# 第五讲　上下文无关文法（二）

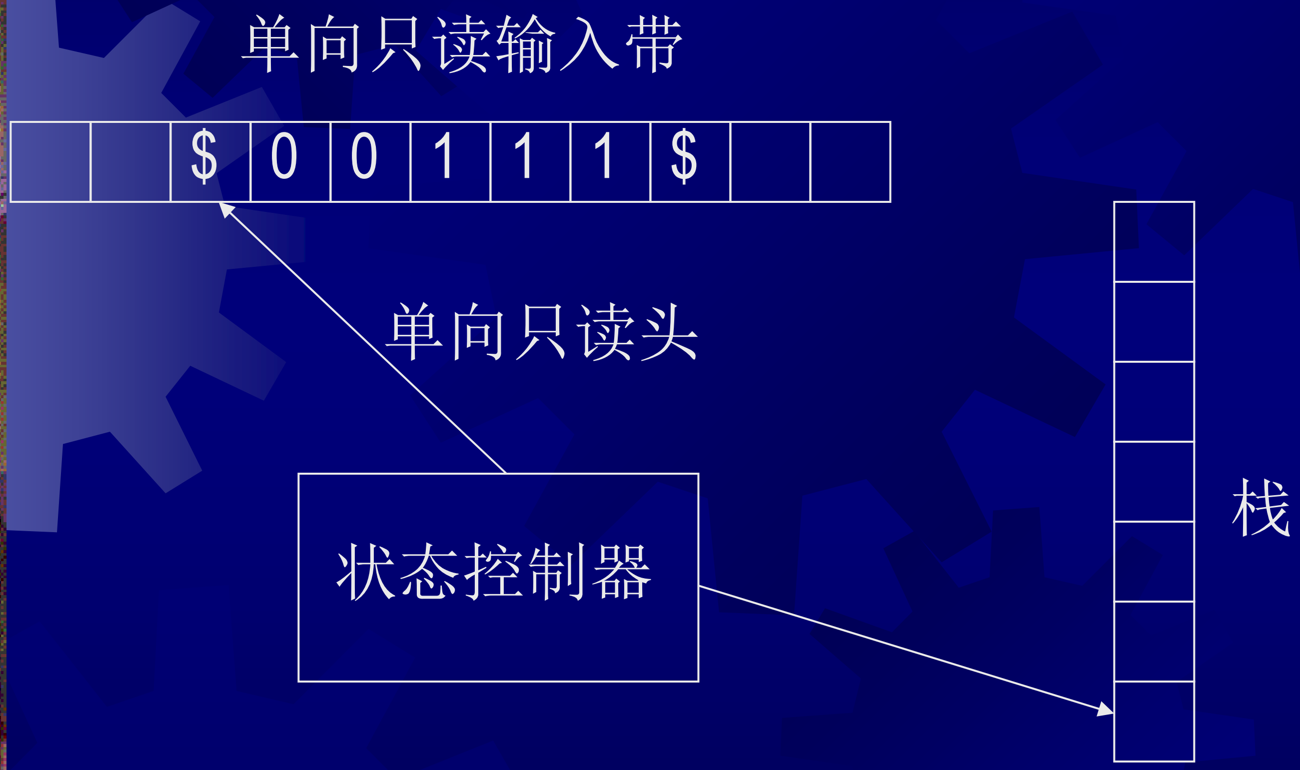
罗　欢
Email：hluo@fzu.edu.cn

# 主要内容

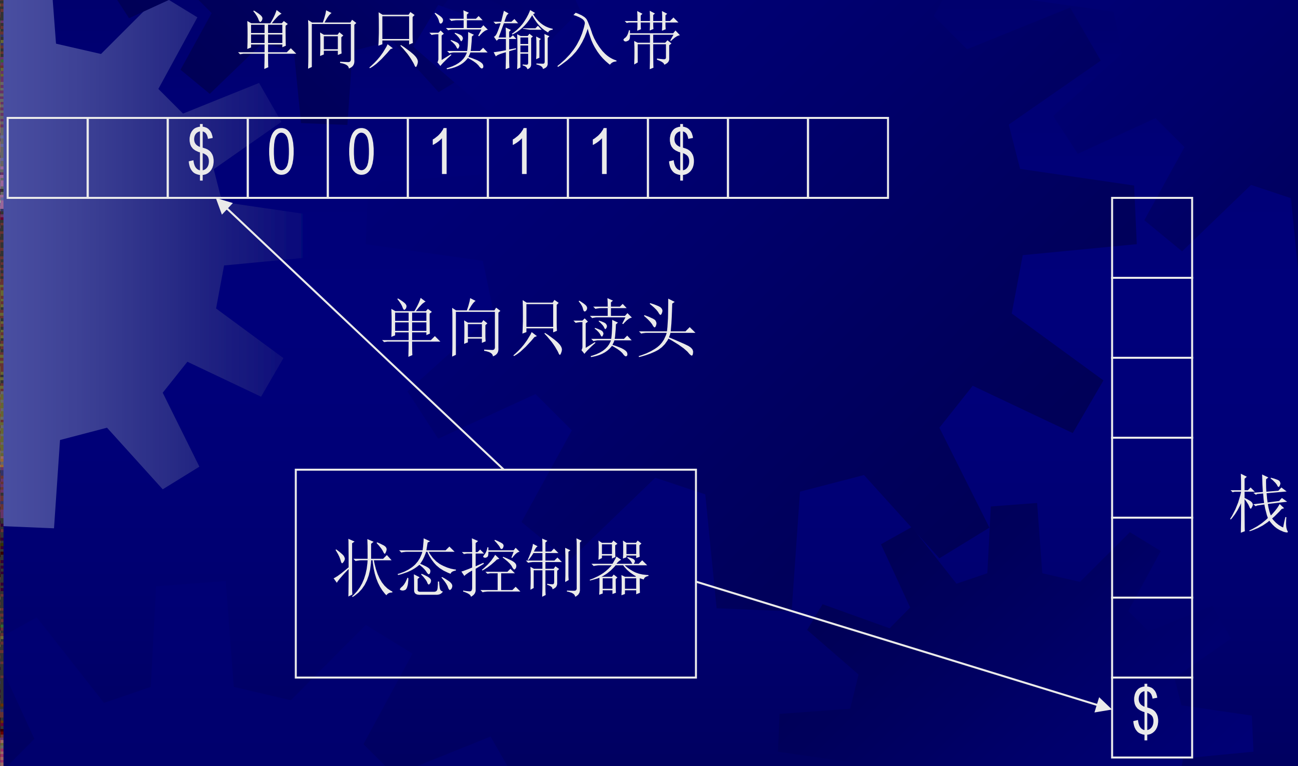
- 上下文无关语言
  - 上下文无关文法(CFG)
  - 歧义性
  - 乔姆斯基范式
- 下推自动机(PDA)
- PDA与CFG的等价性
- 泵引理 (略)

# 下推自动机

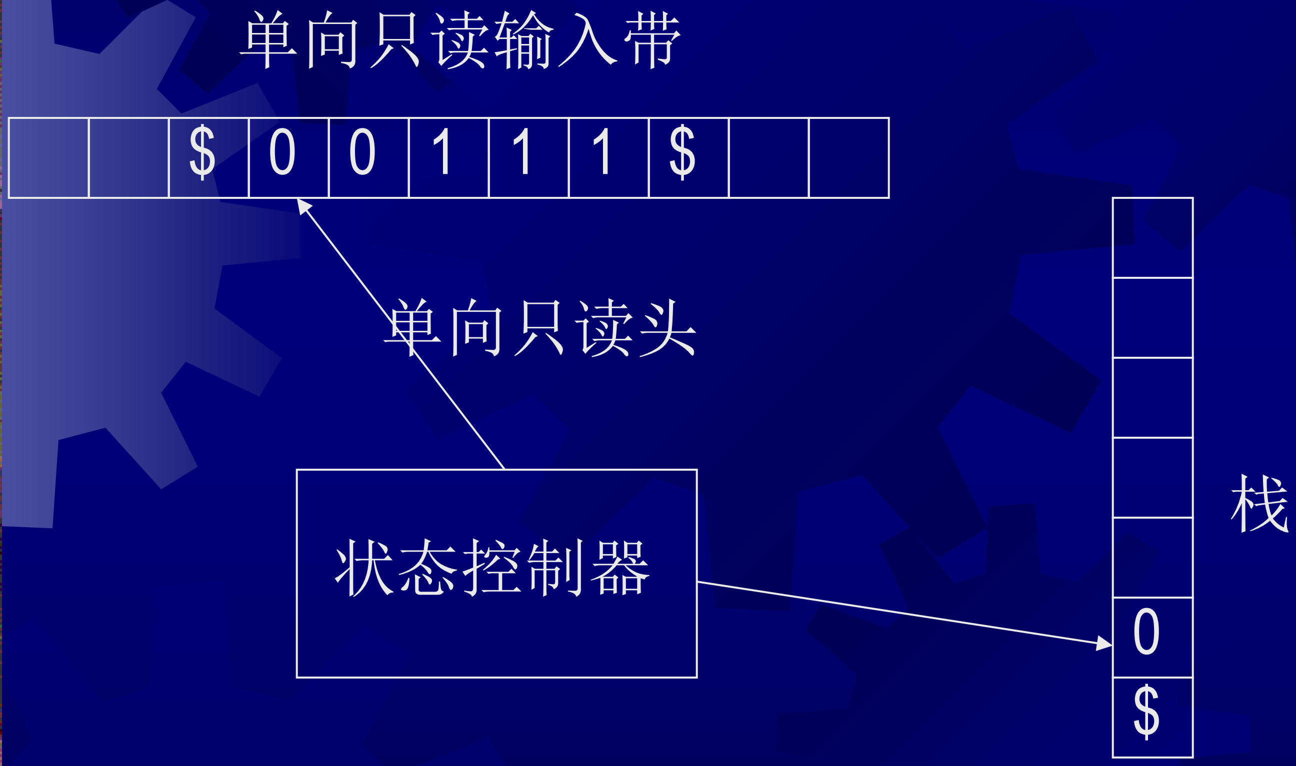
# 下推自动机(PDA)



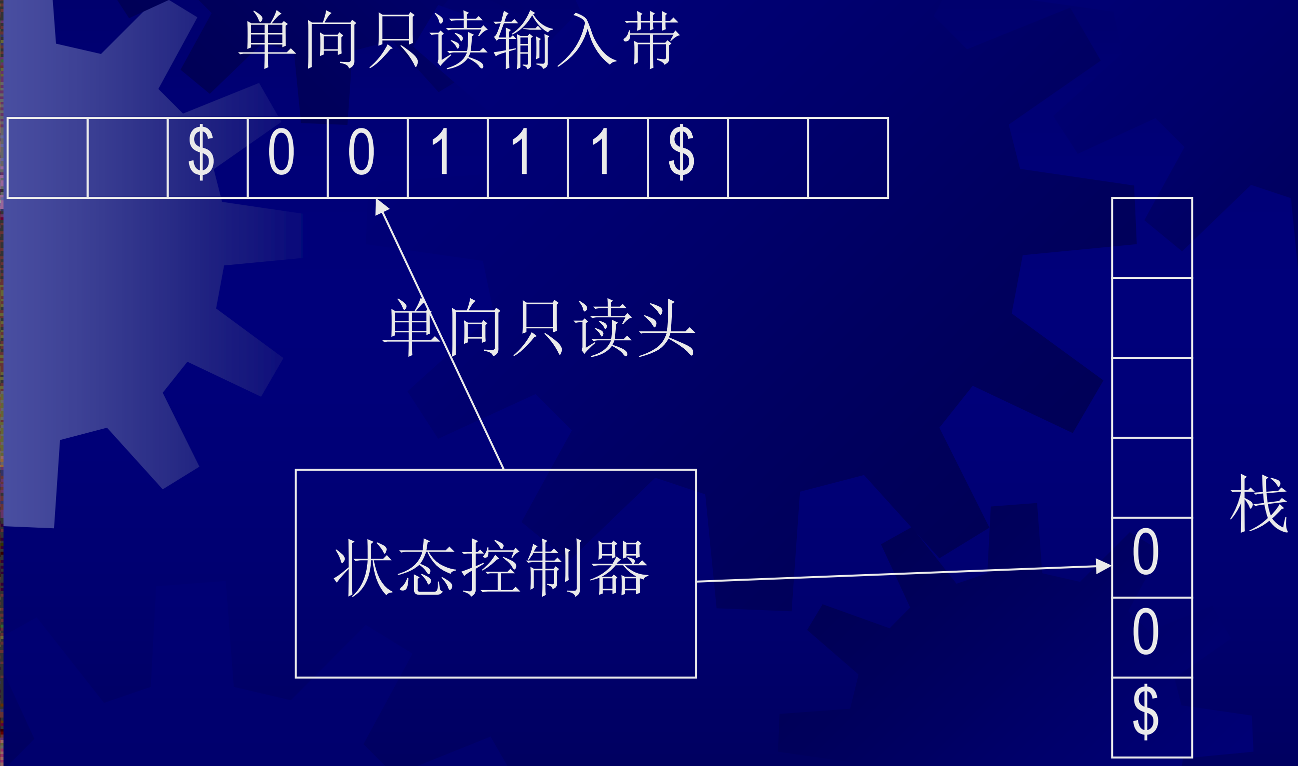
# 下推自动机(PDA)



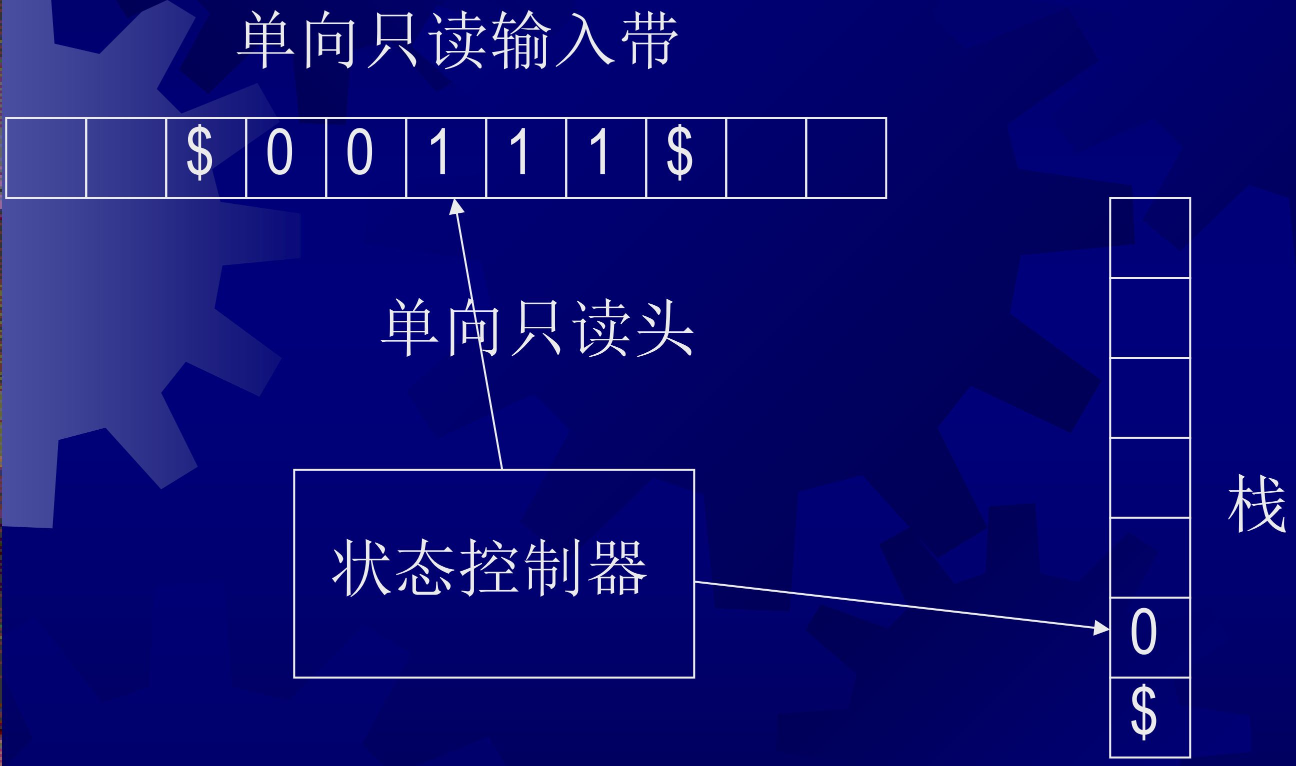
# 下推自动机(PDA)



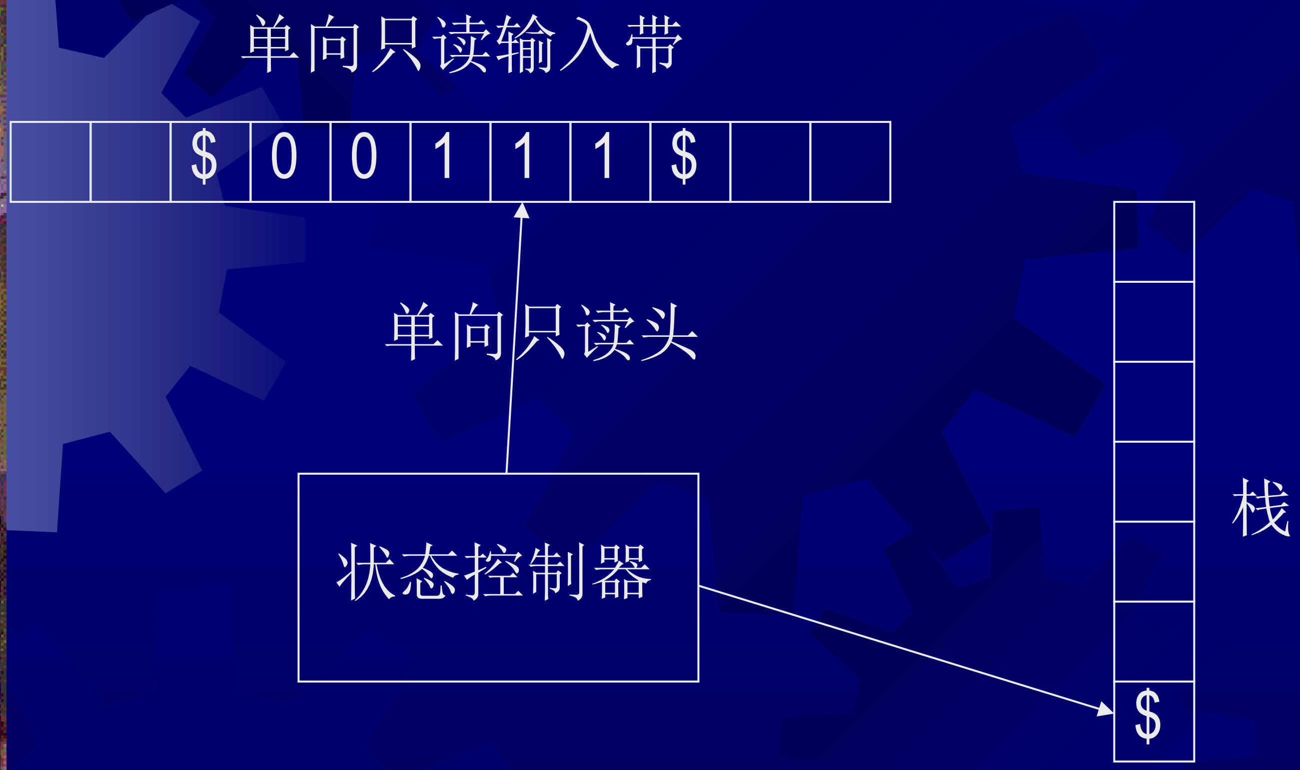
# 下推自动机(PDA)



