上下文无关文法

吴 铤

2017 年 10 月 23 日

上下文无关文法

上下文无关文法

上下文无关文法

“回文”语言：$C = \{w|w = w^R, w \in \{0,1\}^*\}$，其中$w^R$表示$s$的反转。

“回文”语言：$C = \{w|w = w^R, w \in \{0,1\}^*\}$，其中$w^R$表示$s$的反转。

- C不是正则语言；

"回文"语言：$C=\{w|w=w^R, w\in\{0,1\}^*\}$，其中$w^R$表示$s$的反转。

- C不是正则语言；
- 回文的产生：

"回文"语言：$C = \{w|w = w^R, w \in \{0,1\}^*\}$，其中$w^R$表示$s$的反转。

- C不是正则语言；
- 回文的产生：
  - $\varepsilon, 1, 0$是回文；

“回文”语言：$C=\{w|w=w^R, w\in\{0,1\}^*\}$，其中$w^R$表示$s$的反转。

- C不是正则语言；
- 回文的产生：
  - $\varepsilon, 1, 0$是回文；
  - 如果w是回文，则$0w0, 1w1$也是回文；

"回文"语言：$C = \{w | w = w^R, w \in \{0,1\}^*\}$，其中$w^R$表示$s$的反转。

- C不是正则语言；
- 回文的产生：
  - $\varepsilon, 1, 0$是回文；
  - 如果w是回文，则$0w0, 1w1$也是回文；

1. $P \rightarrow \varepsilon, P \rightarrow 0, P \rightarrow 1;$

"回文"语言：$C=\{w|w=w^R, w\in\{0,1\}^*\}$，其中$w^R$表示$s$的反转。

- C不是正则语言；
- 回文的产生：
  - $\varepsilon, 1, 0$是回文；
  - 如果w是回文，则$0w0, 1w1$也是回文；

1. $P\rightarrow\varepsilon, P\rightarrow 0, P\rightarrow 1$;
2. $P\rightarrow 0P0, P\rightarrow 1P1$;

- 替换规则/产生式：$P \to \varepsilon, P \to 0, P \to 1$
  $P \to 0P0, P \to 1P1;$

- 替换规则/产生式：$P \to \varepsilon, P \to 0, P \to 1$
  $P \to 0P0, P \to 1P1;$
  记为$P \to \varepsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$

- **替换规则/产生式**：$P \rightarrow \varepsilon, P \rightarrow 0, P \rightarrow 1$
  $P \rightarrow 0P0, P \rightarrow 1P1;$
  记为$P \rightarrow \varepsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$
- **变元**：P

- 替换规则/产生式：$P \rightarrow \varepsilon, P \rightarrow 0, P \rightarrow 1$
  $P \rightarrow 0P0, P \rightarrow 1P1;$
  记为$P \rightarrow \varepsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$
- 变元：P
- 终结符：$0, 1$

- 替换规则/产生式：$P \to \varepsilon, P \to 0, P \to 1$
  $P \to 0P0, P \to 1P1;$
  记为$P \to \varepsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$
- 变元：P
- 终结符：$0, 1$
- 起始变元：P

上下文无关文法G_1：替换规则/产生式

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

上下文无关文法G_1：替换规则/产生式

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

起始变元：A；

上下文无关文法G_1：替换规则/产生式

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

起始变元：A；终结符：$0, 1, \#$

上下文无关文法G_1：替换规则/产生式

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

起始变元：A；终结符：$0, 1, \#$

语言的生成规则

上下文无关文法G_1：替换规则/产生式

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

起始变元：A；终结符：$0, 1, \#$

语言的生成规则

1. 写下起始变元；

上下文无关文法G_1：替换规则/产生式

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

起始变元：A；终结符：$0, 1, \#$

语言的生成规则

1. 写下起始变元；
2. 取一个已写下的变元，找到以该变元开始的规则并将该变元替换为规则右侧的字符串；

上下文无关文法G_1：替换规则/产生式

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

起始变元：A；终结符：$0, 1, \#$

语言的生成规则

1. 写下起始变元；
2. 取一个已写下的变元，找到以该变元开始的规则并将该变元替换为规则右侧的字符串；
3. 重复步骤2，直至写下的字符串没有变元为止。

上下文无关文法G_1：替换规则/产生式

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

起始变元：A；终结符：$0, 1, \#$

语言的生成规则

1. 写下起始变元；
2. 取一个已写下的变元，找到以该变元开始的规则并将该变元替换为规则右侧的字符串；
3. 重复步骤2，直至写下的字符串没有变元为止。

最左派生：每一步都是替换最左边剩下的变元

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

字符串000#111的派生过程：

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

字符串000#111的派生过程：

$$A$$

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

字符串000#111的派生过程：

$$A \Rightarrow 0A1$$

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

字符串000#111的派生过程：

$$A \Rightarrow 0A1 \Rightarrow 00A11$$

$$A \to 0A1$$

$$A \to B$$

$$B \to \#$$

字符串000#111的派生过程：

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111$$

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

字符串000#111的派生过程：

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111$$

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

字符串000#111的派生过程：

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

定义1 (上下文无关文法(context-free grammar CFG))

定义1 (上下文无关文法(context-free grammar CFG))

上下文无关文法是一个4元组$(V, \sum, R, S)$，且

定义1 (上下文无关文法(context-free grammar CFG))

上下文无关文法是一个4元组$(V, \sum, R, S)$，且

1. 有穷集合V被称为**变元集**;

定义1 (上下文无关文法(context-free grammar CFG))

上下文无关文法是一个4元组$(V, \sum, R, S)$，且

1. 有穷集合V被称为变元集;
2. $\sum$是与V不相交的有穷集合，被称为终结符集;

定义1 (上下文无关文法(context-free grammar CFG))

上下文无关文法是一个4元组$(V, \sum, R, S)$，且

1. 有穷集合V被称为变元集;
2. $\sum$是与V不相交的有穷集合，被称为终结符集;
3. 有穷规则集R中的规则由一个变元和一个由变元及终结符组成的字符串构成;

定义1 (上下文无关文法(context-free grammar CFG))

上下文无关文法是一个4元组$(V, \sum, R, S)$，且

1. 有穷集合V被称为变元集;
2. $\sum$是与V不相交的有穷集合，被称为终结符集;
3. 有穷规则集R中的规则由一个变元和一个由变元及终结符组成的字符串构成;
4. 起始变元$S \in V$

- 若u, v, ω是由变元及终结符构成的字符串，而$A \to \omega$是一条规则，则称uAv生成$u\omega v$。记为$uAv \Rightarrow u\omega v$;

- 若u, v, ω是由变元及终结符构成的字符串，而$A \to \omega$是一条规则，则称uAv生成$u\omega v$。记为$uAv \Rightarrow u\omega v$;
- 若$u = v$，或存在$u_1, \cdots, u_k$，使得

$$u \Rightarrow u_1 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v$$

则称u派生v。记为$u \overset{*}{\Rightarrow} v$;

- 若u, v, ω是由变元及终结符构成的字符串，而$A \to \omega$是一条规则，则称uAv生成$u\omega v$。记为$uAv \Rightarrow u\omega v$；
- 若$u = v$，或存在$u_1, \cdots, u_k$，使得

$$u \Rightarrow u_1 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v$$

  则称u派生v。记为$u \overset{*}{\Rightarrow} v$；
- $\{\omega \in \sum^* | S \overset{*}{\Rightarrow} \omega\}$被称为是该文法的语言；

- 若u, v, ω是由变元及终结符构成的字符串，而$A \rightarrow \omega$是一条规则，则称uAv生成$u\omega v$。记为$uAv \Rightarrow u\omega v$;
- 若$u = v$，或存在$u_1, \cdots, u_k$，使得

$$u \Rightarrow u_1 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v$$

则称u派生v。记为$u \stackrel{*}{\Rightarrow} v$;

- $\{\omega \in \sum^* | S \stackrel{*}{\Rightarrow} \omega\}$被称为是该文法的语言;
- 能够用上下文无关文法生成的语言被称为上下文无关语言(CFL);

例1

给出文法G:

$$S \to aAb \mid bBa$$
$$A \to aAb \mid bBa$$
$$B \to c$$

判定$aabcabb$是否能够由其派生得出?

