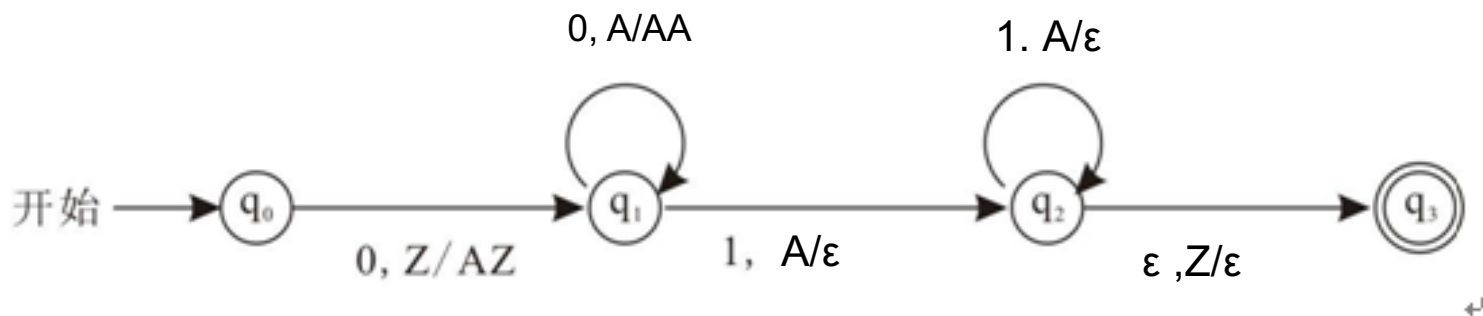
例10. 构造一个PDA按终结状态方式接受语言 $\{0^n 1^n \mid n \geq 1\}$ 。

□ 形式化定义:

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

$\delta(q_0, 0, Z) = \{(q_1, AZ)\}$	在栈中加入A
$\delta(q_1, 0, A) = \{(q_1, AA)\}$	在栈中加入A
$\delta(q_1, 1, A) = \{(q_2, \varepsilon)\}$	遇1消去栈顶符号
$\delta(q_2, 1, A) = \{(q_2, \varepsilon)\}$	遇1消去栈顶符号
$\delta(q_2, \varepsilon, Z) = \{(q_3, Z)\}$	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 $(q_0, w, Z_0) \vdash^*(q, \varepsilon, \alpha)$ 。

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 $(q_0, w, Z_0) \vdash^*(q, \varepsilon, \varepsilon)$ 。

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 \leftrightarrow 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 \rightarrow 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 \rightarrow 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。 $CFL L \rightarrow CFG G \rightarrow PDA P$

Proof Idea: 如何用PDA P模拟CFG G?

$$L = \{ 0^n 1^n \mid n \geq 0 \}$$

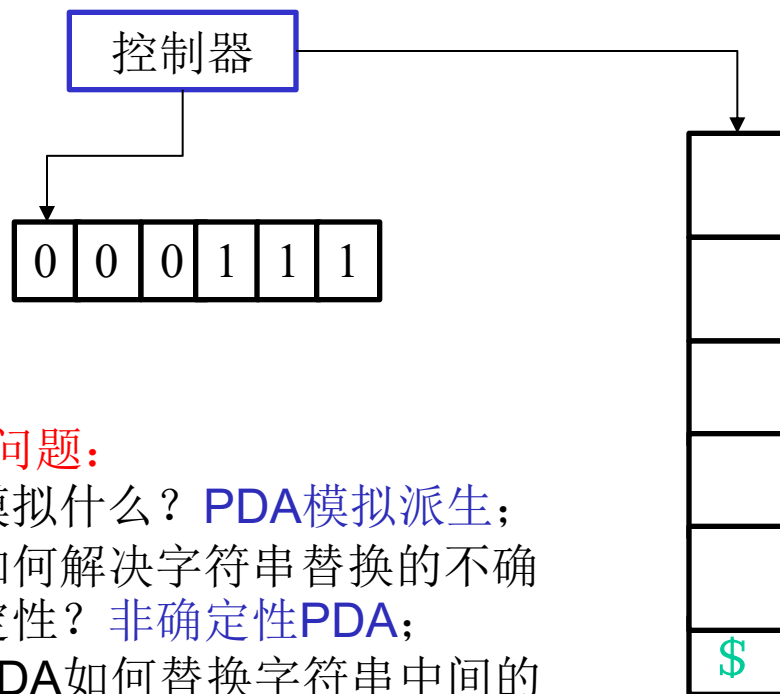
$$G: S \rightarrow 0S1 \mid \varepsilon$$