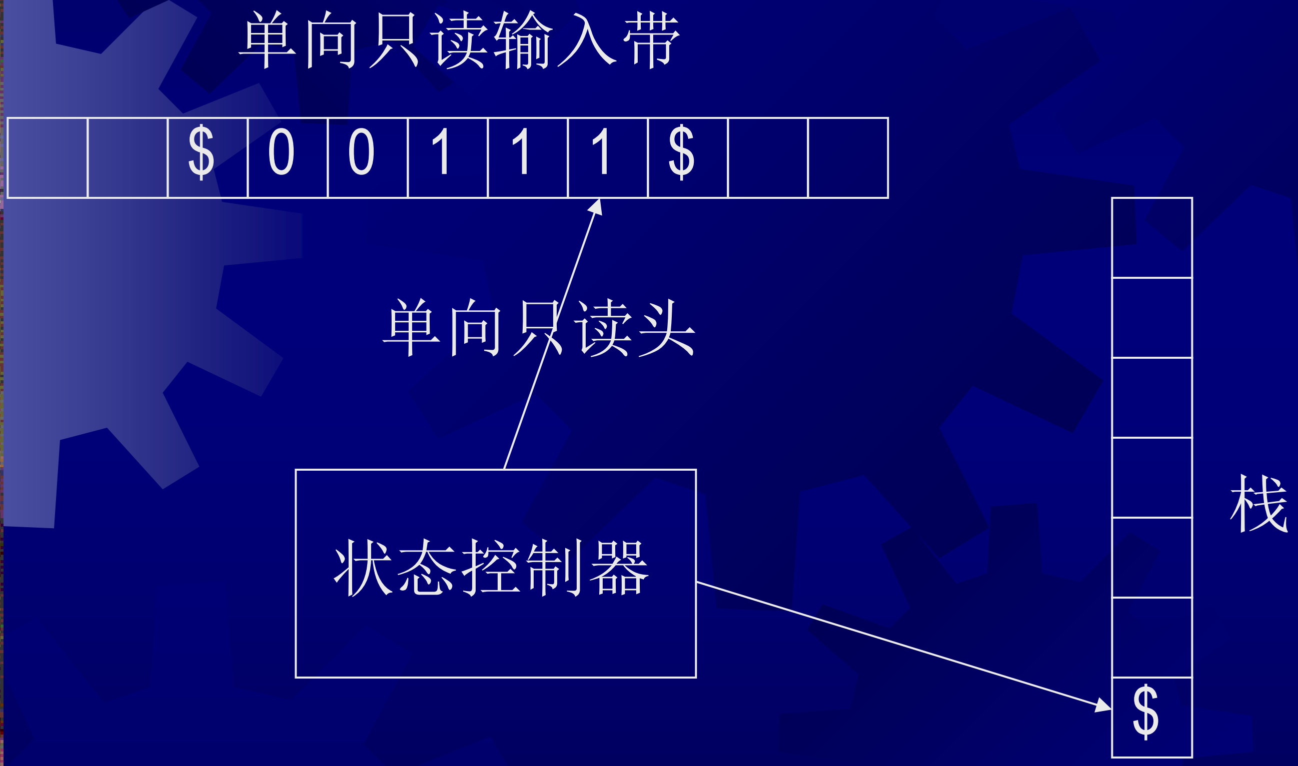
# 下推自动机(PDA)



# 下推自动机(PDA)



# 下推自动机(PDA)



# 例

- $\{ 0^n1^n \mid n \geq 0 \}$
  - 读输入中的符号,
    - 每读一个0,把一个0推入栈
    - 一旦看见1之后,
      就每读一个1,把一个0弹出栈
      - 如果当读完输入串时,
        恰好排空栈中的0, 则接受; 否则拒绝
      - 如果在还有1没有读完时, 栈就排空,
        则拒绝
      - 如果在1读完时, 栈中还有0, 则拒绝
      - 如果0出现在1的后面, 则拒绝

# 非确定性

- $\{ 0^n1^n \mid n\geq 0 \}$
  - 确定型PDA, 即DPDA
- $\{ a^nb^nc^m, a^nb^mc^n \mid m,n\geq 0 \}$
  - 固有歧义
  - 非确定型PDA, 即NPDA, 简称PDA
- 一般说PDA指的是非确定型PDA

# 下推自动机的定义

# PDA的形式定义

- 定义3.8: PDA $M=(Q,\Sigma,\Gamma,\delta,q_0,F)$,

其中

1) $Q$: 有穷状态集

2) $\Sigma$: 输入字母表, $\Sigma_\varepsilon=\Sigma\cup\{\varepsilon\}$

3) $\Gamma$: 栈字母表, $\Gamma_\varepsilon=\Gamma\cup\{\varepsilon\}$

4) $\delta$: $Q\times\Sigma_\varepsilon\times\Gamma_\varepsilon\rightarrow P(Q\times\Gamma_\varepsilon)$,

转移函数

5) $q_0\in Q$: 初始状态

6) $F\subseteq Q$: 接受状态(终结状态)集

# PDA计算的形式定义

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

输入$w=w_1w_2\ldots w_m$, $w_i\in\Sigma_\varepsilon$

- 计算: 状态-栈符号串序列

$(r_0,s_0), (r_1,s_1),\ldots, (r_m ,s_m)$,

其中 $r_i\in Q$, $s_i\in\Gamma^*$, 满足

1) $(r_0,s_0)=(q_0,\varepsilon)$;

2) $(r_{i+1},b)\in\delta(r_i,w_{i+1},a)$;

其中 $s_i=at$; $s_{i+1}=bt$,

$a,b\in\Gamma_\varepsilon$, $t\in\Gamma^*$

# PDA计算的形式定义

- 接受计算:

  3) $r_m \in F$;

  (或者同时要求$s_m=\varepsilon$)

- M接受w:

  M对输入w存在接受计算

- M(识别,接受)的语言:

  $L(M) = \{ x \mid M接受x \}$

# 下推自动机的例子

# 例3.9

- $L(M_1)=\{ 0^n1^n \mid n\geq 0 \}$

# 例3.9

- $L(M_1)=\{\ 0^n1^n \mid n\geq 0\ \}$

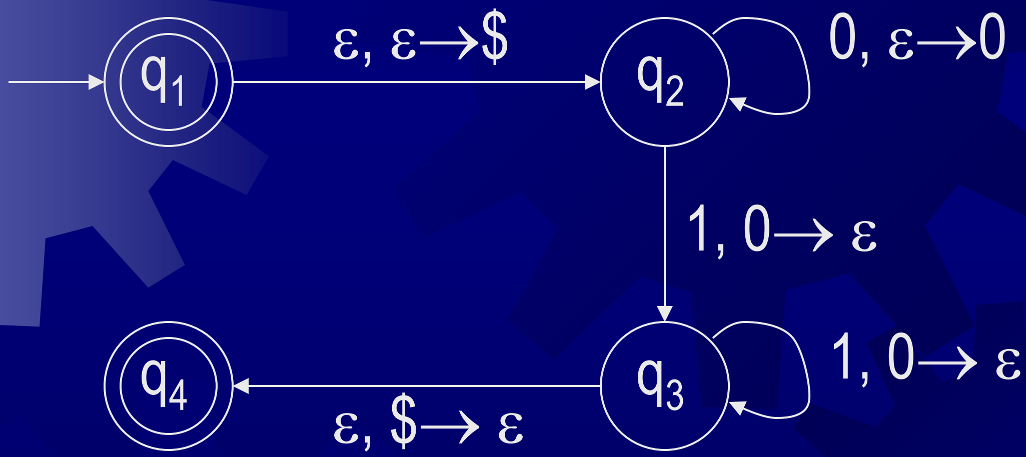
  $M_1=(\{q_1,q_2,q_3,q_4\},\{0,1\},\{0,\$\},\delta,q_1,\{q_1,q_4\})$

  δ表

| 输入 | | 0 | | | 1 | | | ε | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 栈 | | 0 | $\$$ | ε | 0 | $\$$ | ε | 0 | $\$$ | ε |
| 状态 | $q_1$ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | $\{(q_2,\$)\}$ |
| | $q_2$ | ∅ | ∅ | $\{(q_2,0)\}$ | $\{(q_3,\varepsilon)\}$ | ∅ | ∅ | ∅ | ∅ | ∅ |
| | $q_3$ | ∅ | ∅ | ∅ | $\{(q_3,\varepsilon)\}$ | ∅ | ∅ | ∅ | $\{(q_4,\varepsilon)\}$ | ∅ |
| | $q_4$ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |

# 例3.9 (用$判断是否空栈)

- $L(M_1)=\{ 0^n1^n \mid n\geq 0 \}$




$M_1$

# 例3.10

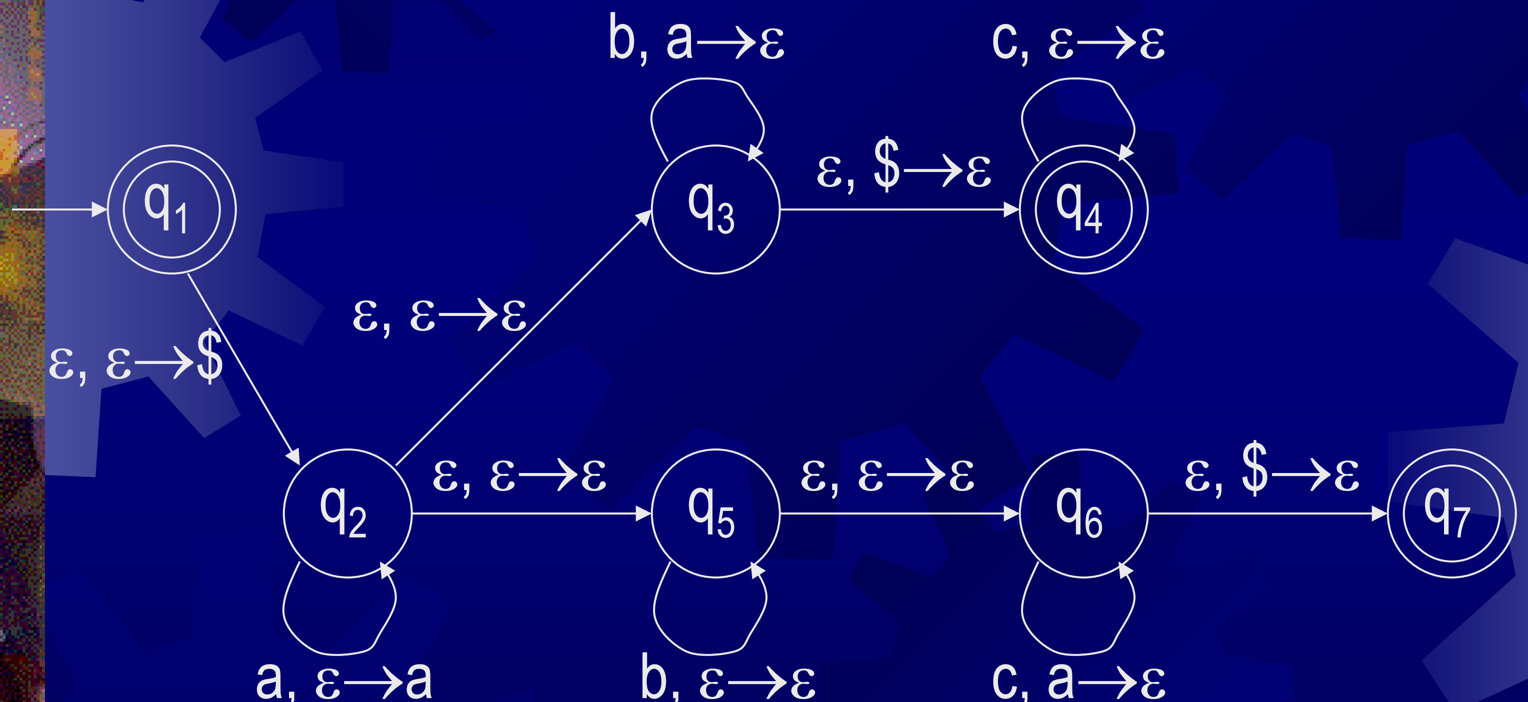
- $L(M_2)=\{ a^nb^nc^m, a^nb^mc^n \mid m,n\geq 0 \}$,

# 例3.10

- $L(M_2)=\{ a^n b^n c^m, a^n b^m c^n \mid m,n\geq 0 \}$,
  - 先读a,并且把a推入堆栈
  - 利用非确定性,
    猜想a是与b匹配还是与c匹配
  - 与b匹配
    - 每读到一个输入符号b,就从堆栈中弹出一个a,若从输入中读b结束时堆栈排空,则读若干个c后接受; 否则拒绝
  - 与c匹配
    - 读若干个b后,每读到一个输入符号c,就从堆栈中弹出一个a, 若输入结束时堆栈排空,则接受;否则拒绝

# 例3.10

- $L(M_2)=\{\ a^nb^nc^m,\ a^nb^mc^n \mid m,n\geq 0\ \}$,




M2

# 例3.11

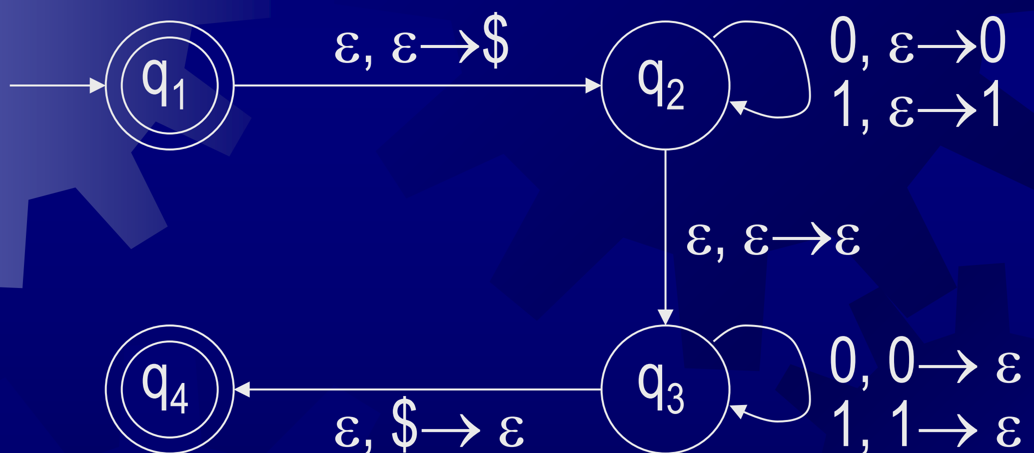
- $L(M_3)=\{ ww^R \mid w \in \{0,1\}^* \}$

# 例3.11

- $L(M_3)=\{ ww^R \mid w\in\{0,1\}^* \}$
  - 开始时,把读到的符号推入堆栈中
  - 在每一步,非确定性地猜测已到达字符串的中点,然后变成
  - 把读到的每一个符号弹出堆栈,检查在输入中和在堆栈顶读到的符号是否一样
  - 如果它们总是一样的,并且输入结束时堆栈同时排空,则接受;否则拒绝

# 例3.11

- $L(M_3)=\{ ww^R \mid w \in \{0,1\}^* \}$




$M_3$

# 总结

- 概念
  - 上下文无关文法(CFG)
  - 上下文无关语言(CFL)
  - 派生,最左派生,固有歧义性
  - 乔姆斯基范式(CNF)
  - 下推自动机(PDA)
- 方法
  - 设计CFG, 设计PDA
  - 求等价CNF

# 内容提要

- 等价性
  - CFG与PDA互相转化
- 泵引理
  - 证明非上下文无关语言
- 封闭性
  - CFL对正则运算封闭
  - CFL对交、补不封闭

# PDA与CFG等价性

# PDA与CFG等价性

- **定理3.12: 一个语言是CFL, 当且仅当存在PDA识别这个语言.**

# PDA与CFG等价性

- **定理3.12:** 一个语言是CFL,
当且仅当存在PDA识别这个语言.

- **引理3.13:** 如果一个语言是CFL,
则存在PDA识别这个语言.

# PDA与CFG等价性

- 定理3.12: 一个语言是CFL,
当且仅当存在PDA识别这个语言.

- 引理3.13: 如果一个语言是CFL,
则存在PDA识别这个语言.

- 引理3.15: 如果一个语言被PDA
识别, 则这个语言是CFL.

# 从CFG构造PDA的算法

# 引理3.13

- **引理3.13: 如果一个语言是CFL, 则存在PDA识别这个语言.**
- **证明思路**

# 引理3.13

- **引理3.13: 如果一个语言是CFL, 则存在PDA识别这个语言.**
- **证明思路**
  - **CFG通过派生来产生串**

# 引理3.13

- **引理3.13: 如果一个语言是CFL, 则存在PDA识别这个语言.**
- **证明思路**
  - **CFG通过派生来产生串**
  - **PDA如何识别这个串?**

# 引理3.13

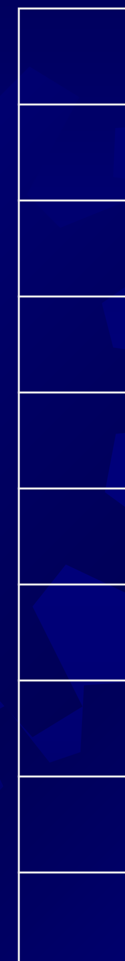
- **引理3.13: 如果一个语言是CFL, 则存在PDA识别这个语言.**
- **证明思路**
  - **CFG通过派生来产生串**
  - **PDA如何识别这个串?**
    - **PDA在堆栈中模拟派生过程**

# 例

- 文法$G_1$:

$$A \rightarrow 0A1$$
$$A \rightarrow B$$
$$B \rightarrow \#$$

- 派生:



# 例

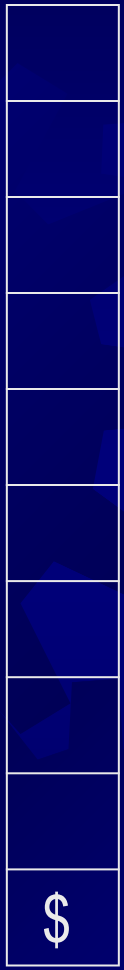
- 文法$G_1$:

  $A \rightarrow 0A1$

  $A \rightarrow B$

  $B \rightarrow \#$

- 派生:



# 例

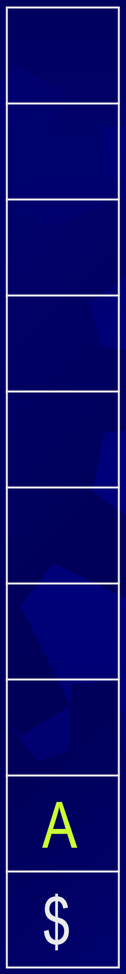
- 文法$G_1$:

  $A \rightarrow 0A1$

  $A \rightarrow B$

  $B \rightarrow \#$

- 派生:　A



# 例

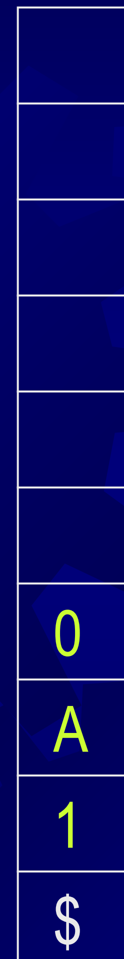
- 文法$G_1$:

  $A \to 0A1$

  $A \to B$

  $B \to \#$

- 派生:　$A \Rightarrow 0A1$



# 例

- 文法$G_1$:

  $A \to 0A1$

  $A \to B$

  $B \to \#$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11$

| |
|---|
| |
| |
| |
| |
| 0 |
| 0 |
| A |
| 1 |
| 1 |
| $ |

# 例

- 文法$G_1$:

  $A \rightarrow 0A1$

  $A \rightarrow B$

  $B \rightarrow \#$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111$

| |
|---|
| |
| |
| 0 |
| 0 |
| 0 |
| A |
| 1 |
| 1 |
| 1 |
| $ |

# 例

- 文法$G_1$:

$$A \rightarrow 0A1$$
$$A \rightarrow B$$
$$B \rightarrow \#$$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111$

| |
|---|
| |
| |
| 0 |
| 0 |
| 0 |
| B |
| 1 |
| 1 |
| 1 |
| $ |

# 例

- 文法$G_1$:

$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

| |
|---|
| |
| |
| 0 |
| 0 |
| 0 |
| # |
| 1 |
| 1 |
| 1 |
| $ |

# 引理3.13

- 引理3.13: 如果一个语言是CFL, 则存在PDA识别这个语言.
- 证明思路
  - CFG通过派生来产生串
  - PDA如何识别这个串?
    - PDA在栈中模拟派生过程
    - 让栈顶的终结符与输入进行匹配

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

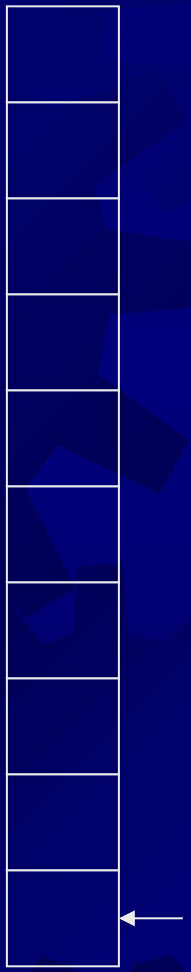
| |
|---|
| |
| |
| 0 |
| 0 |
| 0 |
| # |
| 1 |
| 1 |
| 1 |
| $ |

# 引理3.13

- 引理3.13: 如果一个语言是CFL, 则存在PDA识别这个语言.
- 证明思路
  - CFG通过派生来产生串
  - PDA如何识别这个串?
    - PDA在栈中模拟派生过程
    - 让栈顶终结符与输入进行匹配
    - 栈顶是变元,就模拟派生过程
    - 栈顶是终结符,就进行匹配

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |



- 文法$G_1$:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- 派生:

# 例

| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |
|---|---|---|---|---|---|---|---|---|

- 文法$G_1$:

  $A \to 0A1$

  $A \to B$

  $B \to \#$

- 派生:

| |
|---|
| |
| |
| |
| |
| |
| |
| |
| |
| |
| $ |

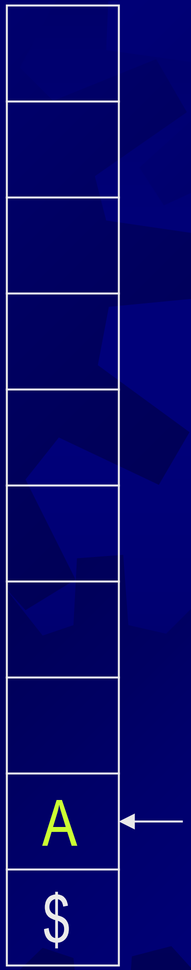
# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