- 自顶向下：

- 自顶向下：
  1. 由于给定的串是以a开头的，所以选择$S \rightarrow aAb$，即$S \Rightarrow aAb$；

- 自顶向下：
  1. 由于给定的串是以a开头的，所以选择$S \to aAb$，即$S \Rightarrow aAb$;
  2. 由于其中只有变元A且第二个符号是a，所以选择$A \to aAb$，即$aAb \Rightarrow aaAbb$;

- 自顶向下：
  1. 由于给定的串是以a开头的，所以选择$S \to aAb$，即$S \Rightarrow aAb$；
  2. 由于其中只有变元A且第二个符号是a，所以选择$A \to aAb$，即$aAb \Rightarrow aaAbb$；
  3. 由于其中只有变元A且第三个符号是b，所以选择$A \to bBa$，即$aaAbb \Rightarrow aabBabb$；

- 自顶向下：
  1. 由于给定的串是以a开头的，所以选择$S \to aAb$，即$S \Rightarrow aAb$；
  2. 由于其中只有变元A且第二个符号是a，所以选择$A \to aAb$，即$aAb \Rightarrow aaAbb$；
  3. 由于其中只有变元A且第三个符号是b，所以选择$A \to bBa$，即$aaAbb \Rightarrow aabBabb$；
  4. 由于其中只有变元B，所以选择$B \to c$，即$aabBabb \Rightarrow aabcabb$；

语法分析树：对于文法$(V, \sum, R, S)$来说，其语法分析树是满足以下条件的树：

语法分析树：对于文法$(V, \sum, R, S)$来说，其语法分析树是满足以下条件的树：

- 根节点的标号为S；

语法分析树：对于文法$(V, \sum, R, S)$来说，其语法分析树是满足以下条件的树：

- 根节点的标号为S；
- 每个内部节点的标号是V中的一个变元；

语法分析树：对于文法$(V, \sum, R, S)$来说，其语法分析树是满足以下条件的树：

- 根节点的标号为S；
- 每个内部节点的标号是V中的一个变元；
- 每个叶节点的标号可以是一个变元、一个终止符或ε；

语法分析树：对于文法$(V, \sum, R, S)$来说，其语法分析树是满足以下条件的树：

- 根节点的标号为S；
- 每个内部节点的标号是V中的一个变元；
- 每个叶节点的标号可以是一个变元、一个终止符或ε；
- 如果叶节点的标号是ε，则其一定是其父节点唯一的子节点；

语法分析树：对于文法$(V, \sum, R, S)$来说，其语法分析树是满足以下条件的树：

- 根节点的标号为S；
- 每个内部节点的标号是V中的一个变元；
- 每个叶节点的标号可以是一个变元、一个终止符或ε；
- 如果叶节点的标号是ε，则其一定是其父节点唯一的子节点；
- 如果某个内部节点的标号是A,且其子节点的标号从左至右分别为$X_1, X_2, \cdots, X_k$，则$A \to X_1X_2\cdots X_k$必定是一个产生式。

语法分析树： 对于文法$(V,\sum,R,S)$来说，其语法分析树是满足以下条件的树：

- 根节点的标号为S；
- 每个内部节点的标号是V中的一个变元；
- 每个叶节点的标号可以是一个变元、一个终止符或ε；
- 如果叶节点的标号是ε，则其一定是其父节点唯一的子节点；
- 如果某个内部节点的标号是A,且其子节点的标号从左至右分别为$X_1,X_2,\cdots,X_k$，则$A\to X_1X_2\cdots X_k$必定是一个产生式。

画出$A\to 0A1\mid B, B\to \#$派生出000#111对应的语法分析树

例2 (设计产生所有匹配的圆括号串的文法)

例2 (设计产生所有匹配的圆括号串的文法)

$$G = (\{B\}, \{(,)\}, P, B)$$

例2 (设计产生所有匹配的圆括号串的文法)

$G = (\{B\}, \{(,)\}, P, B)$

- 两个括号匹配的串连接后所得的串仍然是括号匹配的:

例2 (设计产生所有匹配的圆括号串的文法)

$G = (\{B\}, \{(,)\}, P, B)$

- 两个括号匹配的串连接后所得的串仍然是括号匹配的：$B \rightarrow BB$

例2 (设计产生所有匹配的圆括号串的文法)

$G = (\{B\}, \{(,)\}, P, B)$

- 两个括号匹配的串连接后所得的串仍然是括号匹配的：$B \to BB$
- 用括号把一个括号匹配的串括起来仍然是括号匹配的：$B \to (B)$

例2 (设计产生所有匹配的圆括号串的文法)

$G = (\{B\}, \{(,)\}, P, B)$

- 两个括号匹配的串连接后所得的串仍然是括号匹配的：$B \to BB$
- 用括号把一个括号匹配的串括起来仍然是括号匹配的：$B \to (B)$

例2 (设计产生所有匹配的圆括号串的文法)

$G = (\{B\}, \{(,)\}, P, B)$

- 两个括号匹配的串连接后所得的串仍然是括号匹配的：$B \to BB$
- 用括号把一个括号匹配的串括起来仍然是括号匹配的：$B \to (B)$
- 空串是括号匹配的串：

例2 (设计产生所有匹配的圆括号串的文法)

$G = (\{B\}, \{(,)\}, P, B)$

- 两个括号匹配的串连接后所得的串仍然是括号匹配的：$B \rightarrow BB$
- 用括号把一个括号匹配的串括起来仍然是括号匹配的：$B \rightarrow (B)$
- 空串是括号匹配的串：$B \rightarrow \varepsilon$

例2 (设计产生所有匹配的圆括号串的文法)

$$G = (\{B\}, \{(,)\}, P, B)$$

- 两个括号匹配的串连接后所得的串仍然是括号匹配的：$B \to BB$
- 用括号把一个括号匹配的串括起来仍然是括号匹配的：$B \to (B)$
- 空串是括号匹配的串：$B \to \varepsilon$

即P包含$B \to BB \mid (B) \mid \varepsilon$

例2 (设计产生所有匹配的圆括号串的文法)

$G = (\{B\}, \{(,)\}, P, B)$

- 两个括号匹配的串连接后所得的串仍然是括号匹配的：$B \to BB$
- 用括号把一个括号匹配的串括起来仍然是括号匹配的：$B \to (B)$
- 空串是括号匹配的串：$B \to \varepsilon$

即P包含$B \to BB \mid (B) \mid \varepsilon$

$L(G)$不是正则语言!

例3 (对$+, \times, (\quad)$符合算术计算次序要求表达式的CFG)

例3 (对$+, \times, (\quad)$符合算术计算次序要求表达式的CFG)

- 表达式可以是单个“项”或者是另一个表达式与“项”的“和”

例3 (对$+, \times, (\quad)$符合算术计算次序要求表达式的CFG)

- 表达式可以是单个“项”或者是另一个表达式与“项”的“和”
- 项可以是单个“因子”或项与“因子”的“乘积”

例3 (对+, ×, ( )符合算术计算次序要求表达式的CFG)

- 表达式可以是单个“项”或者是另一个表达式与“项”的“和”
- 项可以是单个“因子”或项与“因子”的“乘积”
- 因子可以是单个常量或是某个被“()”包围的表达式

例4 (对$+, \times, ()$符合算术计算次序要求的CFG)

考虑文法$G = (V, \sum, R, < EXPR >)$,其中

$$V = \{< EXPR >, < TERM >, < FACTOR >\}$$
$$\sum = \{a, +, \times, (,)\}$$

规则集合

$$< EXPR > \rightarrow$$

例4 (对$+, \times, ()$符合算术计算次序要求的CFG)