文法G产生 $w=000111$ 的过程如下:

$$\begin{aligned} S &\rightarrow 0S1 && \textcircled{1} \\ &\rightarrow 00S11 && \textcircled{2} \\ &\rightarrow 000S111 && \textcircled{3} \\ &\rightarrow 000111 && \textcircled{4} \end{aligned}$$

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 A , 如何构造一个等价的PDA P

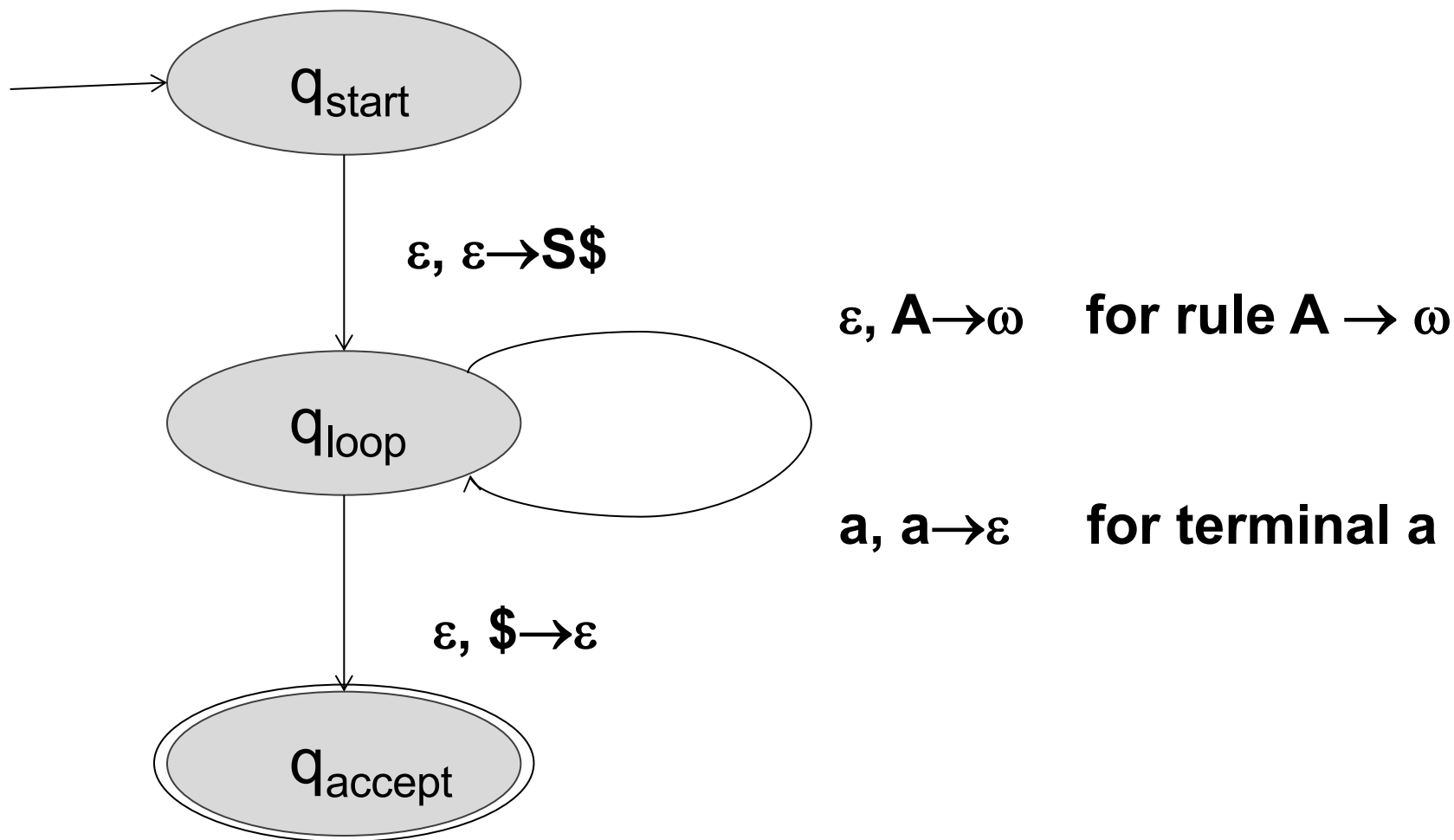
1. Place the marker symbol $\$$ and the start variable on the stack;
2. Repeat the following steps forever.
 - ① If the top of stack is a variable symbol A , **nondeterministically** select one of the rules for A and substitute A by the string on the right-hand side of the rule.
 - ② If the top of stack is a terminal symbol a , read the next symbol from the input and **compare it to a** . if they match, repeat. If they do not match, reject on this branch of the nondeterminism.
 - ③ If the top of stack is the symbol $\$$, enter the accept state. Doing so accepts the input if it has all been read.