- 派生: A



# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

  $A \to 0A1$

  $A \to B$

  $B \to \#$

- 派生: $A \Rightarrow 0A1$

| |
|---|
| |
| |
| |
| |
| |
| |
| 0 |
| A |
| 1 |
| $ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- 派生: $A \Rightarrow 0A1$

| |
|---|
| |
| |
| |
| |
| |
| |
| |
| A |
| 1 |
| $ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| \$ | 0 | 0 | 0 | # | 1 | 1 | 1 | \$ |

- 文法$G_1$:

$$A \rightarrow 0A1$$
$$A \rightarrow B$$
$$B \rightarrow \#$$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11$

| |
|---|
| |
| |
| |
| |
| |
| 0 |
| A |
| 1 |
| 1 |
| \$ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11$

| |
|---|
| |
| |
| |
| |
| |
| |
| A |
| 1 |
| 1 |
| $ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

  $A \rightarrow 0A1$

  $A \rightarrow B$

  $B \rightarrow \#$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111$

| |
|---|
| |
| |
| |
| |
| 0 |
| A |
| 1 |
| 1 |
| 1 |
| $ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

  $A \rightarrow 0A1$

  $A \rightarrow B$

  $B \rightarrow \#$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111$

| |
|---|
| |
| |
| |
| |
| |
| A |
| 1 |
| 1 |
| 1 |
| $ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| \$ | 0 | 0 | 0 | # | 1 | 1 | 1 | \$ |

- 文法$G_1$:

  $A \rightarrow 0A1$

  $A \rightarrow B$

  $B \rightarrow \#$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111$

| |
|---|
| |
| |
| |
| |
| |
| B |
| 1 |
| 1 |
| 1 |
| \$ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

| |
|---|
| |
| |
| |
| |
| |
| # |
| 1 |
| 1 |
| 1 |
| $ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

| |
|---|
| |
| |
| |
| |
| |
| |
| 1 |
| 1 |
| 1 |
| $ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

  $$A \rightarrow 0A1$$
  $$A \rightarrow B$$
  $$B \rightarrow \#$$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

| |
|---|
| |
| |
| |
| |
| |
| |
| |
| 1 |
| 1 |
| $ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

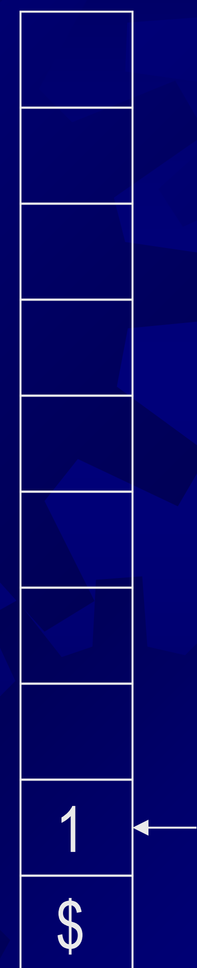
- 文法$G_1$:

  $A \to 0A1$

  $A \to B$

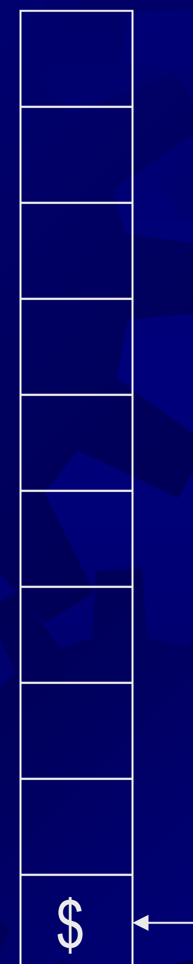
  $B \to \#$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$



# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |



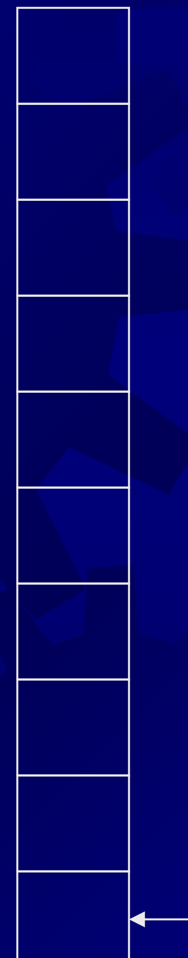

- 文法$G_1$:

$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |



- 文法$G_1$:

  $$A \rightarrow 0A1$$
  $$A \rightarrow B$$
  $$B \rightarrow \#$$

- 派生: $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

# 引理3.13

- **引理3.13: 如果一个语言是CFL, 则存在PDA识别这个语言.**
- **证明思路**
  - **CFG通过派生来产生串**
  - **PDA如何识别这个串?**
  - **PDA在栈中模拟派生过程, 与输入进行匹配**
    - **栈顶是变元,就模拟派生过程**
    - **栈顶是终结符,就进行匹配**
  - **如何把栈顶换成一串符号?**

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

- 派生: A

| |
|---|
| |
| |
| |
| |
| |
| |
| |
| |
| A |
| $ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- 派生: $A \Rightarrow 0A1$

| |
|---|
| |
| |
| |
| |
| |
| |
| 0 |
| A |
| 1 |
| $ |

# 引理3.13

- **引理3.13: 如果一个语言是CFL, 则存在PDA识别这个语言.**
- **证明思路**
  - **CFG通过派生来产生串**
  - **PDA如何识别这个串?**
  - **PDA在栈中模拟派生过程, 与输入进行匹配**
    - 栈顶是变元,就模拟派生过程
    - 栈顶是终结符,就进行匹配
  - 如何把栈顶换成一串符号?

    中间状态

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

  $$A \rightarrow 0A1$$
  $$A \rightarrow B$$
  $$B \rightarrow \#$$

- 派生: A

| |
|---|
| |
| |
| |
| |
| |
| |
| |
| |
| A |
| $ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- 派生: $A \Rightarrow 0A1$

| |
|---|
| |
| |
| |
| |
| |
| |
| |
| |
| 1 |
| $ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

- 文法$G_1$:

$$A \rightarrow 0A1$$
$$A \rightarrow B$$
$$B \rightarrow \#$$

- 派生: $A \Rightarrow 0A1$

| |
|---|
| |
| |
| |
| |
| |
| |
| |
| A |
| 1 |
| $ |

# 例

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $ | 0 | 0 | 0 | # | 1 | 1 | 1 | $ |

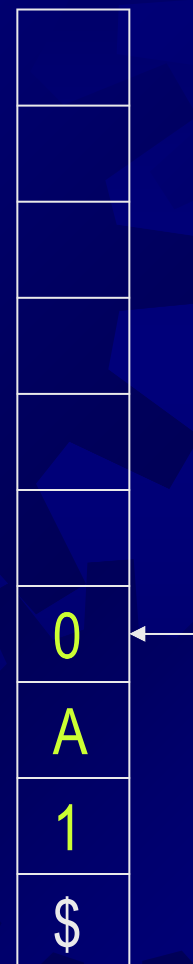
- 文法$G_1$:

  $A \to 0A1$

  $A \to B$

  $B \to \#$

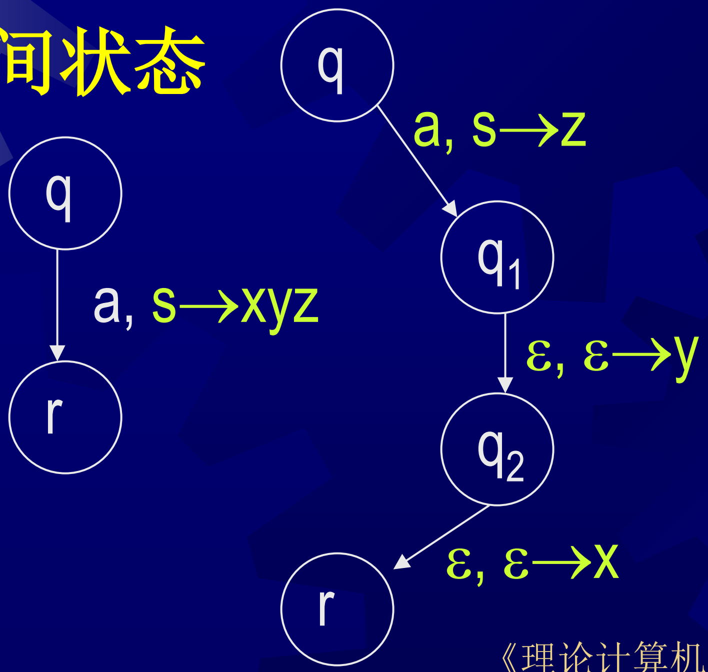
- 派生: $A \Rightarrow 0A1$



# 引理3.13

- **引理3.13:如果一个语言是CFL,则存在PDA识别这个语言.**
- **证明思路**
  - **如何把栈顶换成一串符号?**

**中间状态**



# 引理3.13

- 引理3.13: 如果一个语言是CFL,
则存在PDA识别这个语言.
- 证明: 设A是CFL,根据定义,
存在产生A的CFG $G=(V,\Sigma,R,S)$.
下面构造识别A的PDA

$$P = ( \{q_{start},q_{loop},q_{accept}\}\cup E, \Sigma, V\cup\Sigma, \delta, q_{start}, \{q_{accept}\} ).$$

# P的非形式化描述

- 把标记$和初始变元放入栈中
- 重复下列步骤
  - 如果栈顶是变元A,
    则非确定地选择一条规则A→w.
    把A替换成右边的字符串w.
  - 如果栈顶是终结符a,
    则读输入中下一个符号与a比较.
    如果匹配,则重复.
    如果不匹配,则这个非确定性分支拒绝.
  - 如果栈顶是$,则进入接受状态.
    如果此刻输入全部读完,
    则接受这个输入串.

# 引理3.13

- **引理3.13**: 如果一个语言是CFL, 则存在PDA识别这个语言.
- **证明(续)**: 设P在状态q下读a,

把栈顶s换成$u=u_1u_2\ldots u_k$, 进入状态r.

则引入新状态$q_1,q_2,\ldots,q_{k-1}$,

令 $\delta(q,a,s)$包含$(q_1,u_k)$,

$\delta(q_1,\varepsilon,\varepsilon)=\{(q_2,u_{k-1})\}$,

$\delta(q_2,\varepsilon,\varepsilon)=\{(q_3,u_{k-2})\}$, …,

$\delta(q_{k-1},\varepsilon,\varepsilon)=\{(r,u_1)\}$.

把这些动作记为 $(r,u)\in\delta(q,a,s)$.

所有这类新状态的集合记为E.

# 引理3.13

- 引理3.13: 如果一个语言是CFL, 则存在PDA识别这个语言.
- 证明(续): 转移函数δ如下规定:

$\delta(q_{start},\varepsilon,\varepsilon)=\{(q_{loop},S\$)\}$,

$\delta(q_{loop},\varepsilon,A)=\{\ (q_{loop},w)\ |$

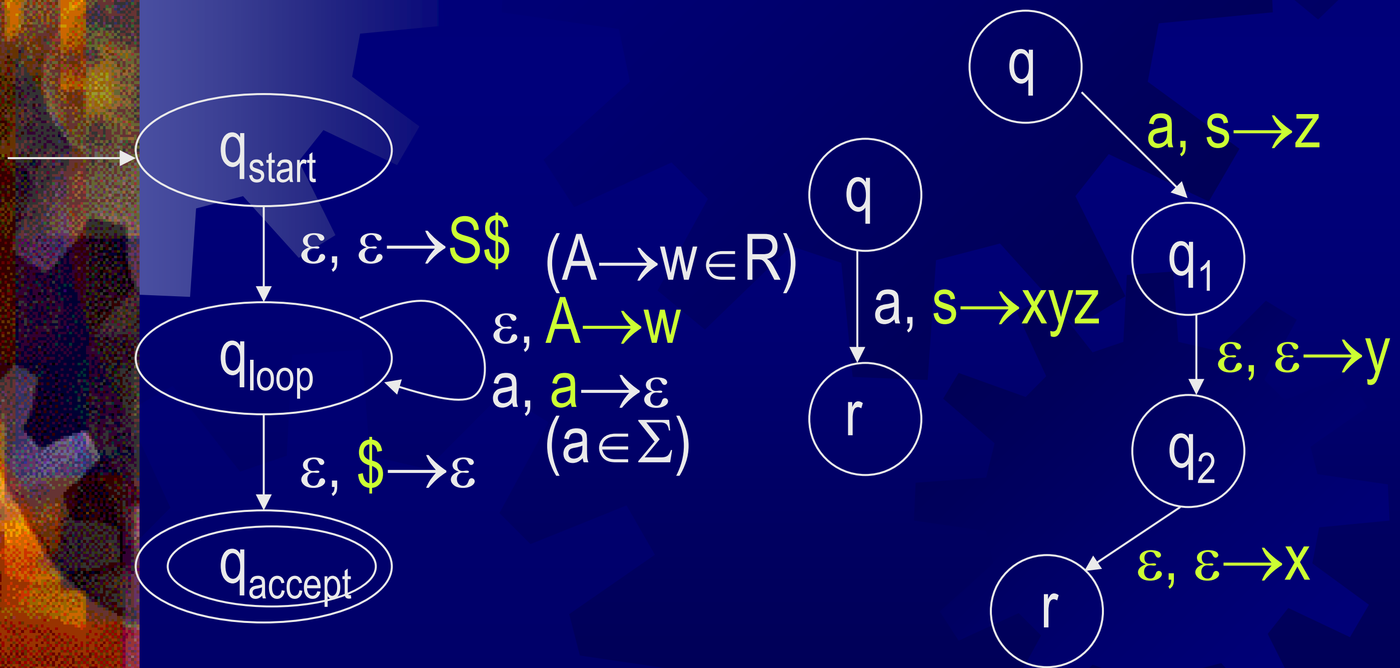
$A\rightarrow w$是R中规则 },

$\delta(q_{loop},a,a)=\{(q_{loop},\varepsilon)\}$,

$\delta(q_{loop},\varepsilon,\$)=\{(q_{accept},\varepsilon)\}$.

# 引理3.13

- 引理3.13:如果一个语言是CFL,则存在PDA识别这个语言.
- 证明(续): 状态图如下.　　#



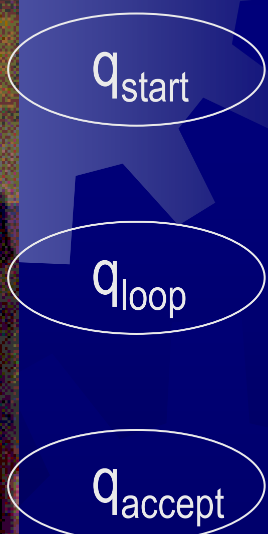
# 从CFG构造PDA的例子

# 例3.14

- 把CFG G: $S \to aTb \mid b,\ T \to Ta \mid \varepsilon$ 转化成等价的PDA $P_1$.

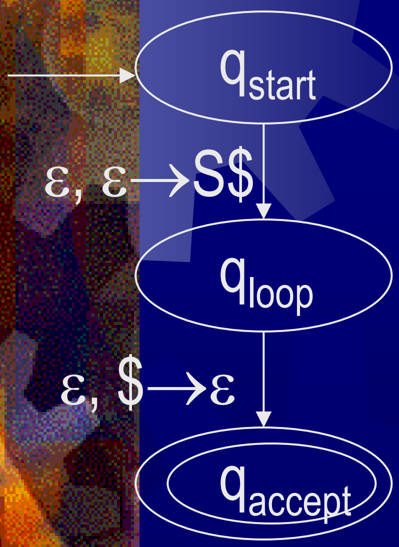
# 例3.14

- 把CFG G: $S \rightarrow aTb \mid b,\ T \rightarrow Ta \mid \varepsilon$ 转化成等价的PDA $P_1$.



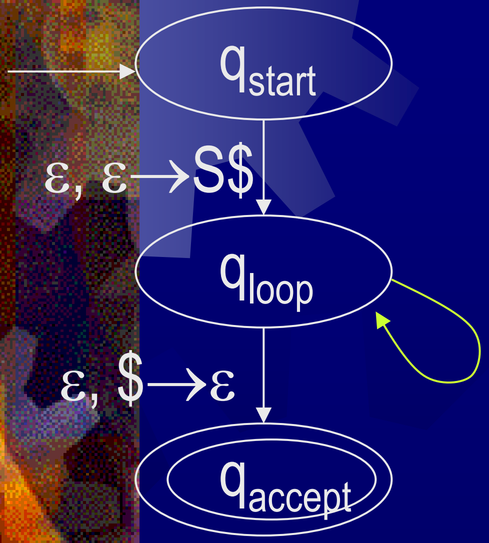
# 例3.14

- 把CFG G: $S \to aTb \mid b,\ T \to Ta \mid \varepsilon$ 转化成等价的PDA $P_1$.



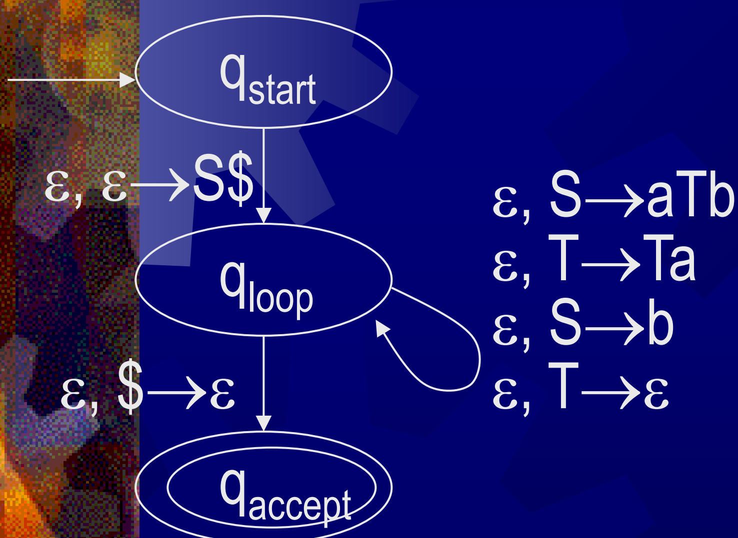
# 例3.14

- 把CFG G: $S \to aTb \mid b,\ T \to Ta \mid \varepsilon$ 转化成等价的PDA $P_1$.



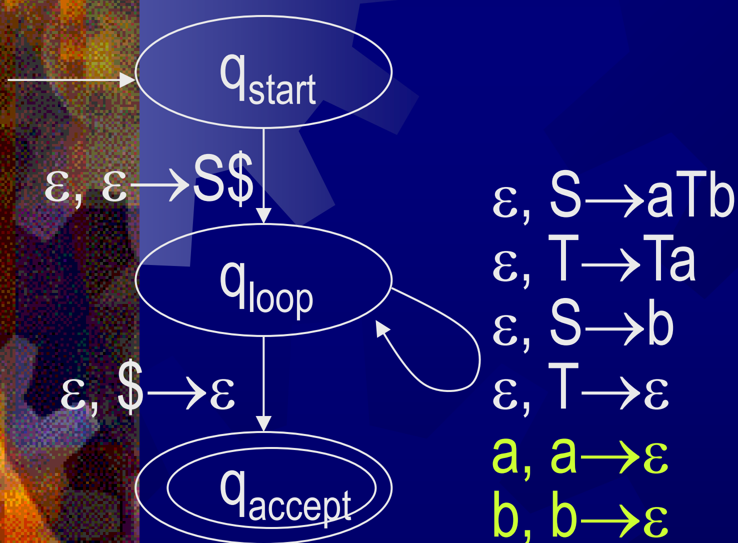
# 例3.14

- 把CFG G: $S \to aTb \mid b,\ T \to Ta \mid \varepsilon$ 转化成等价的PDA $P_1$.



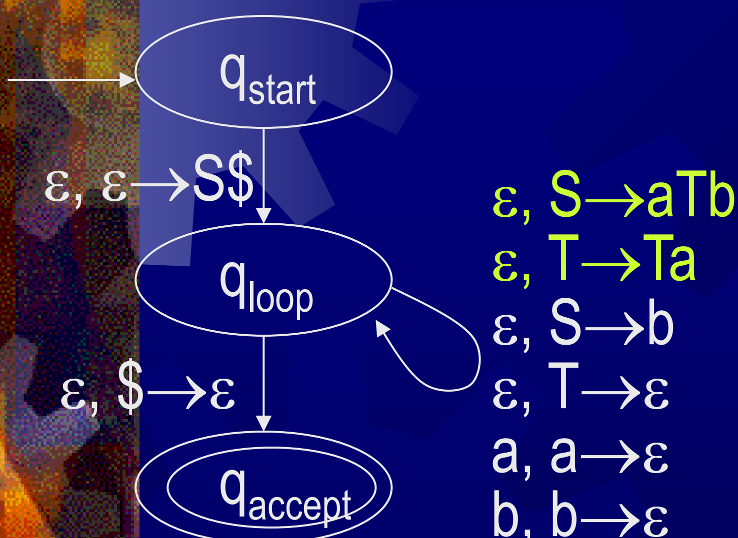
# 例3.14

- 把CFG G: $S \to aTb \mid b,\ T \to Ta \mid \varepsilon$ 转化成等价的PDA $P_1$.