考虑文法$G = (V, \sum, R, < EXPR >)$,其中

$$V = \{< EXPR >, < TERM >, < FACTOR >\}$$
$$\sum = \{a, +, \times, (,)\}$$

规则集合

$$< EXPR > \rightarrow \; < EXPR > + < TERM >$$

例4 (对$+, \times, ()$符合算术计算次序要求的CFG)

考虑文法$G = (V, \sum, R, < EXPR >)$,其中

$$V = \{< EXPR >, < TERM >, < FACTOR >\}$$
$$\sum = \{a, +, \times, (,)\}$$

规则集合

$$< EXPR > \rightarrow \; < EXPR > + < TERM > | < TERM >$$

例4 (对$+, \times, ()$符合算术计算次序要求的CFG)

考虑文法$G = (V, \sum, R, < EXPR >)$,其中

$$V = \{< EXPR >, < TERM >, < FACTOR >\}$$
$$\sum = \{a, +, \times, (,)\}$$

规则集合

$$< EXPR > \rightarrow \ < EXPR > + < TERM > | < TERM >$$

$$< TERM >$$

例4 (对$+, \times, ()$符合算术计算次序要求的CFG)

考虑文法$G = (V, \sum, R, < EXPR >)$,其中

$$V = \{< EXPR >, < TERM >, < FACTOR >\}$$
$$\sum = \{a, +, \times, (,)\}$$

规则集合

$$< EXPR > \rightarrow \ < EXPR > + < TERM > | < TERM >$$
$$< TERM > \ \rightarrow < TERM > \times < FACTOR > | < FACTOR >$$

例4 (对$+, \times, ()$符合算术计算次序要求的CFG)

考虑文法$G = (V, \sum, R, < EXPR >)$,其中

$$V = \{< EXPR >, < TERM >, < FACTOR >\}$$
$$\sum = \{a, +, \times, (,)\}$$

规则集合

$$< EXPR > \rightarrow \ < EXPR > + < TERM > | < TERM >$$
$$< TERM > \ \rightarrow < TERM > \times < FACTOR > | < FACTOR >$$
$$< FACTOR >$$

例4 (对$+, \times, ()$符合算术计算次序要求的CFG)

考虑文法$G = (V, \sum, R, < EXPR >)$,其中

$$V = \{< EXPR >, < TERM >, < FACTOR >\}$$
$$\sum = \{a, +, \times, (,)\}$$

规则集合

$$< EXPR > \rightarrow \ < EXPR > + < TERM > | < TERM >$$
$$< TERM > \ \rightarrow < TERM > \times < FACTOR > | < FACTOR >$$
$$< FACTOR > \ \rightarrow (< EXPR >)|a$$

例4 (对$+, \times, ()$符合算术计算次序要求的CFG)

考虑文法$G = (V, \sum, R, < EXPR >)$,其中

$$V = \{< EXPR >, < TERM >, < FACTOR >\}$$
$$\sum = \{a, +, \times, (,)\}$$

规则集合

$$< EXPR > \rightarrow \; < EXPR > + < TERM > | < TERM >$$
$$< TERM > \; \rightarrow < TERM > \times < FACTOR > | < FACTOR >$$
$$< FACTOR > \; \rightarrow (< EXPR >)|a$$

分别考虑$a + a \times a, (a + a) \times a, a \times (a + a) + a$的派生过程

考虑$\langle E\rangle \rightarrow \langle E\rangle + \langle E\rangle \mid \langle E\rangle \times \langle E\rangle \mid (\langle E\rangle) \mid a$中$a + a \times a$的派生过程

考虑$\langle E\rangle \rightarrow \langle E\rangle + \langle E\rangle \mid \langle E\rangle \times \langle E\rangle \mid (\langle E\rangle) \mid a$中$a + a \times a$的派生过程

考虑$\langle E\rangle \to \langle E\rangle + \langle E\rangle \mid \langle E\rangle \times \langle E\rangle \mid (\langle E\rangle) \mid a$
中$a + a \times a$的派生过程

$$\langle E\rangle \Rightarrow \langle E\rangle \times \langle E\rangle \Rightarrow \langle E\rangle + \langle E\rangle \times \langle E\rangle \Rightarrow a + a \times a$$

考虑$\langle E\rangle \to \langle E\rangle + \langle E\rangle \mid \langle E\rangle \times \langle E\rangle \mid (\langle E\rangle) \mid a$中$a+a\times a$的派生过程

$$\langle E\rangle \Rightarrow \langle E\rangle + \langle E\rangle \Rightarrow \langle E\rangle + \langle E\rangle \times \langle E\rangle \Rightarrow a+a\times a$$

考虑$\langle E\rangle \rightarrow \langle E\rangle + \langle E\rangle \mid \langle E\rangle \times \langle E\rangle \mid (\langle E\rangle) \mid a$中$a+a\times a$的派生过程

$$\langle E\rangle \Rightarrow \langle E\rangle + \langle E\rangle \Rightarrow \langle E\rangle + \langle E\rangle \times \langle E\rangle \Rightarrow a+a\times a$$

定义2 (如果字符串ω在上下文无关文法G中有两个或两个以上不同的(最左)派生，则称G**歧义地**产生ω。)

- 若某个CFLG为几个CFL$G_i, i = 1, \cdots, n$的合并，则

- 若某个CFLG为几个CFL$G_i, i = 1, \cdots, n$的合并，则
  - G的规则就是将$G_i, i = 1, \cdots, n$的合并，及

- 若某个CFLGG为几个CFLG$G_i, i = 1, \cdots, n$的合并，则
  - G的规则就是将$G_i, i = 1, \cdots, n$的合并，及
  - 将$S \rightarrow S_1 \mid S_2 \mid \cdots \mid S_n$加入$G$的规则集合之中，其中$S_i$是$G_i$的起始变元。

- 若某个CFLG为几个CFL$G_i, i=1,\cdots,n$的合并，则
  - G的规则就是将$G_i, i=1,\cdots,n$的合并，及
  - 将$S \to S_1 \mid S_2 \mid \cdots \mid S_n$加入$G$的规则集合之中，其中$S_i$是$G_i$的起始变元。
- 正则语言对应文法的构造：

- 若某个CFLG为几个CFL$G_i, i = 1, \cdots, n$的合并，则
  - G的规则就是将$G_i, i = 1, \cdots, n$的合并，及
  - 将$S \to S_1 \mid S_2 \mid \cdots \mid S_n$加入$G$的规则集合之中，其中$S_i$是$G_i$的起始变元。
- 正则语言对应文法的构造：
  - 对于DFA的每个状态q_i，定义一个变元R_i；

- 若某个CFLG为几个CFL$G_i, i=1,\cdots,n$的合并，则
  - G的规则就是将$G_i, i=1,\cdots,n$的合并，及
  - 将$S \rightarrow S_1 \mid S_2 \mid \cdots \mid S_n$加入$G$的规则集合之中，其中$S_i$是$G_i$的起始变元。
- 正则语言对应文法的构造：
  - 对于DFA的每个状态q_i，定义一个变元R_i；
  - 若$\delta(q_i, a)=q_j$，则将$R_i \rightarrow aR_j$加入CFG；

- 若某个CFLGG为几个CFLG$G_i, i = 1, \cdots, n$的合并，则
  - G的规则就是将$G_i, i = 1, \cdots, n$的合并，及
  - 将$S \to S_1 \mid S_2 \mid \cdots \mid S_n$加入$G$的规则集合之中，其中$S_i$是$G_i$的起始变元。
- 正则语言对应文法的构造：
  - 对于DFA的每个状态q_i，定义一个变元R_i；
  - 若$\delta(q_i, a) = q_j$，则将$R_i \to aR_j$加入CFG；
  - 若q_i是接受状态，则将$R_i \to \varepsilon$加入CFG；