6.5 PDA与CFG的等价性



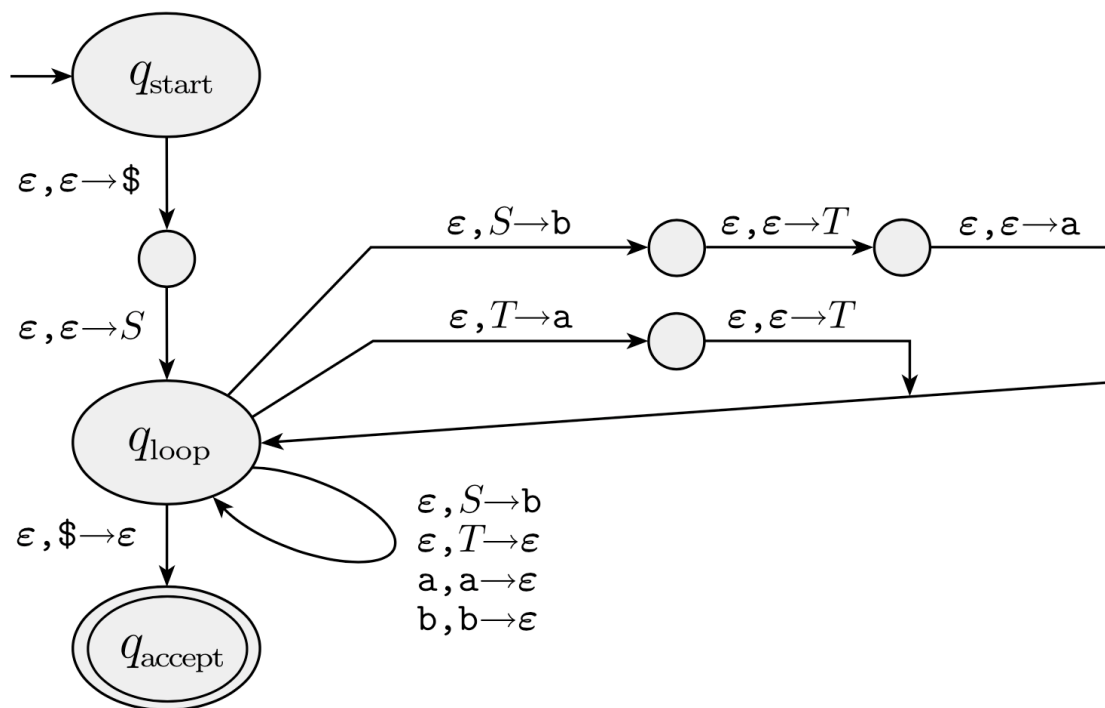
如何把文法G转换成PDA