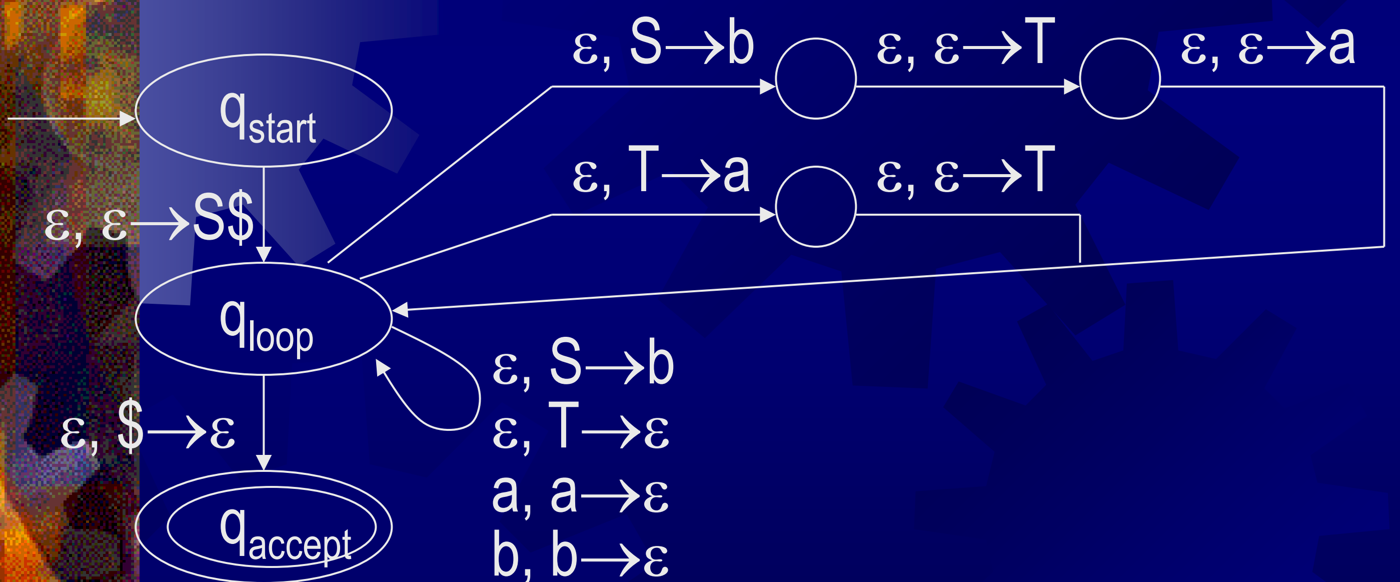
# 例3.14

- 把CFG G: $S \to aTb \mid b,\ T \to Ta \mid \varepsilon$ 转化成等价的PDA $P_1$.



# 例3.14

- 把CFG G: $S \to aTb \mid b,\ T \to Ta \mid \varepsilon$ 转化成等价的PDA $P_1$.



# 从PDA构造CFG的算法

# 引理3.15

- **引理3.15: 如果一个语言被PDA识别,则这个语言是CFL.**
- **证明思路**

# 引理3.15

- 引理3.15: 如果一个语言被PDA识别,则这个语言是CFL.
- 证明思路
  - 用CFG派生出PDA接受的串

# 引理3.15

- **引理3.15: 如果一个语言被PDA识别,则这个语言是CFL.**
- **证明思路**
  - 用CFG派生出PDA接受的串
  - 变元基本起状态的作用(如DFA到CFG)

# 引理3.15

- **引理3.15: 如果一个语言被PDA识别,则这个语言是CFL.**
- **证明思路**
  - 用CFG派生出PDA接受的串
  - 变元基本起状态的作用
  - 如何消除栈的影响?

# 引理3.15

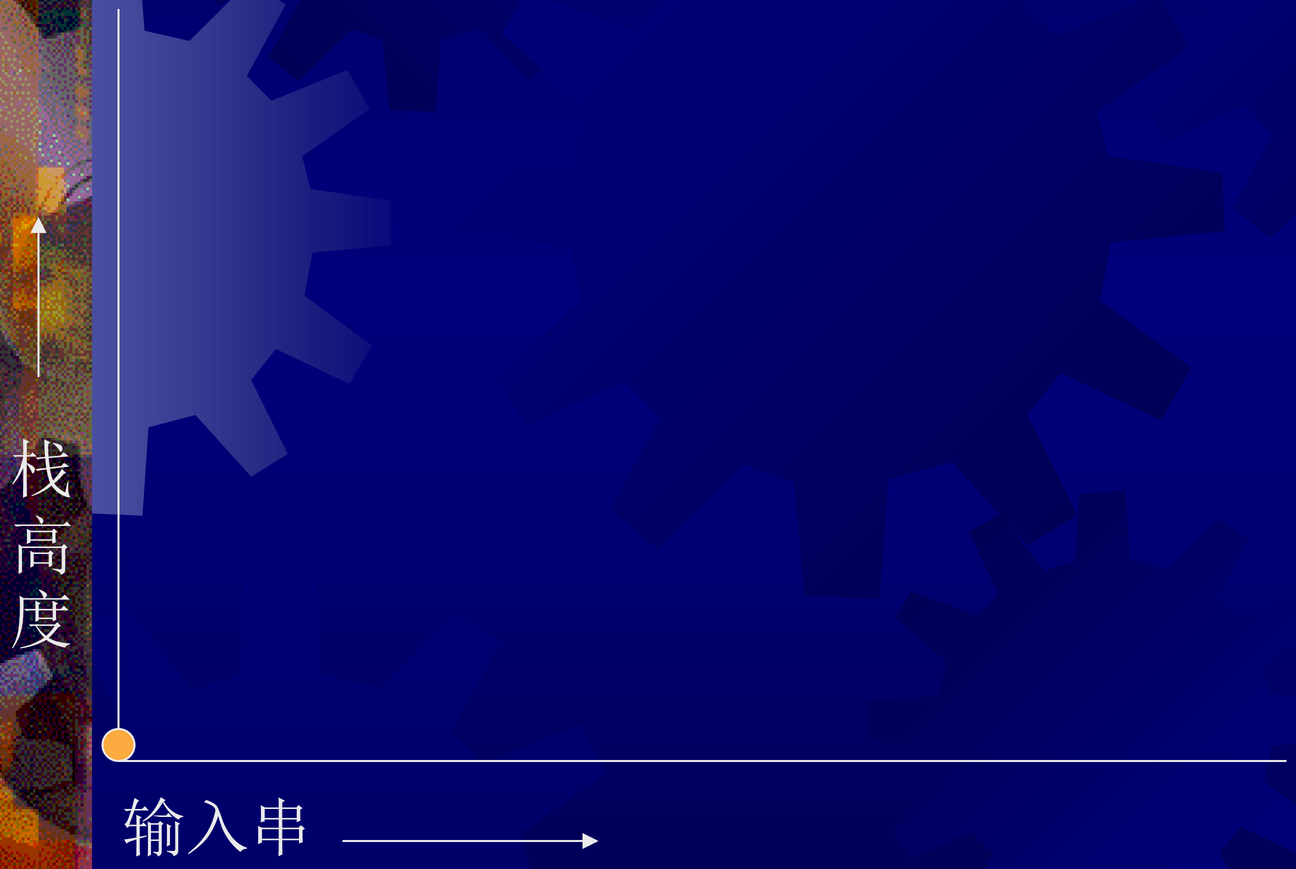
- **引理3.15**: 如果一个语言被PDA识别,则这个语言是CFL.
- **证明思路**
  - 用CFG派生出PDA接受的串
  - 变元基本起状态的作用
  - 如何消除栈的影响?
    分析PDA工作过程