- 若某个CFLG为几个CFL$G_i, i=1,\cdots,n$的合并，则
  - G的规则就是将$G_i, i=1,\cdots,n$的合并，及
  - 将$S \to S_1 \mid S_2 \mid \cdots \mid S_n$加入$G$的规则集合之中，其中$S_i$是$G_i$的起始变元。
- 正则语言对应文法的构造：
  - 对于DFA的每个状态q_i，定义一个变元R_i；
  - 若$\delta(q_i,a)=q_j$，则将$R_i \to aR_j$加入CFG；
  - 若q_i是接受状态，则将$R_i \to \varepsilon$加入CFG；
  - 若q_0是起始状态，则将R_0作为CFG的起始变元；

- 若某个CFLG为几个CFL$G_i, i = 1, \cdots, n$的合并，则
  - G的规则就是将$G_i, i = 1, \cdots, n$的合并，及
  - 将$S \to S_1 \mid S_2 \mid \cdots \mid S_n$加入$G$的规则集合之中，其中$S_i$是$G_i$的起始变元。
- 正则语言对应文法的构造：
  - 对于DFA的每个状态q_i，定义一个变元R_i；
  - 若$\delta(q_i, a) = q_j$，则将$R_i \to aR_j$加入CFG；
  - 若q_i是接受状态，则将$R_i \to \varepsilon$加入CFG；
  - 若q_0是起始状态，则将R_0作为CFG的起始变元；
- 考察子串：

- 若某个CFLG为几个CFL$G_i, i = 1, \cdots, n$的合并，则
  - G的规则就是将$G_i, i = 1, \cdots, n$的合并，及
  - 将$S \to S_1 \mid S_2 \mid \cdots \mid S_n$加入$G$的规则集合之中，其中$S_i$是$G_i$的起始变元。
- 正则语言对应文法的构造：
  - 对于DFA的每个状态q_i，定义一个变元R_i；
  - 若$\delta(q_i, a) = q_j$，则将$R_i \to aR_j$加入CFG；
  - 若q_i是接受状态，则将$R_i \to \varepsilon$加入CFG；
  - 若q_0是起始状态，则将R_0作为CFG的起始变元；
- 考察子串：
- 递归性的构造

例5 (构造得到$\{0^n1^n|n \geq 0\} \cup \{1^n0^n|n \geq 0\}$的文法)

例5 (构造得到$\{0^n1^n|n \geq 0\} \cup \{1^n0^n|n \geq 0\}$的文法)

- $\{0^n1^n|n \geq 0\}$文法的构造:
- $\{1^n0^n|n \geq 0\}$文法的构造:

例5 (构造得到$\{0^n1^n|n \geq 0\} \cup \{1^n0^n|n \geq 0\}$的文法)

- $\{0^n1^n|n \geq 0\}$文法的构造: $S_1 \rightarrow 0S_11 \mid \varepsilon$
- $\{1^n0^n|n \geq 0\}$文法的构造:

例5 (构造得到$\{0^n1^n|n \geq 0\} \cup \{1^n0^n|n \geq 0\}$的文法)

- $\{0^n1^n|n \geq 0\}$文法的构造：$S_1 \rightarrow 0S_11 \mid \varepsilon$
- $\{1^n0^n|n \geq 0\}$文法的构造：$S_2 \rightarrow 1S_20 \mid \varepsilon$

例5 (构造得到$\{0^n1^n|n \geq 0\} \cup \{1^n0^n|n \geq 0\}$的文法)

- $\{0^n1^n|n \geq 0\}$文法的构造：$S_1 \rightarrow 0S_11 \mid \varepsilon$
- $\{1^n0^n|n \geq 0\}$文法的构造：$S_2 \rightarrow 1S_20 \mid \varepsilon$
- $S \rightarrow S_1 \mid S_2$

例6 (构造识别以下语言的文法)

1. $\{w|w$至少包含3个1$\}$
2. $\{w|w$的长度是奇数且正中间的符号是0$\}$
3. $\{wcw^R|w \in \{a,b\}^*\}$
4. $\{w|w$中a的数目比b多$, w \in \{a,b\}^*\}$

定义3 (乔姆斯基范式)

如果上下文无关文法的每个规则都具有如下形式:

$$A \to BC, \quad A \to a$$

其中a是终结符，A, B, C是任意变元且B, C不是起始变元。同时允许$S \to \varepsilon$，其中S为起始变元，则称其为乔姆斯基范式。

定义3 (乔姆斯基范式)

如果上下文无关文法的每个规则都具有如下形式:

$$A \rightarrow BC, \quad A \rightarrow a$$

其中a是终结符，A, B, C是任意变元且B, C不是起始变元。同时允许$S \rightarrow \varepsilon$，其中$S$为起始变元，则称其为乔姆斯基范式。

定理1

任意上下文无关文法都可以用乔姆斯基范式的上下文无关文法产生。

例7（乔姆斯基范式转换）

例7 (乔姆斯基范式转换)

- 新起始变元的添加：添加新起始变元S_0以及规则$S_0 \rightarrow S$：

例7 (乔姆斯基范式转换)

- 新起始变元的添加：添加新起始变元S_0以及规则$S_0 \to S$:

$$
\begin{aligned}
S &\to ASA \mid aB \\
A &\to B \mid S \\
B &\to b \mid \varepsilon
\end{aligned}
$$

例7 (乔姆斯基范式转换)

- 新起始变元的添加：添加新起始变元S_0以及规则$S_0 \rightarrow S$:

$$S_0 \rightarrow S$$

$$S \rightarrow ASA \mid aB$$
$$A \rightarrow B \mid S$$
$$B \rightarrow b \mid \varepsilon$$

例7 (乔姆斯基范式转换)

- 新起始变元的添加：添加新起始变元S_0以及规则$S_0 \to S$：

$$\begin{aligned} S &\to ASA \mid aB \\ A &\to B \mid S \\ B &\to b \mid \varepsilon \end{aligned} \qquad \begin{aligned} S_0 &\to S \\ S &\to ASA \mid aB \\ A &\to B \mid S \\ B &\to b \mid \varepsilon \end{aligned}$$

- ε规则：删除所有形如$A \to \varepsilon$的规则。同时

- ε规则：删除所有形如$A \to \varepsilon$的规则。同时
  - 对规则右边出现的每个A，删除A后得到一个新规则。

- ε规则：删除所有形如$A \rightarrow \varepsilon$的规则。同时
  - 对规则右边出现的每个A，删除A后得到一个新规则。
    - $R \rightarrow uAv$：

- ε规则：删除所有形如$A \to \varepsilon$的规则。同时
  - 对规则右边出现的每个A，删除A后得到一个新规则。
    - $R \to uAv$：$R \to uv$

- ε规则：删除所有形如$A \to \varepsilon$的规则。同时
  - 对规则右边出现的每个A，删除A后得到一个新规则。
    - $R \to uAv$：$R \to uv$
    - $R \to uAvAw$：

- ε规则：删除所有形如$A \to \varepsilon$的规则。同时
    - 对规则右边出现的每个A，删除A后得到一个新规则。
        - $R \to uAv$：$R \to uv$
        - $R \to uAvAw$：$R \to uvAw, R \to uAvw, R \to uvw$

- ε规则：删除所有形如$A \to \varepsilon$的规则。同时
  - 对规则右边出现的每个A，删除A后得到一个新规则。
    - $R \to uAv$：$R \to uv$
    - $R \to uAvAw$：$R \to uvAw, R \to uAvw, R \to uvw$
    - $R \to A$：

- ε规则：删除所有形如$A \rightarrow \varepsilon$的规则。同时
  - 对规则右边出现的每个A，删除A后得到一个新规则。
    - $R \rightarrow uAv$：$R \rightarrow uv$
    - $R \rightarrow uAvAw$：$R \rightarrow uvAw, R \rightarrow uAvw, R \rightarrow uvw$
    - $R \rightarrow A$：$R \rightarrow \varepsilon$，除非其已经删除过

- ε规则：删除所有形如$A \to \varepsilon$的规则。同时
  - 对规则右边出现的每个A，删除A后得到一个新规则。
    - $R \to uAv$：$R \to uv$
    - $R \to uAvAw$：$R \to uvAw, R \to uAvw, R \to uvw$
    - $R \to A$：$R \to \varepsilon$，除非其已经删除过
  - 直至删除所有不包括起始变元的ε规则。