6.5 PDA与CFG的等价性



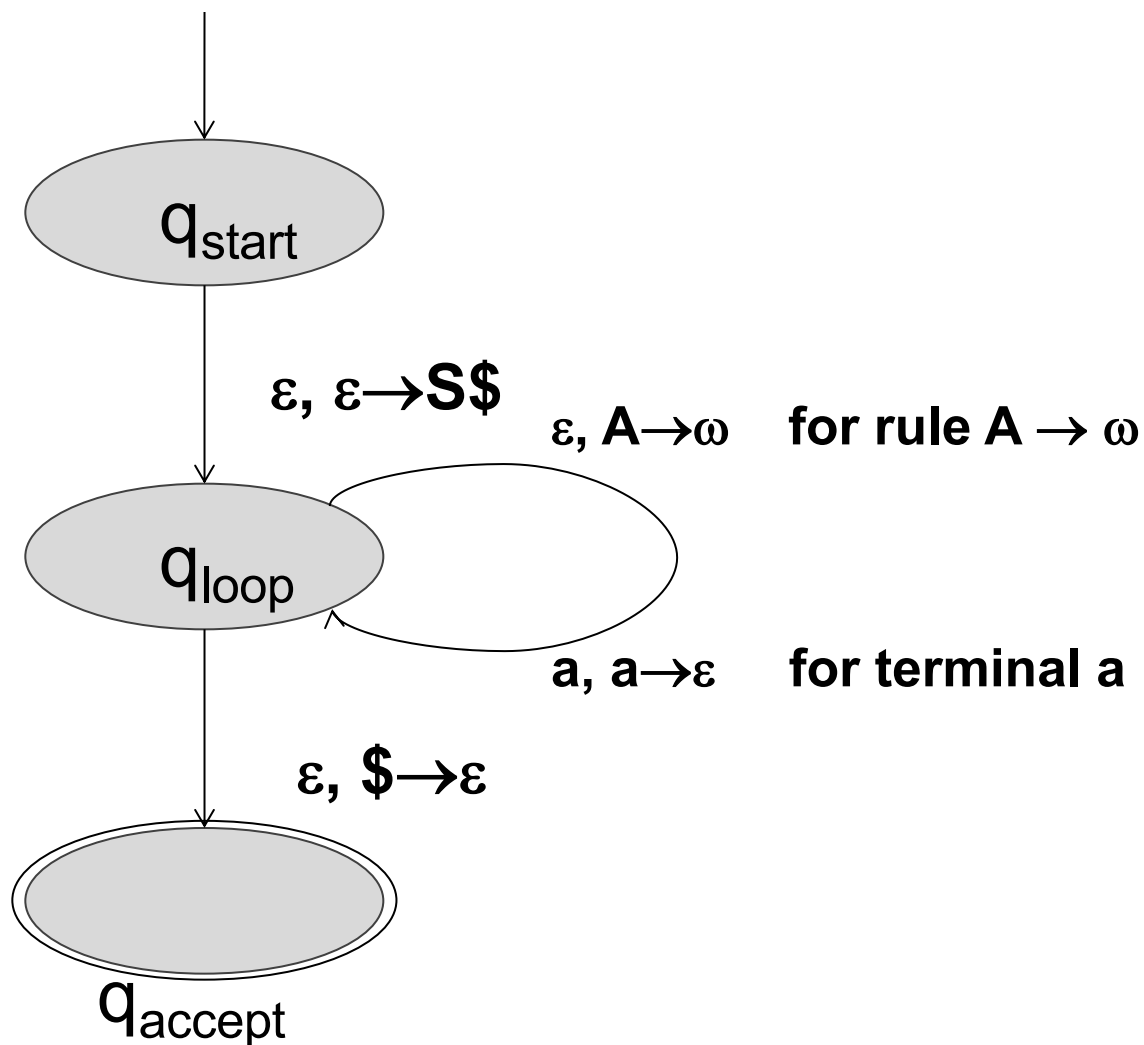
EXP: 把CFG G 转换成PDA P , 其中
 G :