# PDA的工作过程

栈高度

输入串

# PDA的工作过程

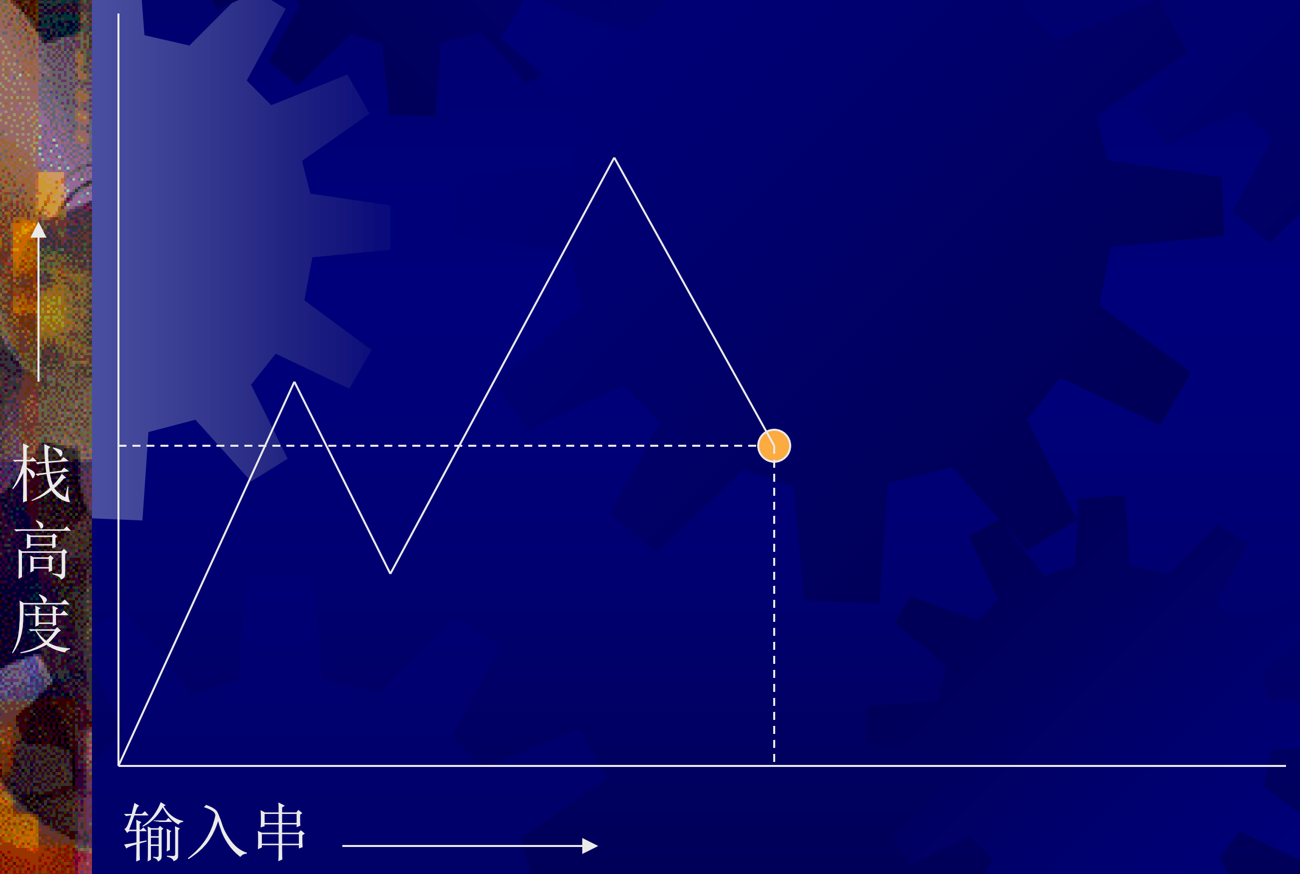


# PDA的工作过程

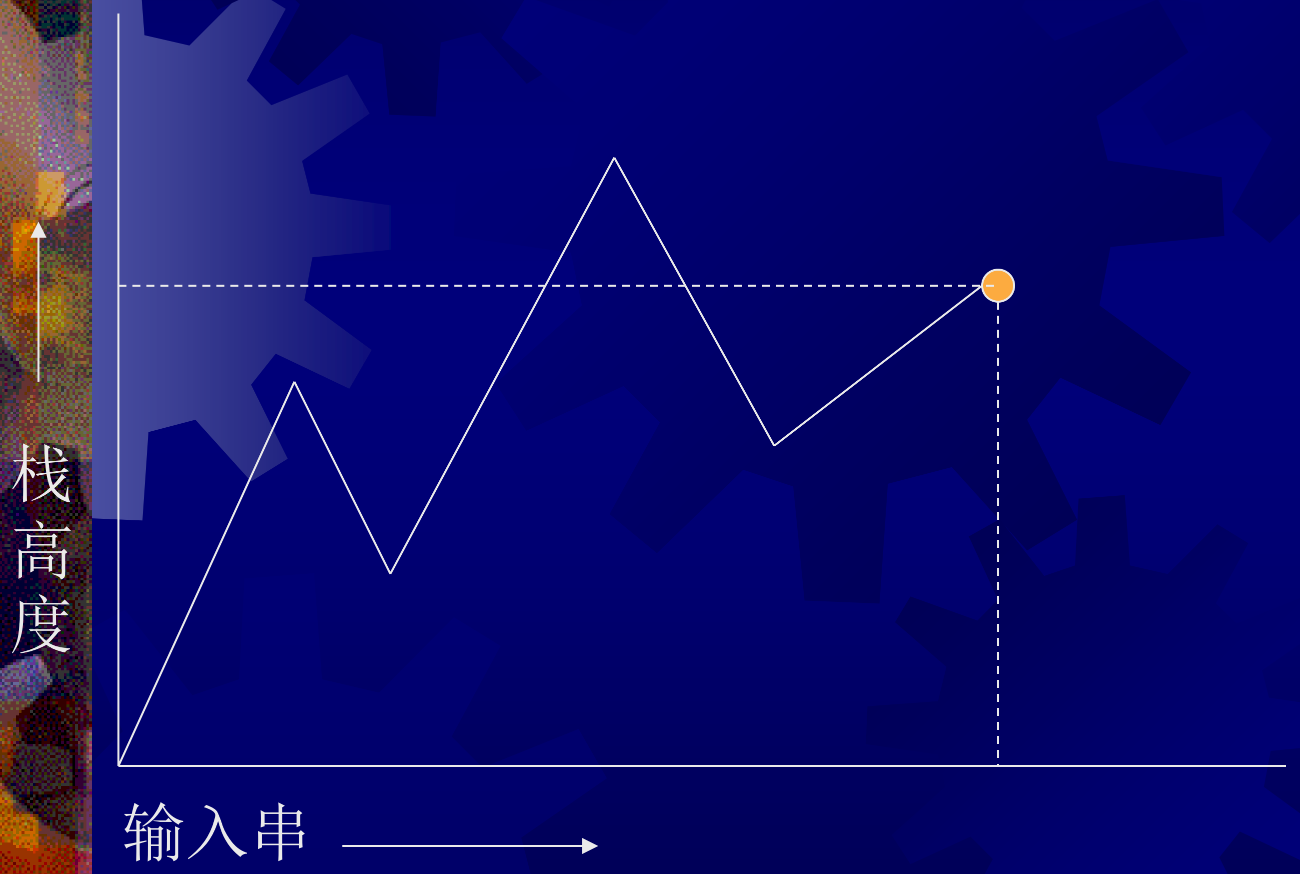
# PDA的工作过程

# PDA的工作过程

# PDA的工作过程

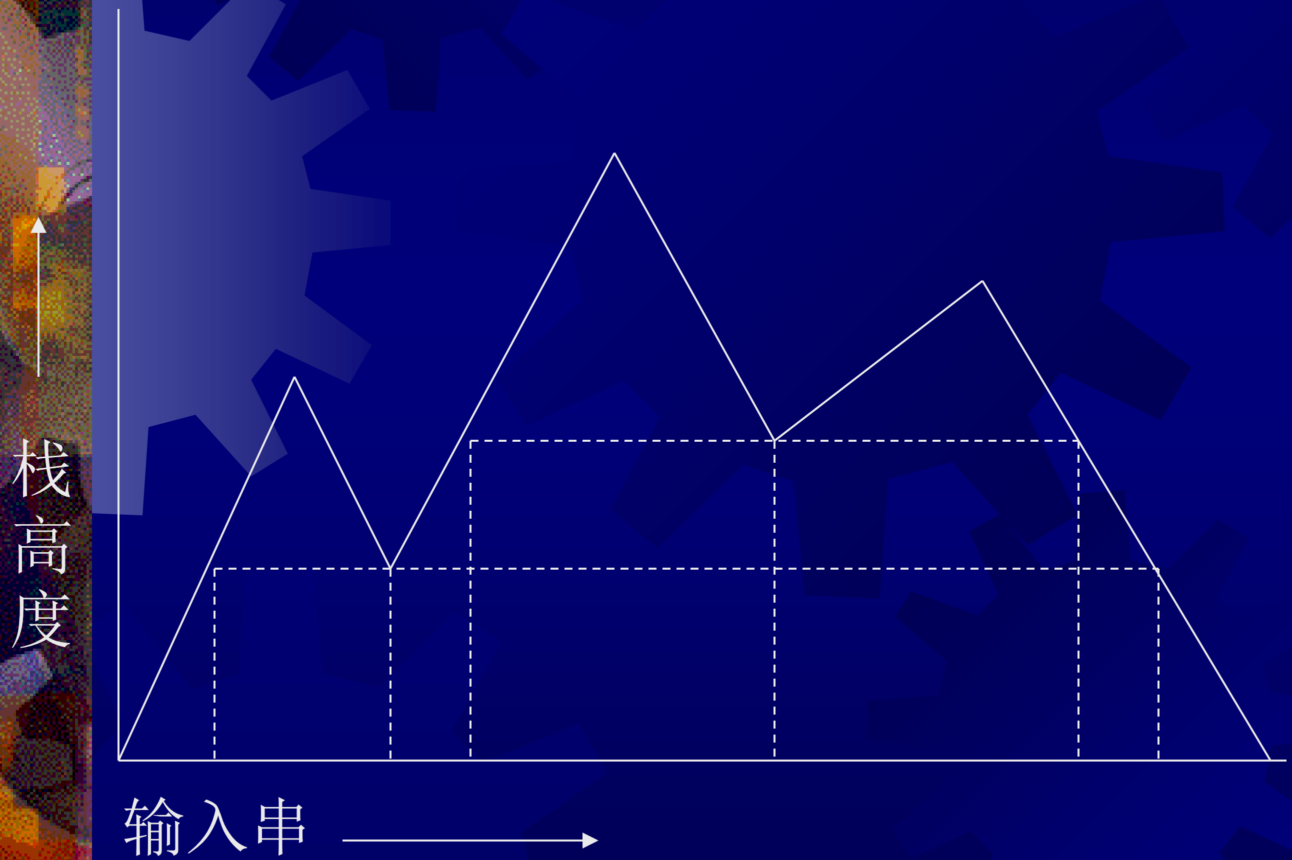


# PDA的工作过程

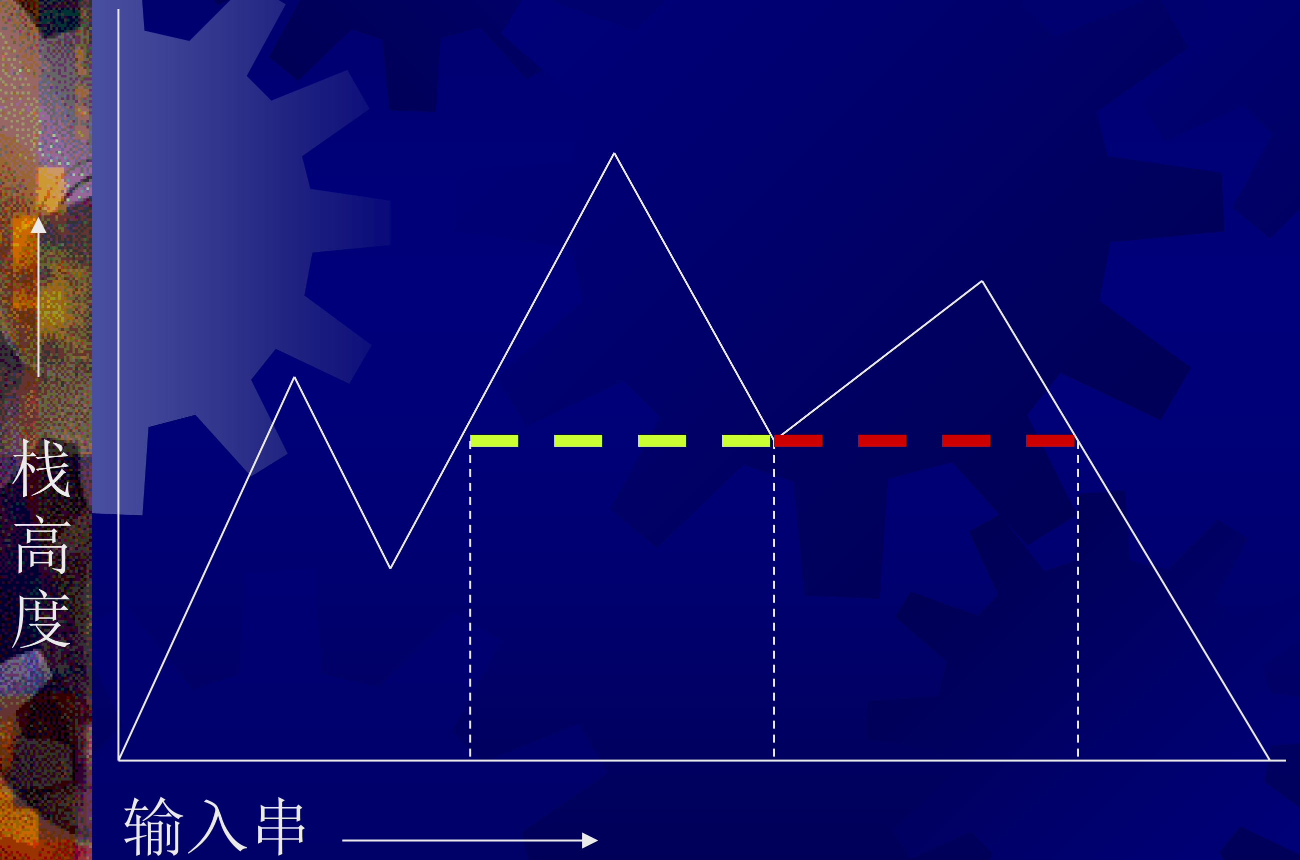


# PDA的工作过程

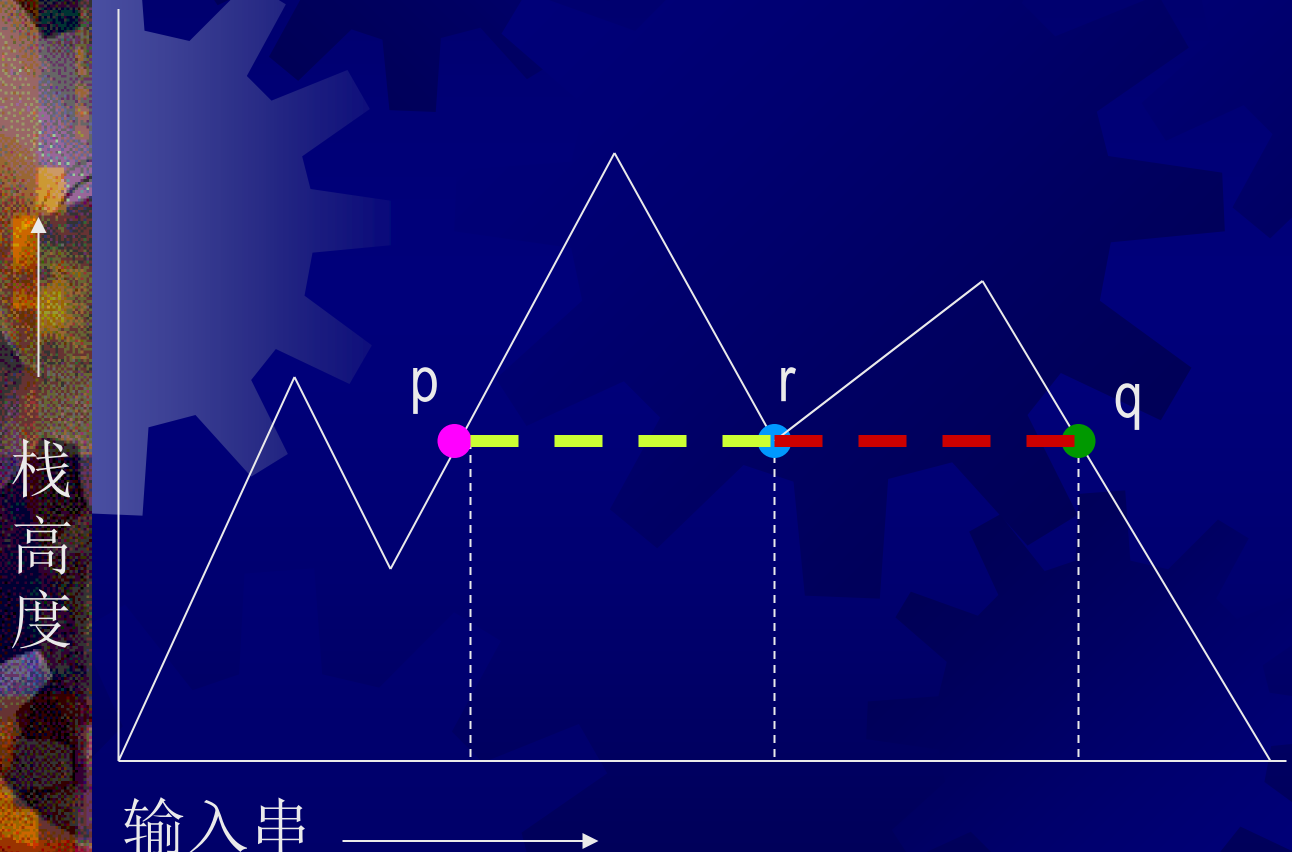
# PDA的工作过程



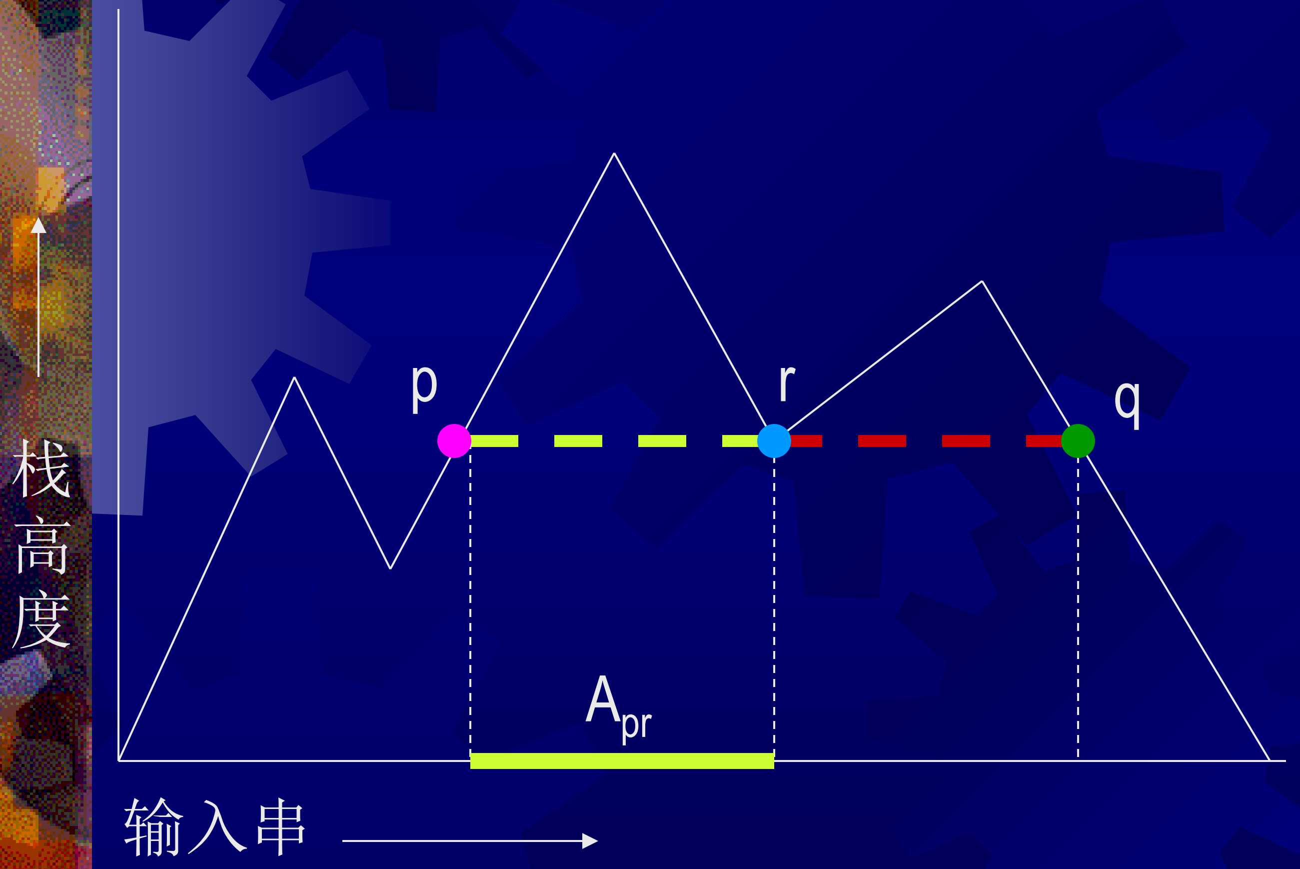
# PDA的工作过程



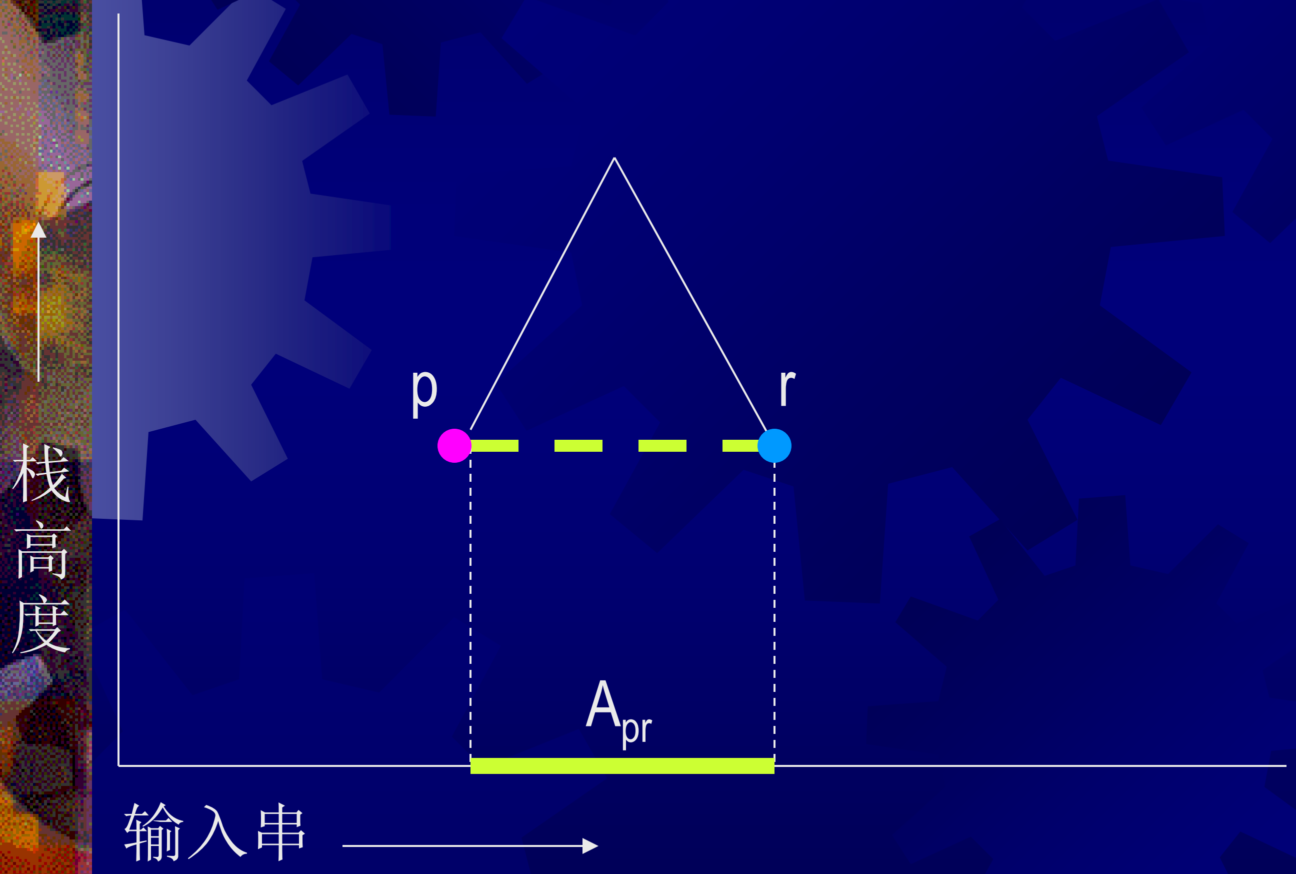
# PDA的工作过程



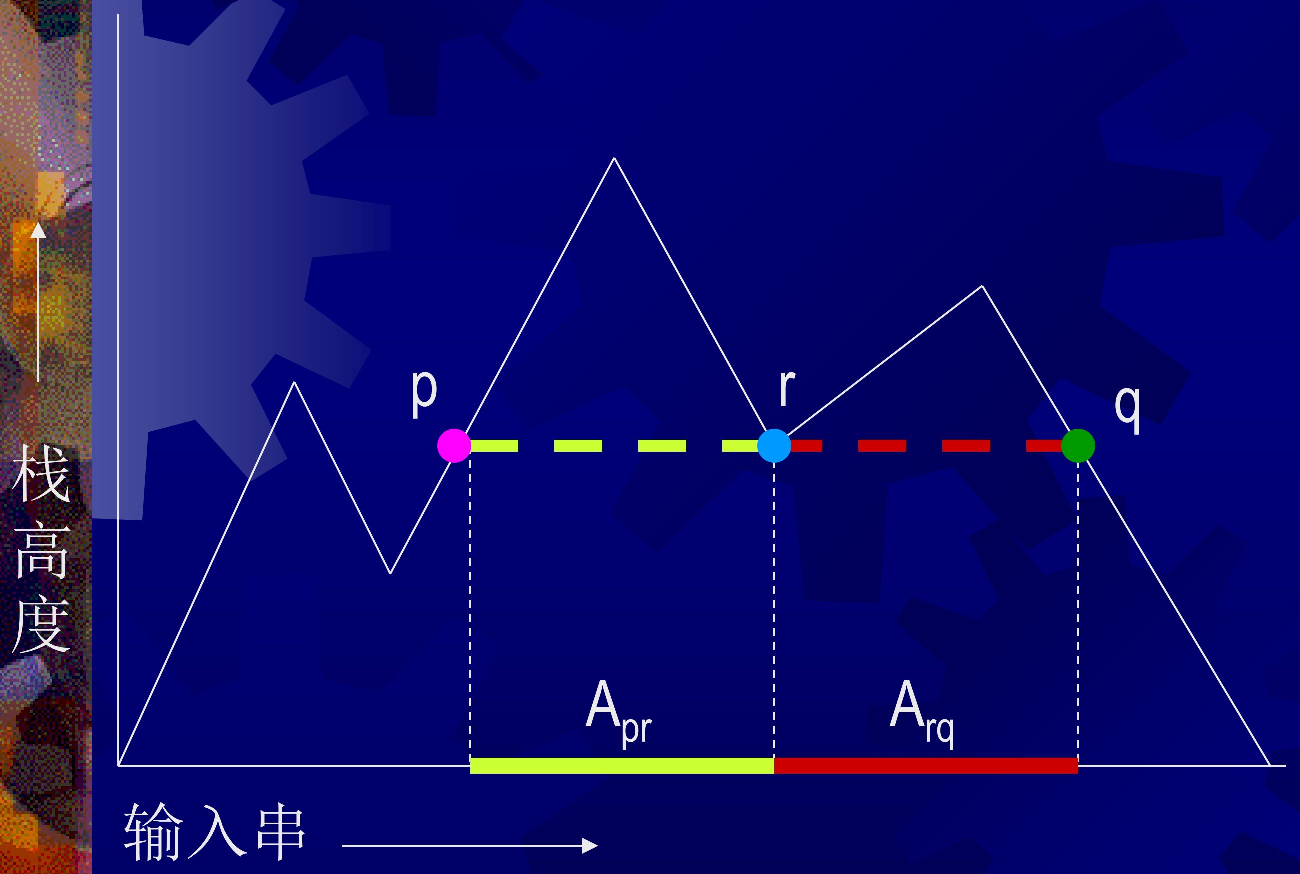
# PDA的工作过程



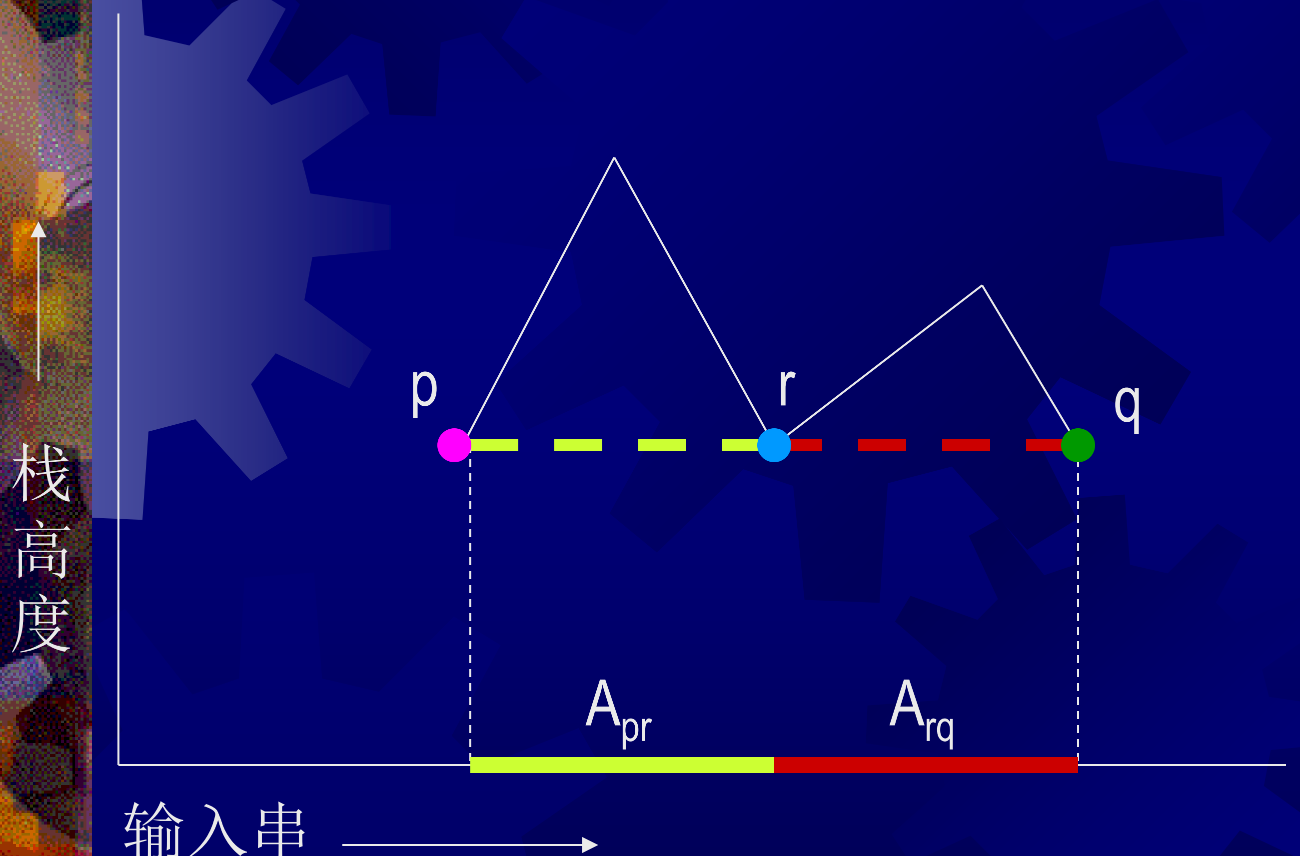
# PDA的工作过程



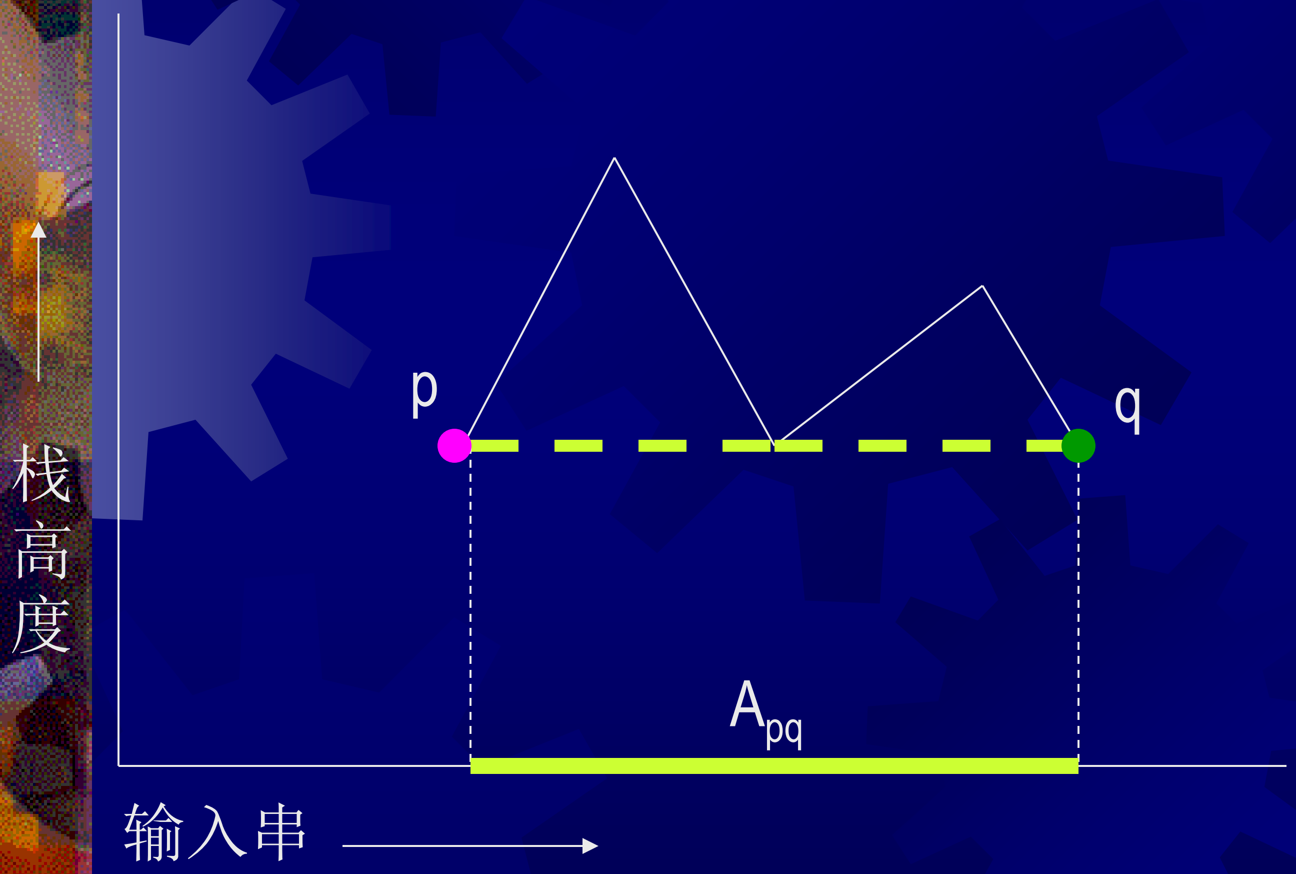
# PDA的工作过程



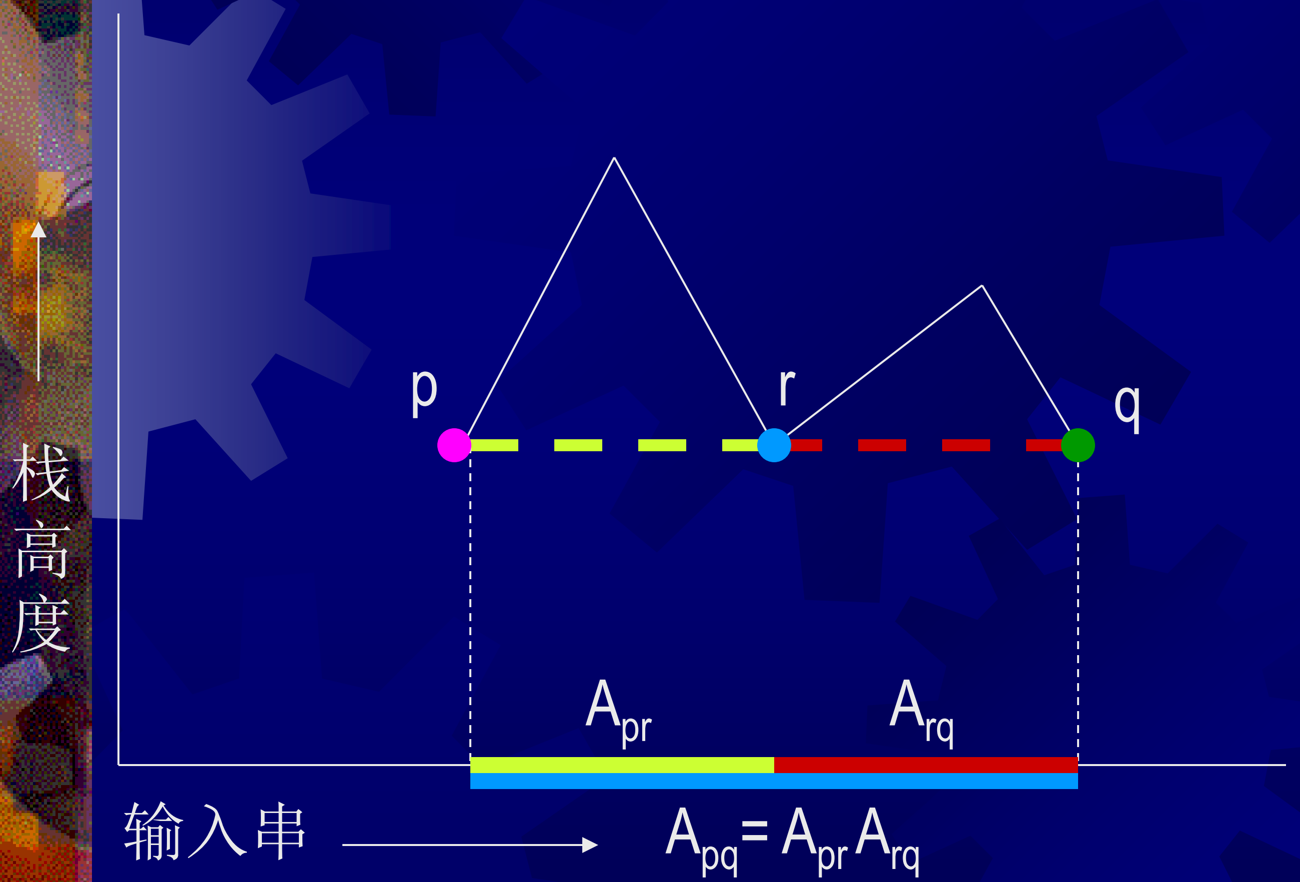
# PDA的工作过程



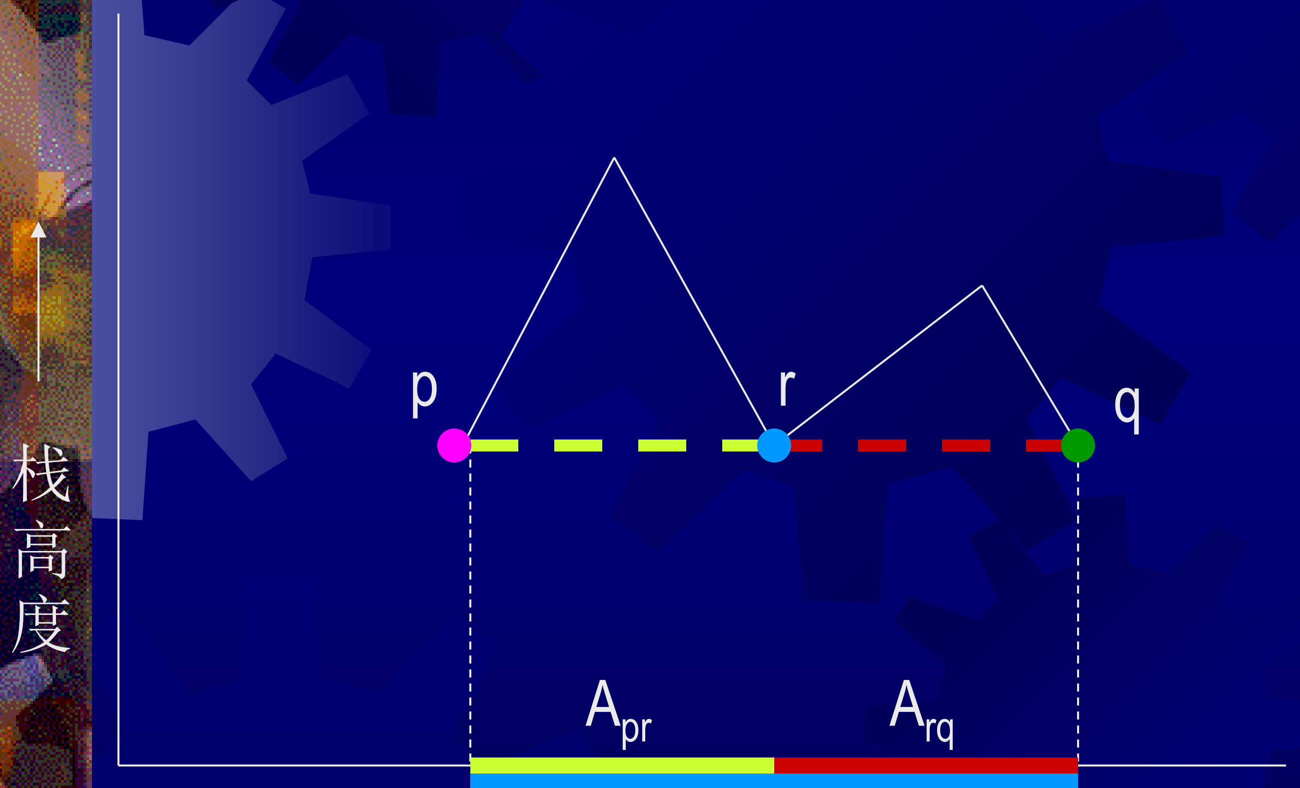
# PDA的工作过程



# PDA的工作过程



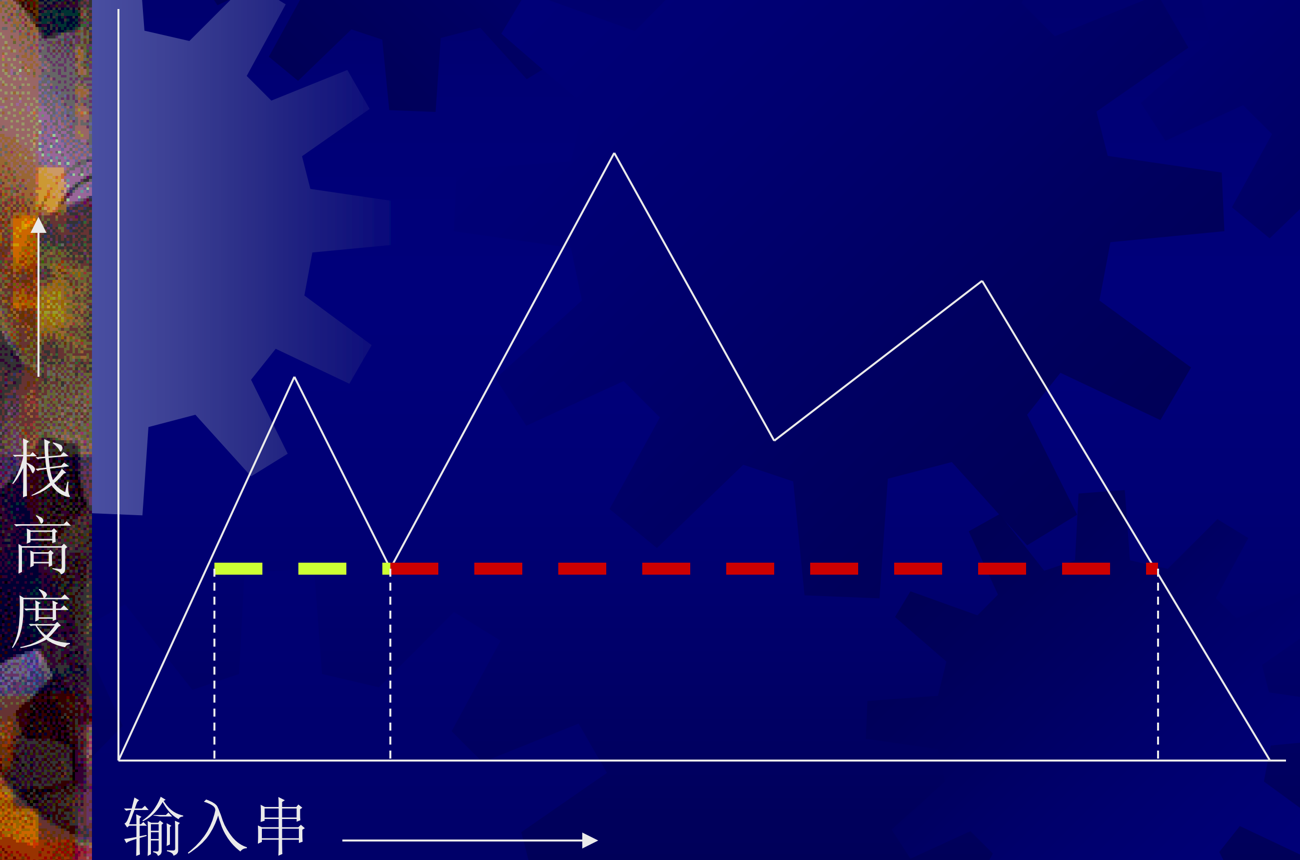
# PDA的工作过程



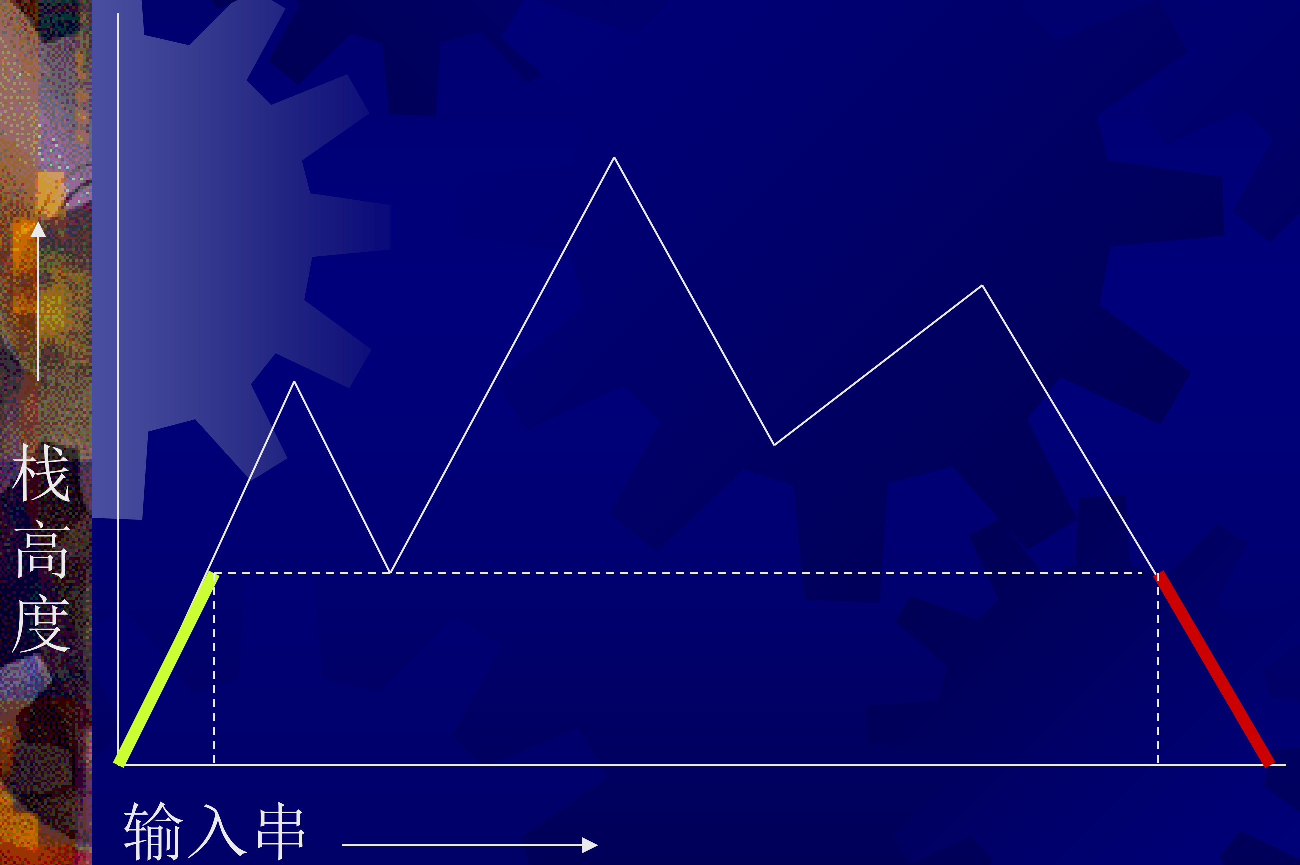

输入串 →

$A_{pq} = A_{pr} A_{rq}$　　$A_{pq} \to A_{pr} A_{rq}$