- ε规则：删除所有形如$A \rightarrow \varepsilon$的规则

$$S_0 \rightarrow S$$
$$S \rightarrow ASA \mid aB$$

$$A \rightarrow B \mid S$$
$$B \rightarrow b \mid \varepsilon$$

- ε规则：删除所有形如$A \to \varepsilon$的规则

$$S_0 \to S$$

$$S \to ASA \mid aB$$

$$A \to B \mid S$$

$$B \to b \mid \varepsilon \qquad\qquad B \to b$$

- ε规则：删除所有形如$A \to \varepsilon$的规则

$$S_0 \to S$$
$$S \to ASA \mid aB$$

$$A \to B \mid S \qquad A \to B \mid S \mid \varepsilon$$
$$B \to b \mid \varepsilon \qquad B \to b$$

- ε规则：删除所有形如$A \to \varepsilon$的规则

$$S_0 \to S$$

$$S \to ASA \mid aB \qquad S \to ASA \mid aB \mid a$$

$$A \to B \mid S \qquad A \to B \mid S \mid \varepsilon$$

$$B \to b \mid \varepsilon \qquad B \to b$$

- ε规则：删除所有形如$A \to \varepsilon$的规则

$$S_0 \to S \qquad S_0 \to S$$

$$S \to ASA \mid aB \qquad S \to ASA \mid aB \mid a$$

$$A \to B \mid S \qquad A \to B \mid S \mid \varepsilon$$

$$B \to b \mid \varepsilon \qquad B \to b$$

- ε规则：删除所有形如$A \rightarrow \varepsilon$的规则

$$S_0 \rightarrow S \qquad S_0 \rightarrow S$$

$$S \rightarrow ASA \mid aB \qquad S \rightarrow ASA \mid aB \mid a$$

$$A \rightarrow B \mid S \qquad A \rightarrow B \mid S \mid \varepsilon \qquad A \rightarrow B \mid S$$

$$B \rightarrow b \mid \varepsilon \qquad B \rightarrow b$$

- ε规则：删除所有形如$A \rightarrow \varepsilon$的规则

$$S_0 \rightarrow S \qquad S_0 \rightarrow S$$

$$S \rightarrow ASA \mid aB \qquad S \rightarrow ASA \mid aB \mid a \qquad S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S$$

$$A \rightarrow B \mid S \qquad A \rightarrow B \mid S \mid \varepsilon \qquad A \rightarrow B \mid S$$

$$B \rightarrow b \mid \varepsilon \qquad B \rightarrow b$$

- ε规则：删除所有形如$A \to \varepsilon$的规则

$$
\begin{array}{lll}
S_0 \to S & S_0 \to S & S_0 \to S \\
S \to ASA \mid aB & S \to ASA \mid aB \mid a & S \to ASA \mid aB \mid a \mid {\color{red}SA} \\
 & & \quad \mid {\color{red}AS} \mid {\color{red}S} \\
A \to B \mid S & A \to B \mid S \mid \varepsilon & A \to B \mid S \\
B \to b \mid \varepsilon & B \to b & B \to b
\end{array}
$$

- 单一规则：删除所有形如$A \to B$的规则,然后只要有一条规则$B \to u$，则添加规则$A \to u$(除非其已是被删除的)

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid aB \mid a \\
&\quad \mid SA \mid AS \mid S \\
A &\to B \mid S \\
B &\to b
\end{aligned}
$$

- 单一规则：删除所有形如$A \to B$的规则,然后只要有一条规则$B \to u$，则添加规则$A \to u$(除非其已是被删除的)

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid aB \mid a \\
&\quad \mid SA \mid AS \mid S \\
A &\to B \mid S \\
B &\to b
\end{aligned}
\qquad
\begin{aligned}
S &\to ASA \mid aB \mid a \\
&\quad \mid SA \mid AS
\end{aligned}
$$

- 单一规则：删除所有形如$A \to B$的规则,然后只要有一条规则$B \to u$，则添加规则$A \to u$(除非其已是被删除的)

$$\begin{aligned} S_0 &\to S \\ S &\to ASA \mid aB \mid a \\ &\mid SA \mid AS \mid S \\ A &\to B \mid S \\ B &\to b \end{aligned} \qquad \begin{aligned} S_0 &\to S \\ S &\to ASA \mid aB \mid a \\ &\mid SA \mid AS \\ A &\to B \mid S \end{aligned}$$

- 单一规则：删除所有形如$A \to B$的规则,然后只要有一条规则$B \to u$，则添加规则$A \to u$(除非其已是被删除的)

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid aB \mid a \\
&\quad \mid SA \mid AS \mid S \\
A &\to B \mid S \\
B &\to b
\end{aligned}
\qquad
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid aB \mid a \\
&\quad \mid SA \mid AS \\
A &\to B \mid S \\
B &\to b
\end{aligned}
$$

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow ASA|aB|a|SA|AS \\
A &\rightarrow B|S \\
B &\rightarrow b
\end{aligned}
$$

$$S_0 \rightarrow S$$
$$S \rightarrow ASA|aB|a|SA|AS$$
$$A \rightarrow B|S$$
$$B \rightarrow b$$

$$S_0 \rightarrow ASA|aB|a|SA|AS$$

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow ASA|aB|a|SA|AS \\ A &\rightarrow B|S \\ B &\rightarrow b \end{aligned}$$

$$\begin{aligned} S_0 &\rightarrow ASA|aB|a|SA|AS \\ S &\rightarrow ASA|aB|a|SA|AS \\ A &\rightarrow B|S \\ B &\rightarrow b \end{aligned}$$

$$S_0 \to ASA|aB|a|SA|AS$$
$$S \to ASA|aB|a|SA|AS$$
$$A \to B|S$$
$$B \to b$$

$$S_0 \rightarrow ASA|aB|a|SA|AS$$

$$S \rightarrow ASA|aB|a|SA|AS$$

$$A \rightarrow B|S \qquad\qquad A \rightarrow S|b$$

$$B \rightarrow b$$

$$S_0 \to ASA|aB|a|SA|AS$$
$$S \to ASA|aB|a|SA|AS$$
$$A \to B|S$$
$$B \to b$$

$$S_0 \to ASA|aB|a|SA|AS$$
$$S \to ASA|aB|a|SA|AS$$
$$A \to S|b$$
$$B \to b$$

$$S_0 \rightarrow ASA|aB|a|SA|AS$$
$$S \rightarrow ASA|aB|a|SA|AS$$
$$A \rightarrow S|b$$
$$B \rightarrow b$$

$$S_0 \rightarrow ASA|aB|a|SA|AS$$

$$S \rightarrow ASA|aB|a|SA|AS$$

$$A \rightarrow S|b$$

$$B \rightarrow b$$

$$A \rightarrow b|ASA|aB|a|SA|AS$$

$$S_0 \to ASA|aB|a|SA|AS$$
$$S \to ASA|aB|a|SA|AS$$
$$A \to S|b$$
$$B \to b$$

$$S_0 \to ASA|aB|a|SA|AS$$
$$S \to ASA|aB|a|SA|AS$$
$$A \to b|\color{red}{ASA|aB|a|SA|AS}$$
$$B \to b$$

- 添加新的变元和规则，使规则转化为合适的形式：

- 添加新的变元和规则，使规则转化为合适的形式：
  - 把每个规则$A \to u_1u_2\cdots u_k$替换为

- 添加新的变元和规则，使规则转化为合适的形式：
  - 把每个规则$A \to u_1 u_2 \cdots u_k$替换为
    - 当$k \geq 3$时，$A \to u_1 A_1, A_1 \to u_2 A_2, \cdots, A_{k-2} \to u_{k-1} u_k$