$$S \rightarrow aTb \mid b$$
$$T \rightarrow Ta \mid \varepsilon$$


6.5 PDA与CFG的等价性



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



6.5 PDA与CFG的等价性



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

6.5 PDA与CFG的等价性



来自张海同学的问题：语言 $L = \{xwx^R \mid x, w \in \{0,1\}^+\}$ 到底是正则的，还是非正则的？

$\{xwx^R \mid x, w \in \{0,1\}^+\}$ 不是正则语言

取 p 为泵长，不妨设 $x = 01^{\frac{p}{2}-1}$, $w = 1$

那么 $xwx^R = 01^{p-1}0$

使 $s = xyz = 01^{p-1}0$, $\because |xy| \leq p \therefore x = 0, y = 1^{p-1}, z = 0$

此时 $xz = 00 \notin L$

因此该语言不是正则语言

6.5 PDA与CFG的等价性



来自刘泽润同学的问题：

Pumping lemma: For every regular language L , there is a *pumping length* p , such that for any string $s \in L$ and $|s| \geq p$, we can write $s = xyz$ with

- 1) $x y^i z \in L$ for every $i \in \{0, 1, 2, \dots\}$ 为什么打圈？
- 2) $|y| > 0$
- 3) $|xy| \leq p$ 什么时候打圈？

这是我们课件上对泵引理的一个解释，在最一开始的学习过程中，我认为 p 可以有一个确定值，确定值等于可以接受 L 的DFA的状态数。因为，当 s 的长度超过了状态数，就可以进行打圈。

经过之后的学习，我们知道了DFA是有一个最小DFA存在，那么我们这个泵长 p 是不是等于可以接受 L 的最小DNA的状态数，或者是小于等于这个状态数呢？

以上就是我关于泵引理的一些小疑问，希望老师批评指正！我在群中添加了您的微信，希望您可以通过一下好友申请进行答疑。

6.5 PDA与CFG的等价性



$L=01^*, p=?$



最小值为2吧

为什么?



星期六 23:38



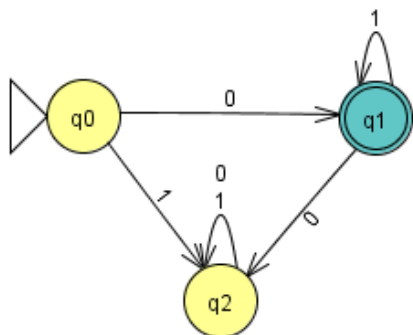
可以用一个状态数为2的DFA来表示



等等, 但他好像不是一个DFA



应该是3



DFA画出来是这个样子的

$s=xyz=0$, 这时就泵不出1了,
 $s=xyz=01$,这是可以对1pump。
所以, 最小的泵长度是2。



6.5 PDA与CFG的等价性



$L=1^*0^*, p=?$

1、如果 $p=0$ ，也就是 $s=\varepsilon$ ，这时，无论如何pump出含有1或0的字符串；

2、 $p=1, s=xyz=1, x=\varepsilon, y=1, z=\varepsilon$ ，请问 xy^iz 属于 L 吗

3、所以 $p \geq 1$

6.5 PDA与CFG的等价性



通过上述讨论，我们得出什么结论？