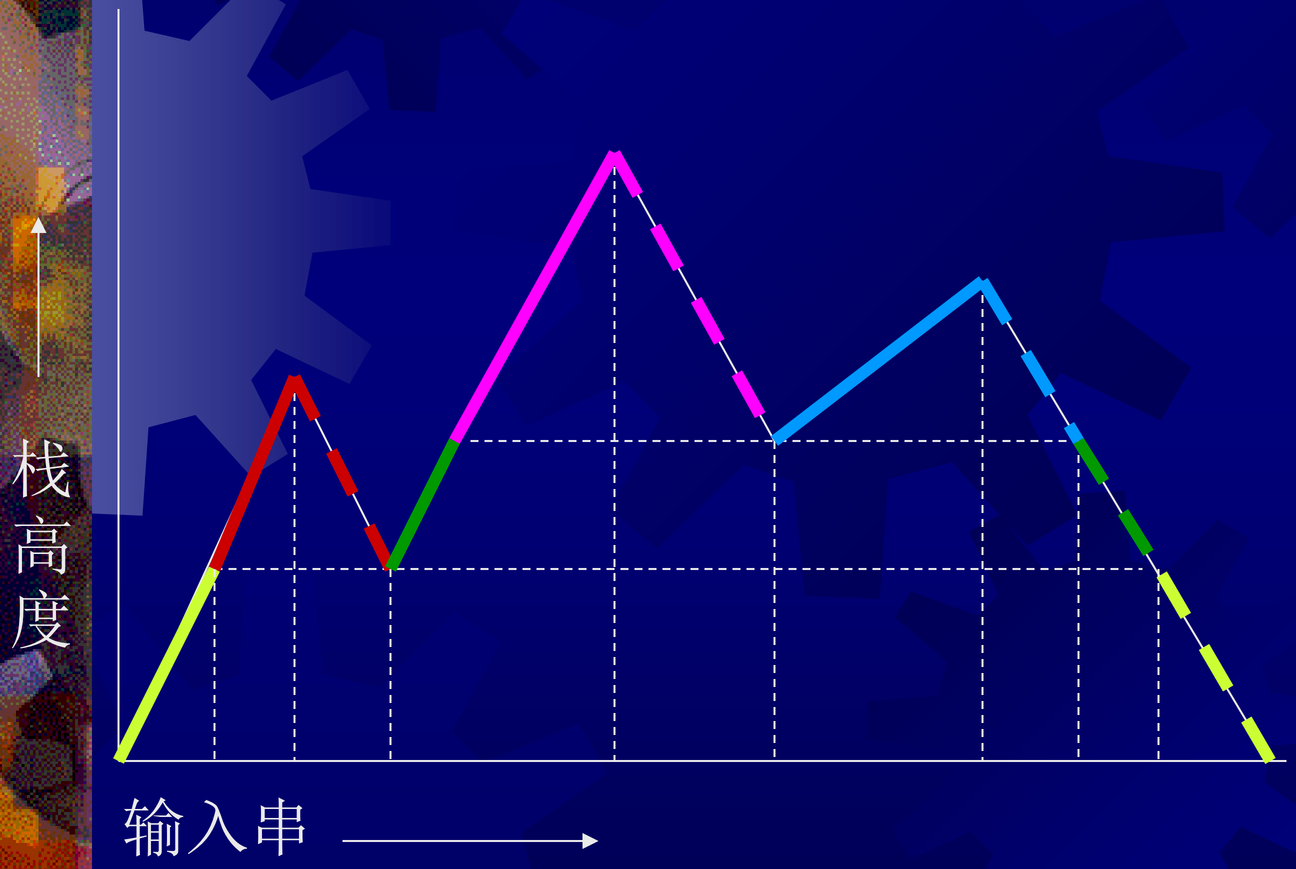
# PDA的工作过程



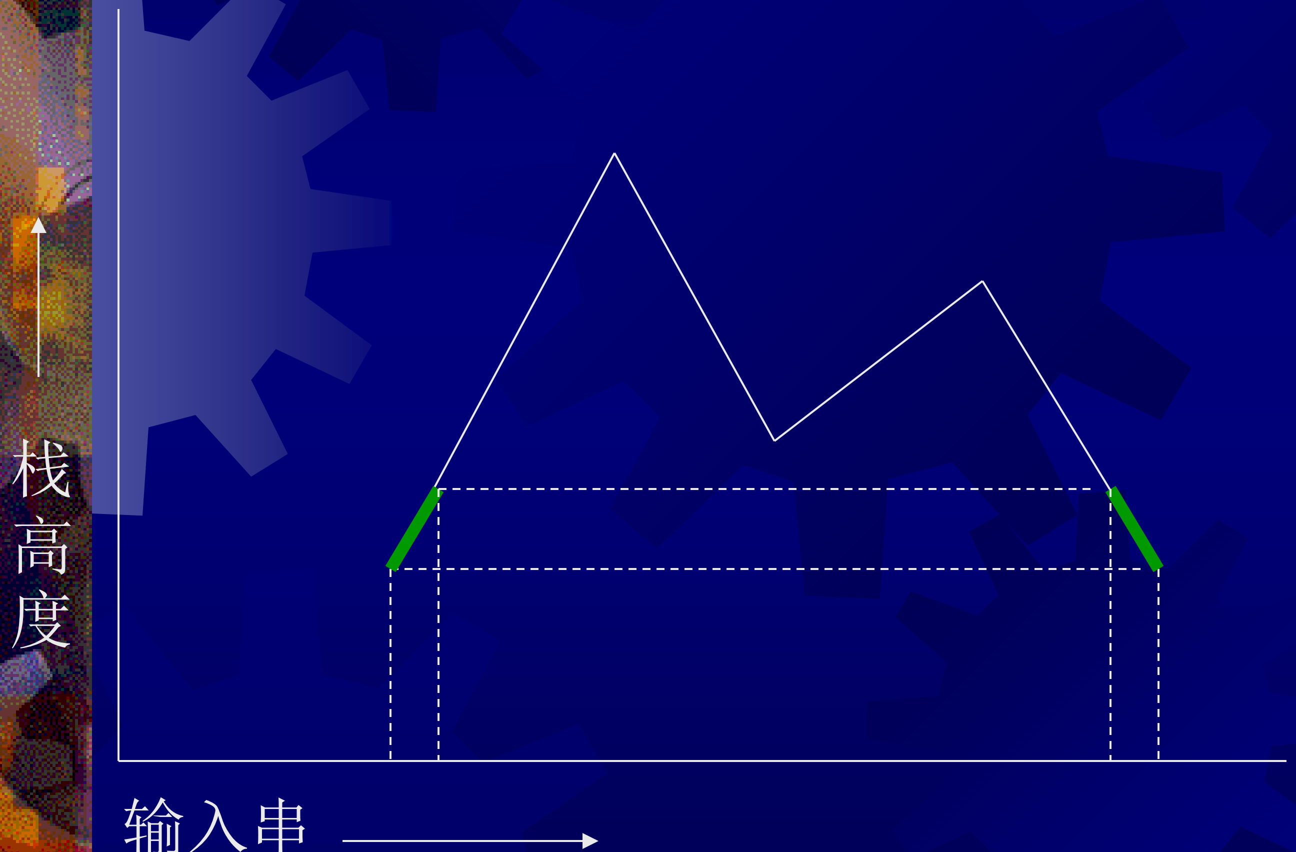
# PDA的工作过程



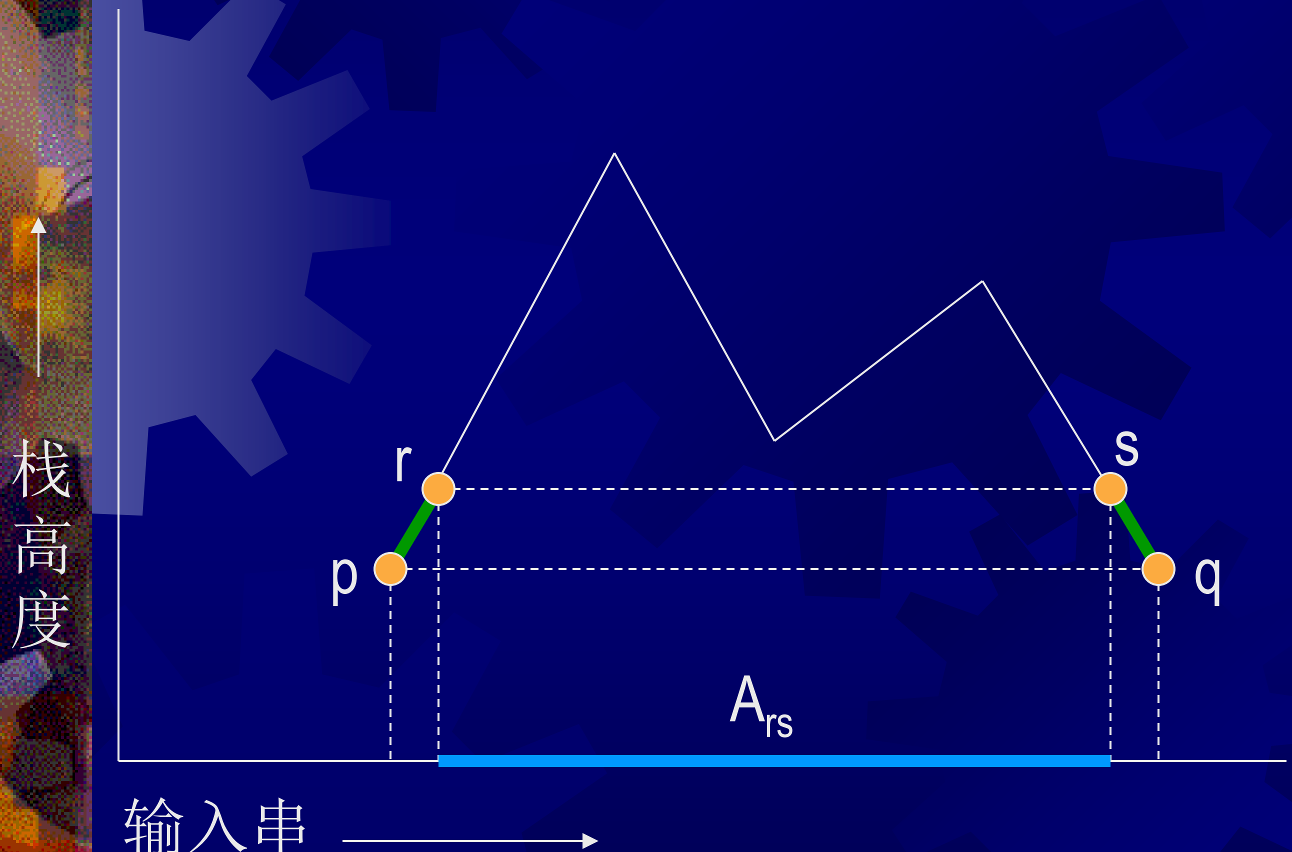
# PDA的工作过程



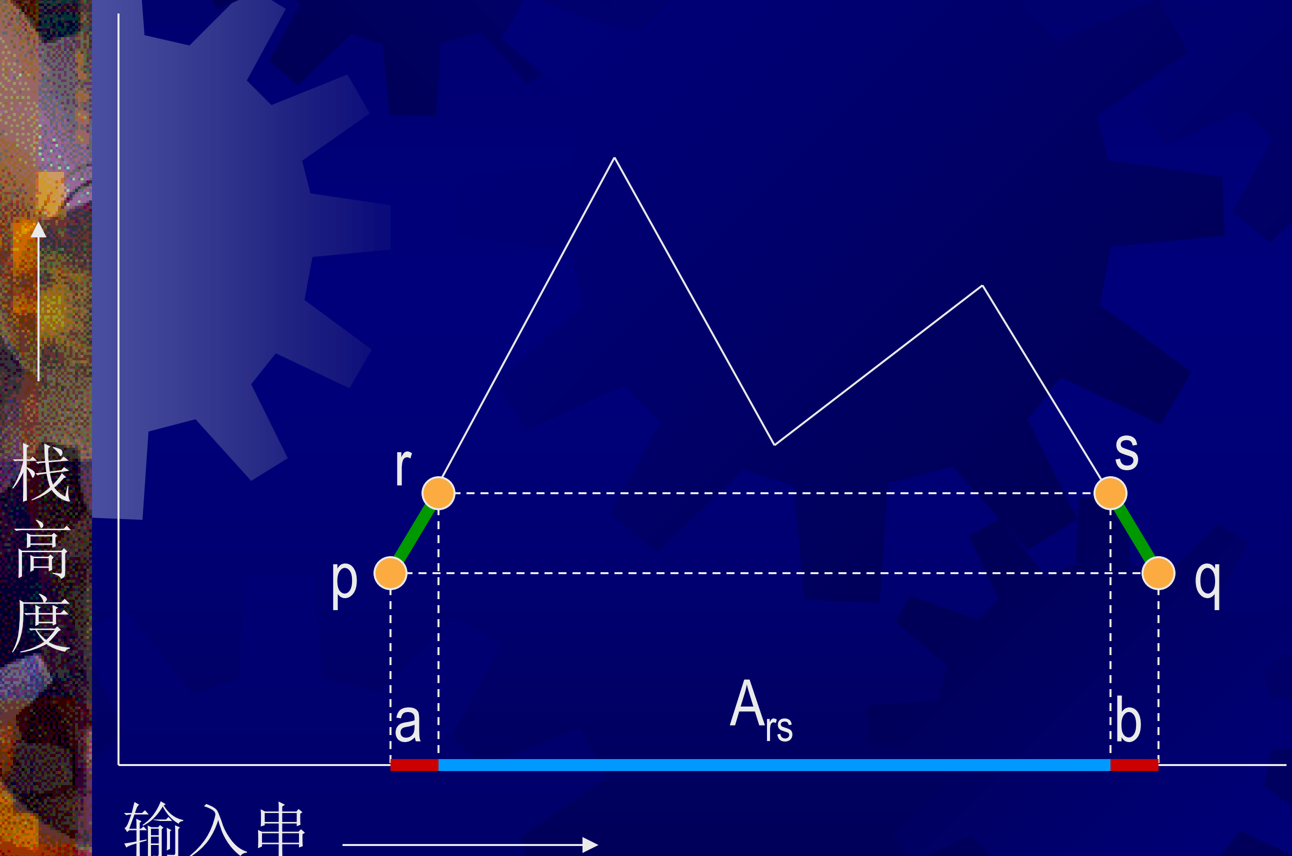
# PDA的工作过程



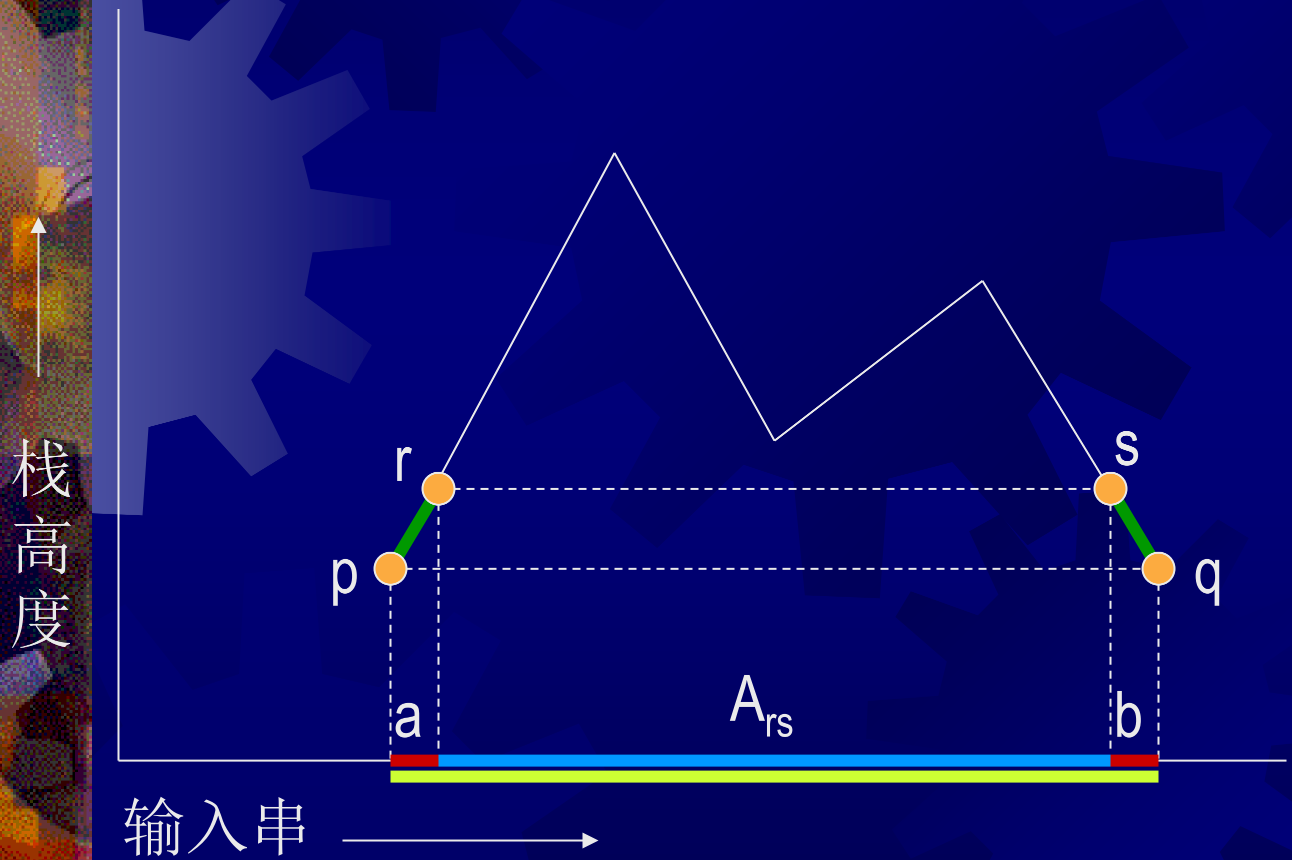
# PDA的工作过程



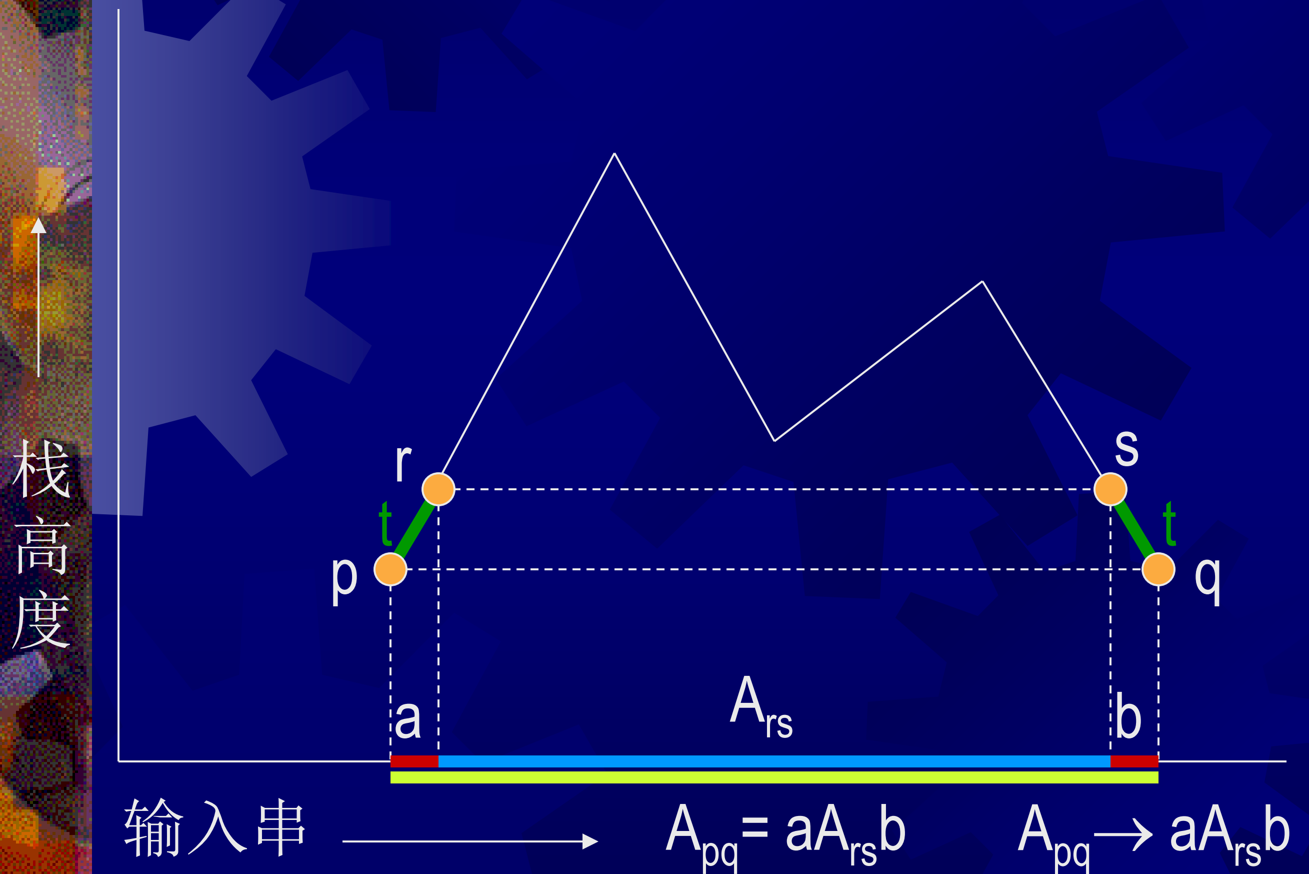
# PDA的工作过程



# PDA的工作过程



# PDA的工作过程



# 特殊形式的PDA

- 只有唯一接受状态$q_{accept}$
- 在接受之前排空栈
- 每一个转移把一个符号推入栈,

  或把一个符号弹出栈,

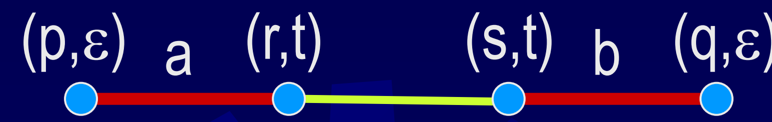
  但不同时做这两个动作
  - 把同时推入和弹出的转移分成两个转移, 经过一个中间状态
  - 把既不推入也不弹出的转移分成两个转移,先推入一个任意符号,再推出这个符号

# 引理3.15


- **引理3.15**: 如果一个语言被PDA识别, 则这个语言是CFL.
- **证明**:设PDA $P=(Q,\Sigma,\Gamma,\delta,q_0,\{q_{accept}\})$, 构造等价CFG

  $G=(\{A_{pq}|p,q\in Q\},\Sigma,R,A_{q0qaccept})$,

  R包含如下规则:
  - $A_{pq}\rightarrow aA_{rs}b$,（任何 $p,q,r,s\in Q$, $a,b\in\Sigma_\varepsilon$, $t\in\Gamma$, $\delta(p,a,\varepsilon)$包含$(r,t)$且 $\delta(s,b,t)$包含$(q,\varepsilon)$ ）