- 添加新的变元和规则，使规则转化为合适的形式：
  - 把每个规则$A \to u_1 u_2 \cdots u_k$替换为
    - 当$k \geq 3$时，$A \to u_1 A_1, A_1 \to u_2 A_2, \cdots, A_{k-2} \to u_{k-1} u_k$
    - 当$k = 2$时，用U_i替换终结符u_i，并增加$U_i \to u_i$

- 添加新的变元和规则，使规则转化为合适的形式：

$$S_0 \to ASA|aB|a|SA|AS$$

$$S \to ASA|aB|a|SA|AS$$

$$A \to b|ASA|aB|a|SA|AS$$

$$B \to b$$

- 添加新的变元和规则，使规则转化为合适的形式：

$$S_0 \rightarrow ASA|aB|a|SA|AS$$

$$S \rightarrow ASA|aB|a|SA|AS$$

$$A \rightarrow b|ASA|aB|a|SA|AS$$

$$B \rightarrow b \qquad\qquad B \rightarrow b$$

- 添加新的变元和规则，使规则转化为合适的形式：

$$S_0 \rightarrow ASA|aB|a|SA|AS$$

$$S \rightarrow ASA|aB|a|SA|AS$$

$$A \rightarrow b|ASA|aB|a|SA|AS$$

$$B \rightarrow b$$

$$B \rightarrow b$$

$$U \rightarrow a$$

$$A_1 \rightarrow SA$$

- 添加新的变元和规则，使规则转化为合适的形式：

$$S_0 \to ASA|aB|a|SA|AS \qquad S_0 \to AA_1|UB|a|SA|AS$$

$$S \to ASA|aB|a|SA|AS \qquad S \to AA_1|UB|a|SA|AS$$

$$A \to b|ASA|aB|a|SA|AS \qquad A \to b|AA_1|UB|a|SA|AS$$

$$B \to b \qquad B \to b$$

$$U \to a$$

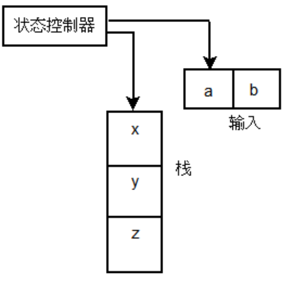
$$A_1 \to SA$$

- 添加新的变元和规则，使规则转化为合适的形式：

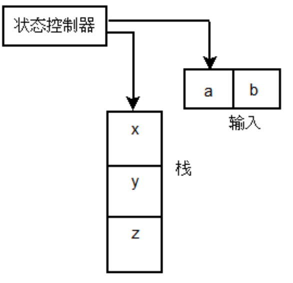
$$
\begin{aligned}
S_0 &\to ASA|aB|a|SA|AS \\
S &\to ASA|aB|a|SA|AS \\
A &\to b|ASA|aB|a|SA|AS \\
B &\to b
\end{aligned}
\qquad
\begin{aligned}
S_0 &\to AA_1|UB|a|SA|AS \\
S &\to AA_1|UB|a|SA|AS \\
A &\to b|AA_1|UB|a|SA|AS \\
B &\to b \\
U &\to a \\
A_1 &\to SA
\end{aligned}
$$

定理2（设G是乔姆斯基范式的CFG，则对于任一长度为$n \geq 1$的字符串$w \in L(G)$，通过CFG G将其派生出来恰好需要$2n-1$步。）

下推自动机(PDA)有额外的可以保存无限信息量的存储设备：栈



下推自动机(PDA)有额外的可以保存无限信息量的存储设备：栈



定义4 ((非确定型)PDA:$(Q, \sum, \Gamma, \delta, q_0, F)$)

- Q是(有限)状态集;
- $\sum$是(有限)输入字母表;
- Γ是(有限)栈字母表;
- $\delta : Q \times \sum_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$,其中

$$\sum_\varepsilon = \sum \cup \{\varepsilon\}, \Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$$

- q_0是起始状态;
- $F \subseteq Q$是(有限)接受状态集;

接受$w = w_1 w_2 \cdots w_m, w_i \in \sum_\epsilon$是指：存在

接受$w = w_1 w_2 \cdots w_m, w_i \in \sum_\epsilon$是指：存在

- 状态序列$r_0, r_1, \cdots, r_m \in Q$

接受$w = w_1 w_2 \cdots w_m, w_i \in \sum_\epsilon$是指：存在

- 状态序列$r_0, r_1, \cdots, r_m \in Q$
- 栈内容序列$s_0, s_1, \cdots, s_m \in \Gamma^*$

满足

接受$w = w_1 w_2 \cdots w_m, w_i \in \sum_\epsilon$是指：存在

- 状态序列$r_0, r_1, \cdots, r_m \in Q$
- 栈内容序列$s_0, s_1, \cdots, s_m \in \Gamma^*$

满足

1. $r_0 = q_0, s_0 = \epsilon$

接受$w = w_1 w_2 \cdots w_m, w_i \in \sum_\epsilon$是指：存在

- 状态序列$r_0, r_1, \cdots, r_m \in Q$
- 栈内容序列$s_0, s_1, \cdots, s_m \in \Gamma^*$

满足

1. $r_0 = q_0, s_0 = \epsilon$
2. $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a), i = 0, 1, \cdots, m-1$，其中$s_i = at, s_{i+1} = bt$

接受$w = w_1 w_2 \cdots w_m, w_i \in \sum_\epsilon$是指：存在

- 状态序列$r_0, r_1, \cdots, r_m \in Q$
- 栈内容序列$s_0, s_1, \cdots, s_m \in \Gamma^*$

满足

1. $r_0 = q_0, s_0 = \epsilon$
2. $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a), i = 0, 1, \cdots, m-1$，其中$s_i = at, s_{i+1} = bt$
3. $r_m \in F$

下推自动机(Pushdown automata PDA)

$$\delta(r_i, w_{i+1}, a) = \{(r_{i+1}, b)\}:$$

$\delta(r_i, w_{i+1}, a) = \{(r_{i+1}, b)\}$：表示当PDA当前状态为$r_i$、输入符号为$w_{i+1}$且栈顶为$a$时

$\delta(r_i, w_{i+1}, a) = \{(r_{i+1}, b)\}$：表示当PDA当前状态为$r_i$、输入符号为$w_{i+1}$且栈顶为$a$时

- PDA由状态r_i进入状态r_{i+1}；

$\delta(r_i, w_{i+1}, a) = \{(r_{i+1}, b)\}$：表示当PDA当前状态为$r_i$、输入符号为$w_{i+1}$且栈顶为$a$时

- PDA由状态r_i进入状态r_{i+1}；
- 栈顶的a被b代替

$\delta(r_i, w_{i+1}, a) = \{(r_{i+1}, b)\}$：表示当PDA当前状态为$r_i$、输入符号为$w_{i+1}$且栈顶为$a$时

- PDA由状态r_i进入状态r_{i+1}；
- 栈顶的a被b代替

1. $\delta(r_i, \varepsilon, a)$：

$\delta(r_i, w_{i+1}, a) = \{(r_{i+1}, b)\}$：表示当PDA当前状态为$r_i$、输入符号为$w_{i+1}$且栈顶为$a$时

- PDA由状态r_i进入状态r_{i+1}；
- 栈顶的a被b代替

1. $\delta(r_i, \varepsilon, a)$：不读入输入符号

$\delta(r_i, w_{i+1}, a) = \{(r_{i+1}, b)\}$：表示当PDA当前状态为$r_i$、输入符号为$w_{i+1}$且栈顶为$a$时

- PDA由状态r_i进入状态r_{i+1}；
- 栈顶的a被b代替

1. $\delta(r_i, \varepsilon, a)$：不读入输入符号
2. $\delta(r_i, w_{i+1}, \varepsilon)$：

$\delta(r_i, w_{i+1}, a) = \{(r_{i+1}, b)\}$：表示当PDA当前状态为$r_i$、输入符号为$w_{i+1}$且栈顶为$a$时