  - $A_{pq}\rightarrow A_{pr}A_{rq}$ ,（任何 $p,q,r\in Q$ ）
  - $A_{pp}\rightarrow\varepsilon$,（任何 $p\in Q$ ）




- $A_{pq} \to aA_{rs}b$
- $A_{pq} \to A_{pr}A_{rq}$
- $A_{pp} \to \varepsilon$

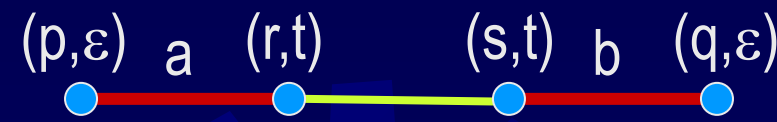
# 引理3.15

- 证明(续): 下面证明$A_{pq}$产生x,当且仅当x把P从状态p和空栈一起带到状态q和空栈.
- 断言3.16: 如果$A_{pq}$产生x,则x把P从状态p和空栈一起带到状态q和空栈.
- 断言3.17: 如果x把P从状态p和空栈一起带到状态q和空栈,则$A_{pq}$产生x.

下面证明这两个断言. #

# 断言3.16

- $A_{pq} \to aA_{rs}b$
- $A_{pq} \to A_{pr}A_{rq}$
- $A_{pp} \to \varepsilon$



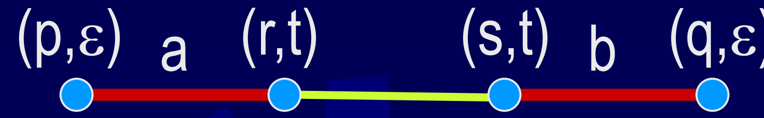

- **断言3.16**: 如果$A_{pq}$产生x, 则x把P从状态p和空栈一起带到状态q和空栈.
- **证明**: (对$A_{pq}$派生x的步数进行归纳).

  (1) $A_{pq} \Rightarrow x$. 只能使用$A_{pp} \to \varepsilon$.
  显然ε把P从状态p和空栈一起带到状态p和空栈.

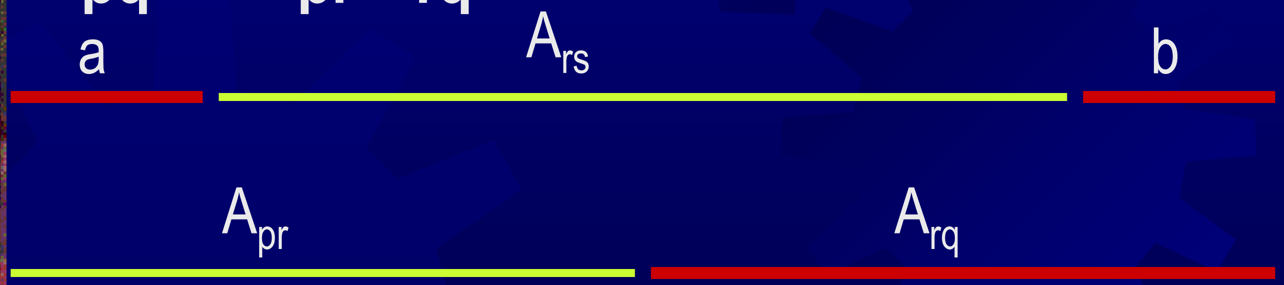
  (2) 假设断言对不超过k步的$A_{pq} \Rightarrow^* x$成立.
  证明断言对k+1步的$A_{pq} \Rightarrow^* x$成立.

# 断言3.16

- $A_{pq} \to aA_{rs}b$
- $A_{pq} \to A_{pr}A_{rq}$
- $A_{pp} \to \varepsilon$

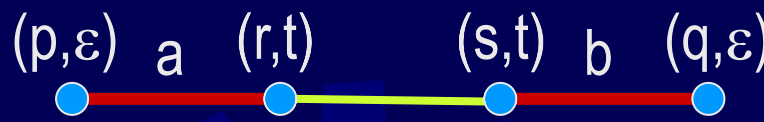



- **断言3.16**: 如果$A_{pq}$产生x, 则x把P从状态p和空栈一起带到状态q和空栈.
- **证明**: (续) 假设$A_{pq} \Rightarrow^* x$使用k+1步. 考虑第一步, 分使用$A_{pq} \to aA_{rs}b$和 $A_{pq} \to A_{pr}A_{rq}$两种情况.

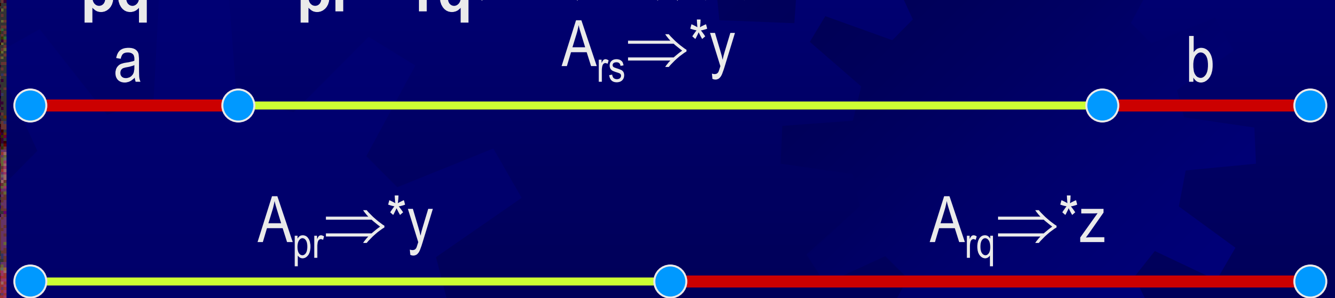


# 断言3.16

- $A_{pq} \to aA_{rs}b$
- $A_{pq} \to A_{pr}A_{rq}$
- $A_{pp} \to \varepsilon$

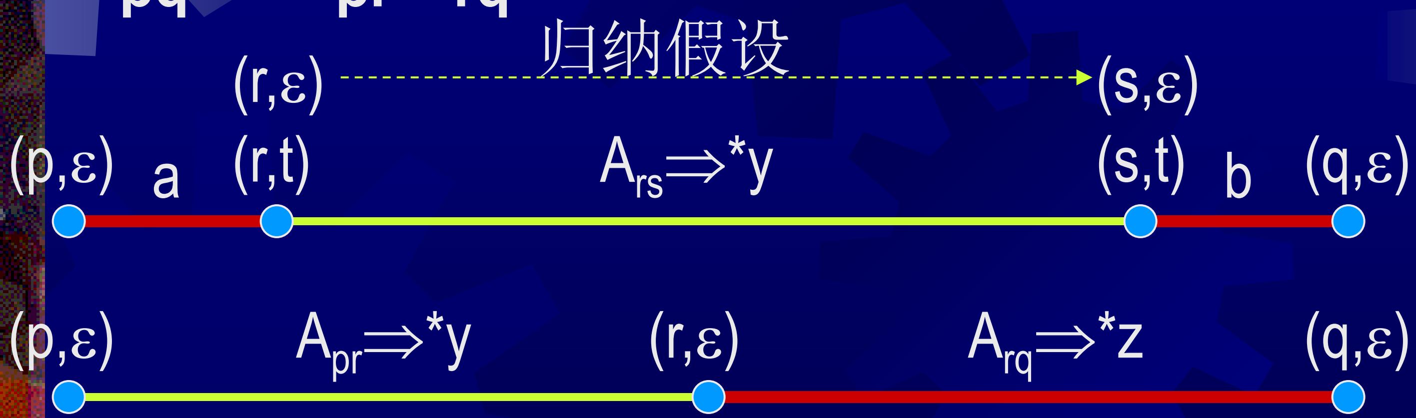



- **断言3.16**: 如果$A_{pq}$产生x, 则x把P从状态p和空栈一起带到状态q和空栈.
- **证明**: (续) 假设$A_{pq} \Rightarrow^* x$使用k+1步. 考虑第一步, 分使用$A_{pq} \to aA_{rs}b$和 $A_{pq} \to A_{pr}A_{rq}$两种情况.

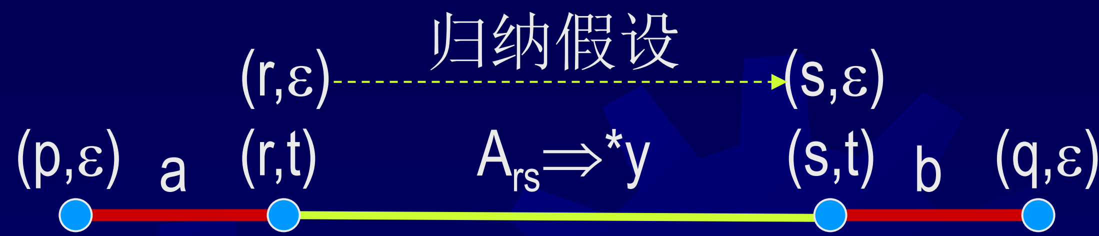


# 断言3.16

- **断言3.16**: 如果$A_{pq}$产生x, 则x把P从状态p和空栈一起带到状态q和空栈.
- **证明**: (续) 假设$A_{pq}\Rightarrow^* x$使用k+1步. 考虑第一步, 分使用$A_{pq}\rightarrow aA_{rs}b$和 $A_{pq}\rightarrow A_{pr}A_{rq}$两种情况.



# 断言3.16




- 证明: (续) (2a)设第一步使用$A_{pq}\Rightarrow aA_{rs}b$.

则$A_{rs}\Rightarrow^{*}y$使用k步, x=ayb. 根据归纳假设,

P从r和空栈读y转移到s和空栈.

因为$A_{pq}\rightarrow aA_{rs}b$是规则, 所以存在t,

使得δ(p,a,ε)包含(r,t)且δ(s,b,t)包含(q,ε),

即P从状态p和空栈开始,

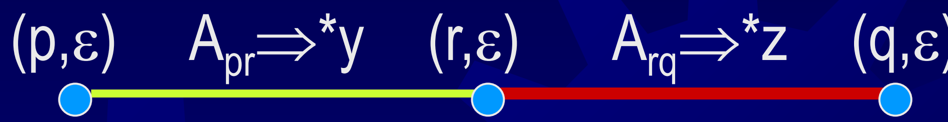
　读a后转入状态r, 把t推入栈.

P可以读y转移到s,栈中保留t.

　接着读b后转到q并把t弹出栈.

所以x把P从p和空栈带到q和空栈.

# 断言3.16




- **断言3.16**: 如果$A_{pq}$产生x, 则x把P从状态p和空栈一起带到状态q和空栈.
- **证明**: (续) (2b)

设第一步使用$A_{pq}\to A_{pr}A_{rq}$,

则都在k步内$A_{pr}\Rightarrow^* y$, $A_{rq}\Rightarrow^* z$, $x=yz$.

根据归纳假设，y把P从p带到r,

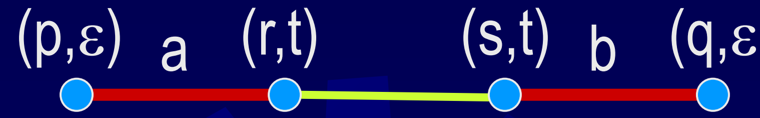
z把P从r带到q.

因为在派生开始和结束时都是空栈,

所以x 把P从p和空栈带到q和空栈.　　#

# 断言3.17

- $A_{pq} \to aA_{rs}b$
- $A_{pq} \to A_{pr}A_{rq}$
- $A_{pp} \to \varepsilon$




- **断言3.17**: 如果x把P从状态p和空栈一起带到状态q和空栈,则$A_{pq}$产生x.
- **证明**:(对x把P从状态p和空栈带到状态q和空栈的步数进行归纳).

(1) 0步计算.

设计算开始和结束时都是状态p,

要证明$A_{pp} \Rightarrow^* x$.

在0步内, P只有读空串的时间, 所以$x=\varepsilon$.

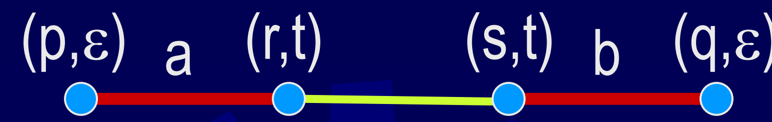
根据G的构造, G有规则$A_{pp} \to \varepsilon$.

(2) 假设断言对不超过k步的计算成立.

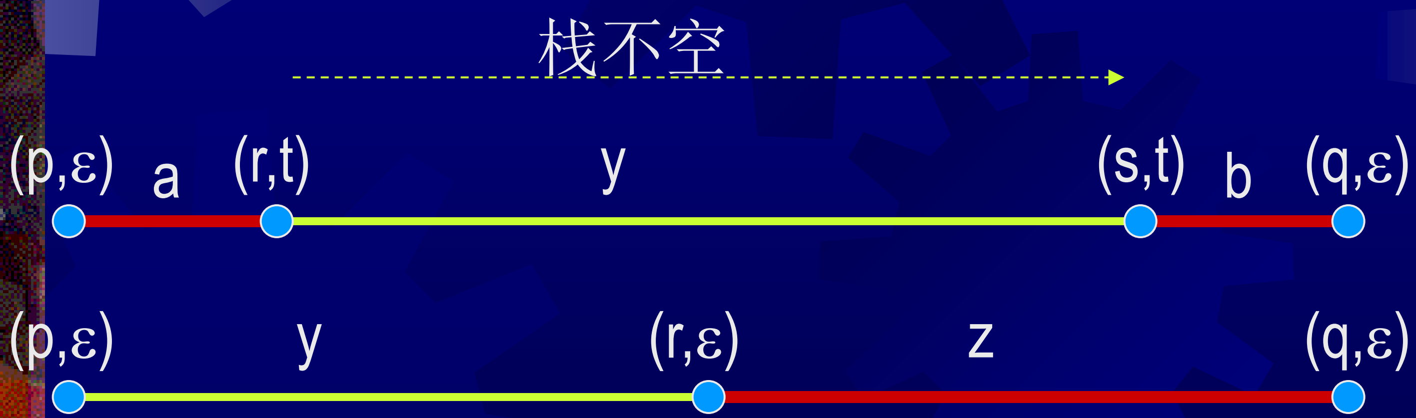
要证明断言对k+1步的计算也成立.

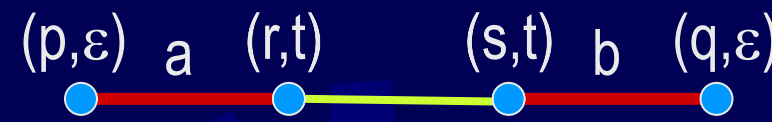
# 断言3.17

- $A_{pq} \to aA_{rs}b$
- $A_{pq} \to A_{pr}A_{rq}$
- $A_{pp} \to \varepsilon$




- **断言3.17**: 如果x把P从状态p和空栈一起带到状态q和空栈,则$A_{pq}$产生x.
- **证明(续)**: 设P有长度为k+1的计算(p,ε),…,(q,ε). 根据是否在计算的中间某个地方栈为空, 分两种情况.

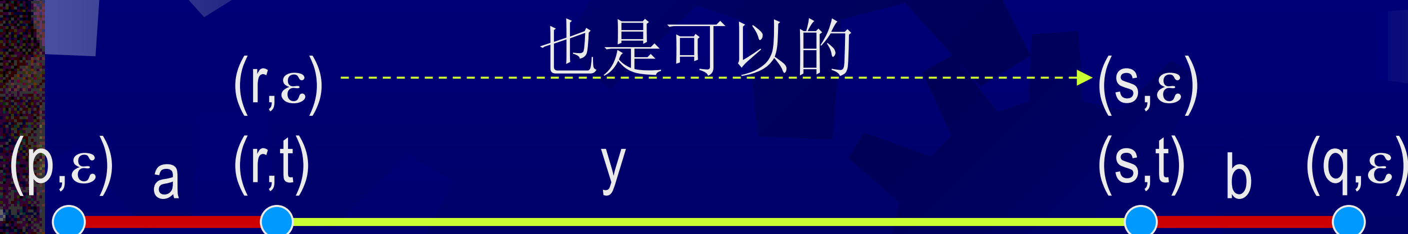


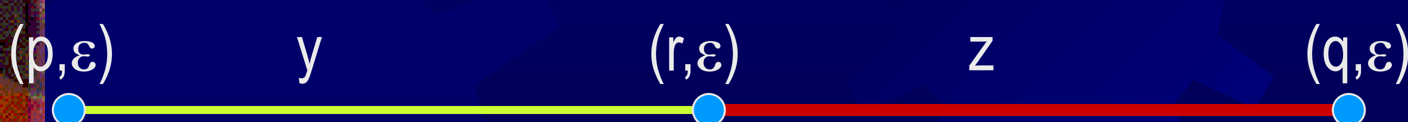



- $A_{pq} \to aA_{rs}b$
- $A_{pq} \to A_{pr}A_{rq}$
- $A_{pp} \to \varepsilon$

# 断言3.17

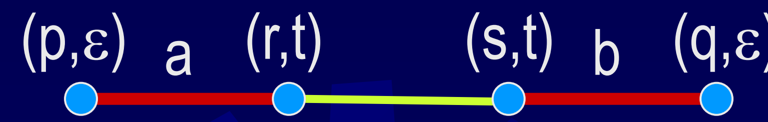
- **断言3.17: 如果x把P从状态p和空栈一起带到状态q和空栈,则$A_{pq}$产生x.**
- **证明(续): 设P有长度为k+1的计算(p,ε),…,(q,ε). 根据是否在计算的中间某个地方栈为空, 分两种情况.**



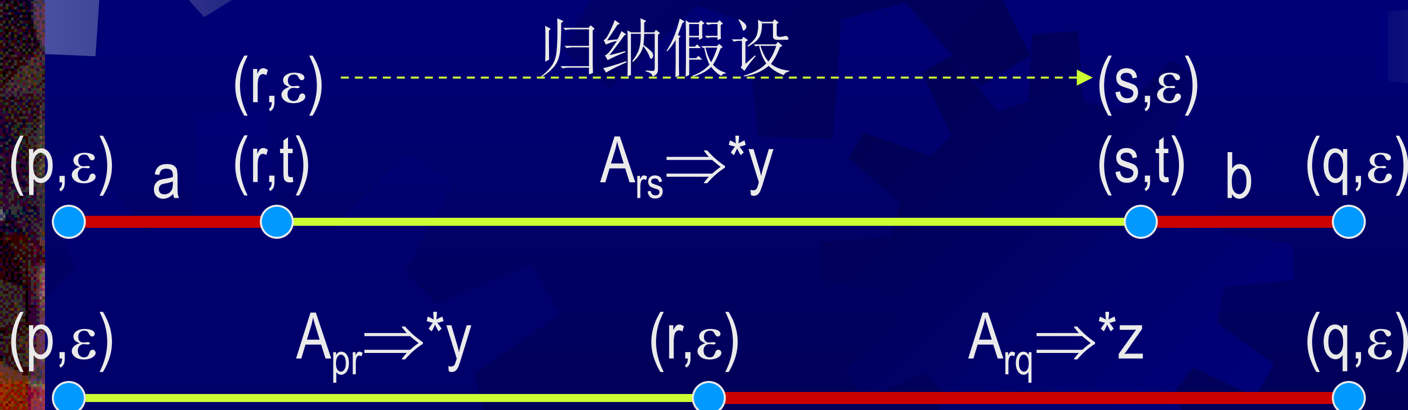



# 断言3.17

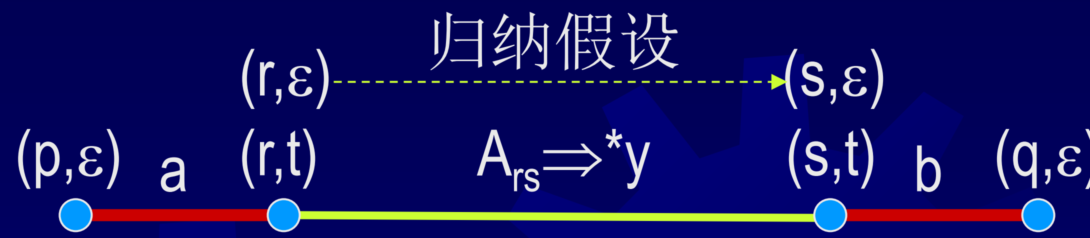
- $A_{pq} \to aA_{rs}b$
- $A_{pq} \to A_{pr}A_{rq}$
- $A_{pp} \to \varepsilon$




- **断言3.17**: 如果x把P从状态p和空栈一起带到状态q和空栈,则$A_{pq}$产生x.
- **证明(续)**: 设P有长度为k+1的计算(p,ε),…,(q,ε). 根据是否在计算的中间某个地方栈为空, 分两种情况.



# 断言3.17




- **断言3.17**: 如果x把P从状态p和空栈一起带到状态q和空栈,则$A_{pq}$产生x.
- **证明(续)**: (1) 设计算为(p,ε),…,(q,ε),

  中间没有空栈, 所读输入符号为x.

则第一步入栈与最后一步出栈的是同一个符号.

设第一步所读输入符号为a, 进入状态r,

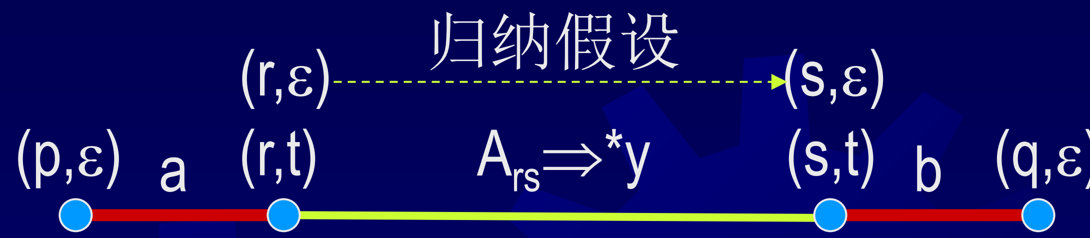
最后一步在状态s所读输入符号为b, 设x=ayb.

那么有计算(p,ε),(r,t),…, (s,t),(q,ε).

所以δ(p,a,ε)包含(r,t)且δ(s,b,t)包含(q,ε),

所以G中有规则$A_{pq}\rightarrow aA_{rs}b$.

# 断言3.17




- **断言3.17: 如果x把P从状态p和空栈一起带到状态q和空栈,则$A_{pq}$产生x.**
- **证明(续): (1) (续) 对输入y的计算是**

  **(r,t),…, (s,t), 长度不超过k-1,**

  **不会出现空栈,**

  **所以对输入y的计算可以是**

  **(r,ε),…, (s,ε), 长度不超过k-1,**

  **根据归纳假设, 有$A_{rs}\Rightarrow^* y$,**

  **由于有规则$A_{pq}\to aA_{rs}b$.**

  **所以$A_{pq}\Rightarrow^* x$.**

# 断言3.17

$(p,\varepsilon)$ $A_{pr}\Rightarrow^* y$ $(r,\varepsilon)$ $A_{rq}\Rightarrow^* z$ $(q,\varepsilon)$

- **断言3.17**: 如果x把P从状态p和空栈一起带到状态q和空栈, 则$A_{pq}$产生x.
- **证明(续)**: (2) 设计算为

$(p,\varepsilon),\ldots(r,\varepsilon),\ldots, (q,\varepsilon)$, 所读输入为x.

则计算$(p,\varepsilon),\ldots(r,\varepsilon)$不超过k步,

设所读输入为y.

计算$(r,\varepsilon),\ldots, (q,\varepsilon)$也不超过k步,

设所读输入为z, x=yz.

根据归纳假设, 有$A_{pr}\Rightarrow^* y$, $A_{rq}\Rightarrow^* z$.

由于G中有规则$A_{pq}\to A_{pr}A_{rq}$,

所以$A_{pq}\Rightarrow^* x$. #

# 引理3.15


- **引理3.15**: 如果一个语言被PDA识别, 则这个语言是CFL.
- **证明**:设PDA $P=(Q,\Sigma,\Gamma,\delta,q_0,\{q_{accept}\})$, 构造等价CFG

  $G=(\{A_{pq}|p,q\in Q\},\Sigma,R,A_{q0qaccept})$,

  R包含如下规则:
  - $A_{pq}\to aA_{rs}b$（任何 $p,q,r,s\in Q$, $a,b\in\Sigma_\varepsilon$, $t\in\Gamma$, $\delta(p,a,\varepsilon)$包含$(r,t)$且 $\delta(s,b,t)$包含$(q,\varepsilon)$ ）
  - $A_{pq}\to A_{pr}A_{rq}$（任何 $p,q,r\in Q$ ）
  - $A_{pp}\to\varepsilon$ ( 任何 $p\in Q$ ) #

# 作业

- 书本 2.1， 2.4， 2.6， 2.14