- PDA由状态r_i进入状态r_{i+1}；
- 栈顶的a被b代替

1. $\delta(r_i, \varepsilon, a)$：不读入输入符号
2. $\delta(r_i, w_{i+1}, \varepsilon)$：不读入栈顶元素，也不弹出符号

$\delta(r_i, w_{i+1}, a) = \{(r_{i+1}, b)\}$：表示当PDA当前状态为$r_i$、输入符号为$w_{i+1}$且栈顶为$a$时

- PDA由状态r_i进入状态r_{i+1}；
- 栈顶的a被b代替

1. $\delta(r_i, \varepsilon, a)$：不读入输入符号
2. $\delta(r_i, w_{i+1}, \varepsilon)$：不读入栈顶元素，也不弹出符号
3. $\delta(r_i, w_{i+1}, a) = \{(r_{i+1}, \varepsilon)\}$：

$\delta(r_i, w_{i+1}, a) = \{(r_{i+1}, b)\}$：表示当PDA当前状态为$r_i$、输入符号为$w_{i+1}$且栈顶为$a$时

- PDA由状态r_i进入状态r_{i+1}；
- 栈顶的a被b代替

1. $\delta(r_i, \varepsilon, a)$：不读入输入符号
2. $\delta(r_i, w_{i+1}, \varepsilon)$：不读入栈顶元素，也不弹出符号
3. $\delta(r_i, w_{i+1}, a) = \{(r_{i+1}, \varepsilon)\}$：弹出栈顶的$a$

例8 (识别语言$\{0^n1^n|n \geq 0\}$的PDA)

基本思想:

例8 (识别语言$\{0^n1^n|n \geq 0\}$的PDA)

基本思想:

- 读取输入串中的符号

例8 (识别语言$\{0^n1^n | n \geq 0\}$的PDA)

基本思想:

- 读取输入串中的符号
  - 当符号为0时，将其推入栈;

例8 (识别语言$\{0^n1^n|n \geq 0\}$的PDA)

基本思想:

- 读取输入串中的符号
  - 当符号为0时，将其推入栈;
  - 当符号为1时，将栈顶的0弹出;

例8 (识别语言$\{0^n1^n|n \geq 0\}$的PDA)

基本思想:

- 读取输入串中的符号
  - 当符号为0时，将其推入栈;
  - 当符号为1时，将栈顶的0弹出;
- 接受：当栈中的0被清空时恰好读完输入串，则接受该输入;

下推自动机举例

$$Q = \{q_1, q_2, q_3, q_4\} \qquad \sum = \{0, 1\}$$

$$\Gamma = \{0, \$\} \qquad F = \{q_1, q_4\}$$

$$Q = \{q_1, q_2, q_3, q_4\} \qquad \sum = \{0, 1\}$$

$$\Gamma = \{0, \$\} \qquad F = \{q_1, q_4\}$$

- q_1：初始状态，其中\$用于标识空栈；

$$Q = \{q_1, q_2, q_3, q_4\} \qquad \sum = \{0, 1\}$$

$$\Gamma = \{0, \$\} \qquad F = \{q_1, q_4\}$$

- q_1：初始状态，其中\$用于标识空栈；
- q_2：读入0时，将0推入栈；

$$Q = \{q_1, q_2, q_3, q_4\} \qquad \sum = \{0, 1\}$$

$$\Gamma = \{0, \$\} \qquad F = \{q_1, q_4\}$$

- q_1：初始状态，其中$用于标识空栈；
- q_2：读入0时，将0推入栈；
- q_3：读入1时，将栈顶0弹出；

$$Q = \{q_1, q_2, q_3, q_4\} \qquad \sum = \{0, 1\}$$

$$\Gamma = \{0, \$\} \qquad F = \{q_1, q_4\}$$

- q_1：初始状态，其中\$用于标识空栈；
- q_2：读入0时，将0推入栈；
- q_3：读入1时，将栈顶0弹出；
- q_4：接受状态，即空栈且输入字符串完毕；

转移函数δ:

转移函数δ:

- 空栈标识:

转移函数δ:

- 空栈标识：$\delta(q_1, \varepsilon, \varepsilon) = \{(q_2, \$)\}$

转移函数δ:

- 空栈标识：$\delta(q_1, \varepsilon, \varepsilon) = \{(q_2, \$)\}$
- 读入0：$\delta(q_2, 0, \varepsilon) = \{(q_2, 0)\}$

转移函数δ:

- 空栈标识：$\delta(q_1, \varepsilon, \varepsilon) = \{(q_2, \$)\}$
- 读入0：$\delta(q_2, 0, \varepsilon) = \{(q_2, 0)\}$
- 读入1：

转移函数δ:

- 空栈标识：$\delta(q_1, \varepsilon, \varepsilon) = \{(q_2, \$)\}$
- 读入0：$\delta(q_2, 0, \varepsilon) = \{(q_2, 0)\}$
- 读入1：
  - $\delta(q_2, 1, 0) = \{(q_3, \varepsilon)\}$

转移函数δ:

- 空栈标识：$\delta(q_1, \varepsilon, \varepsilon) = \{(q_2, \$)\}$
- 读入0：$\delta(q_2, 0, \varepsilon) = \{(q_2, 0)\}$
- 读入1：
  - $\delta(q_2, 1, 0) = \{(q_3, \varepsilon)\}$
  - $\delta(q_3, 1, 0) = \{(q_3, \varepsilon)\}$

转移函数δ:

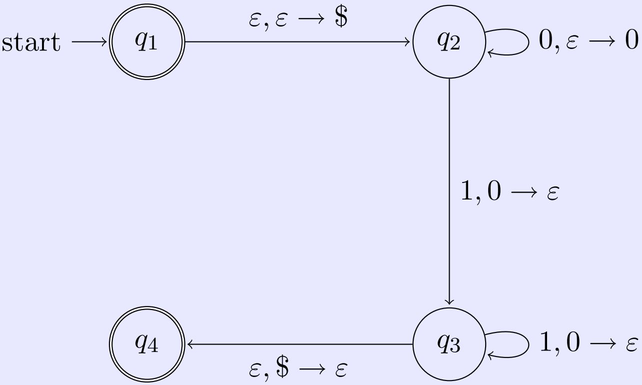
- 空栈标识：$\delta(q_1, \varepsilon, \varepsilon) = \{(q_2, \$)\}$
- 读入0：$\delta(q_2, 0, \varepsilon) = \{(q_2, 0)\}$
- 读入1：
  - $\delta(q_2, 1, 0) = \{(q_3, \varepsilon)\}$
  - $\delta(q_3, 1, 0) = \{(q_3, \varepsilon)\}$
- 接受：$\delta(q_3, \varepsilon, \$) = \{(q_4, \varepsilon)\}$

转移函数δ:

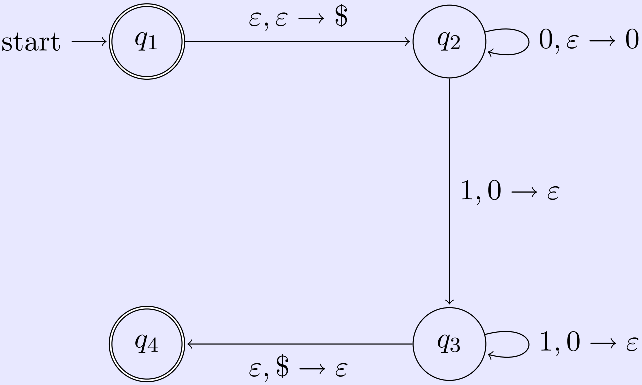
- 空栈标识：$\delta(q_1, \varepsilon, \varepsilon) = \{(q_2, \$)\}$
- 读入0：$\delta(q_2, 0, \varepsilon) = \{(q_2, 0)\}$
- 读入1：
  - $\delta(q_2, 1, 0) = \{(q_3, \varepsilon)\}$
  - $\delta(q_3, 1, 0) = \{(q_3, \varepsilon)\}$
- 接受：$\delta(q_3, \varepsilon, \$) = \{(q_4, \varepsilon)\}$

$a, b \to c$：当读入a时，用c替换栈顶的b

下推自动机举例


start
q_1
$\varepsilon, \varepsilon \to \$$
q_2
$0, \varepsilon \to 0$
$1, 0 \to \varepsilon$
q_3
$1, 0 \to \varepsilon$
q_4
$\varepsilon, \$ \to \varepsilon$

下推自动机举例



考虑输入为0011, 0110的情况

- 对于输入0011：

- 对于输入0110：

- 对于输入0011：

$$(q_2, 0011, \$)$$

- 对于输入0110：

- 对于输入0011：

$$(q_2, 0011, \$) \vdash (q_2, 011, 0\$)$$

- 对于输入0110：

- 对于输入0011：

$$(q_2, 0011, \$) \vdash (q_2, 011, 0\$) \vdash (q_2, 11, 00\$)$$

- 对于输入0110：

- 对于输入0011：

$$(q_2, 0011, \$) \vdash (q_2, 011, 0\$) \vdash (q_2, 11, 00\$)$$

$$\vdash (q_3, 1, 0\$)$$

- 对于输入0110：

- 对于输入0011：

$$(q_2, 0011, \$) \vdash (q_2, 011, 0\$) \vdash (q_2, 11, 00\$)$$

$$\vdash (q_3, 1, 0\$) \vdash (q_3, \varepsilon, \$)$$

- 对于输入0110：

- 对于输入0011：

$$(q_2, 0011, \$) \vdash (q_2, 011, 0\$) \vdash (q_2, 11, 00\$)$$

$$\vdash (q_3, 1, 0\$) \vdash (q_3, \varepsilon, \$)$$

- 对于输入0110：

$$(q_2, 0110, \$)$$

- 对于输入0011:

$$(q_2, 0011, \$) \vdash (q_2, 011, 0\$) \vdash (q_2, 11, 00\$)$$

$$\vdash (q_3, 1, 0\$) \vdash (q_3, \varepsilon, \$)$$

- 对于输入0110:

$$(q_2, 0110, \$) \vdash (q_2, 110, 0\$)$$

- 对于输入0011：

$$(q_2, 0011, \$) \vdash (q_2, 011, 0\$) \vdash (q_2, 11, 00\$)$$

$$\vdash (q_3, 1, 0\$) \vdash (q_3, \varepsilon, \$)$$

- 对于输入0110：

$$(q_2, 0110, \$) \vdash (q_2, 110, 0\$) \vdash (q_3, 10, \$)$$

例9 ($\{ww^R | w \in \{0,1\}^*\}$)

例9 ($\{ww^R | w \in \{0,1\}^*\}$)

- *CFG:*
- *PDA:*

例9 ($\{ww^R|w \in \{0,1\}^*\}$)

- *CFG:*
  - $P \to \varepsilon$
- *PDA:*

例9 ($\{ww^R | w \in \{0,1\}^*\}$)

- *CFG:*
  - $P \to \varepsilon$
  - $P \to 0P0 | 1P1$
- *PDA:*

例9 ($\{ww^R|w \in \{0,1\}^*\}$)

- *CFG:*
  - $P \to \varepsilon$
  - $P \to 0P0|1P1$
- *PDA:*
  - 空栈标识:

例9 ($\{ww^R | w \in \{0,1\}^*\}$)

- *CFG:*
  - $P \rightarrow \varepsilon$
  - $P \rightarrow 0P0|1P1$
- *PDA:*
  - 空栈标识: $\delta(q_1, \varepsilon, \varepsilon) = \{(q_2, \$)\}$

- 在状态q_2时：

- 在状态q_2时：
  - 读入输入符号并将其压入栈

- 在状态q_2时：
  - 读入输入符号并将其压入栈
  - 同时猜测已经达到w的末位，进入状态q_3

- 在状态q_2时：
  - 读入输入符号并将其压入栈
  - 同时猜测已经达到w的末位，进入状态q_3
- 在状态q_3时：

- 在状态q_2时：
  - 读入输入符号并将其压入栈
  - 同时猜测已经达到w的末位，进入状态q_3
- 在状态q_3时：
  - 将输入符号与栈顶符号进行比较

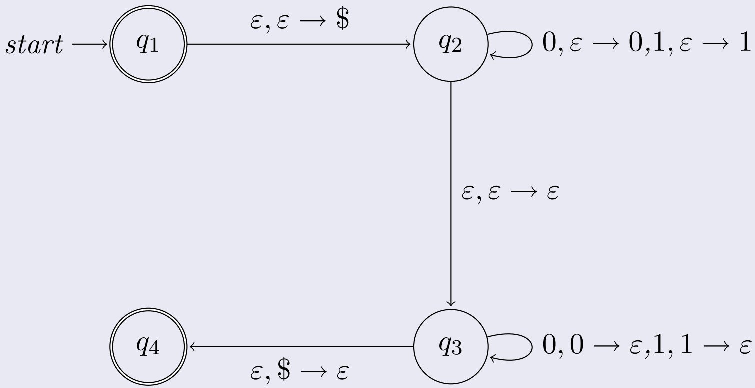
- 在状态q_2时：
  - 读入输入符号并将其压入栈
  - 同时猜测已经达到w的末位，进入状态q_3
- 在状态q_3时：
  - 将输入符号与栈顶符号进行比较
    - 如果相同，则弹出栈顶符号并继续比较

- 在状态q_2时：
  - 读入输入符号并将其压入栈
  - 同时猜测已经达到w的末位，进入状态q_3
- 在状态q_3时：
  - 将输入符号与栈顶符号进行比较
    - 如果相同，则弹出栈顶符号并继续比较
    - 如果不同，则杀死该分支

- 在状态q_2时：
  - 读入输入符号并将其压入栈
  - 同时猜测已经达到w的末位，进入状态q_3
- 在状态q_3时：
  - 将输入符号与栈顶符号进行比较
    - 如果相同，则弹出栈顶符号并继续比较
    - 如果不同，则杀死该分支
- 如果堆栈为空且输入结束，则接受；

例10 $(\{ww^{\mathcal{R}} | w \in \{0,1\}^*\}$，$w^{\mathcal{R}}$表示倒写的$w)$

例10 ($\{ww^{\mathcal{R}} | w \in \{0,1\}^*\}$, $w^{\mathcal{R}}$表示倒写的w)



定义5 (瞬时描述)

定义5 (瞬时描述)

下推自动机M的瞬时描述是：(q, w, γ)，其中

定义5 (瞬时描述)

下推自动机M的瞬时描述是：(q, w, γ)，其中

- q：当前状态；

定义5 (瞬时描述)

下推自动机M的瞬时描述是：(q, w, γ)，其中

- q：当前状态；
- w：剩余的输入串；

定义5 (瞬时描述)

下推自动机M的瞬时描述是：(q, w, γ)，其中

- q：当前状态；
- w：剩余的输入串；
- γ：当前堆栈的内容；

定义5 (瞬时描述)

下推自动机M的瞬时描述是：(q, w, γ)，其中

- q：当前状态;
- w：剩余的输入串;
- γ：当前堆栈的内容;

$\vdash$:

定义5 (瞬时描述)

下推自动机M的瞬时描述是：(q, w, γ)，其中

- q：当前状态；
- w：剩余的输入串；
- γ：当前堆栈的内容；

$\vdash$：设$(p, \alpha) \in \delta(q, a, X)$，则$\forall w \in \sum^*, \beta \in \Gamma^*$，有$(q, aw, X\beta) \vdash (p, w, \alpha\beta)$

定义5（瞬时描述）

下推自动机M的瞬时描述是：(q, w, γ)，其中

- q：当前状态；
- w：剩余的输入串；
- γ：当前堆栈的内容；

$\vdash$：设$(p, \alpha) \in \delta(q, a, X)$，则$\forall w \in \sum^*, \beta \in \Gamma^*$，有$(q, aw, X\beta) \vdash (p, w, \alpha\beta)$

$\vdash^*$：表示任意多次（包括零次）转换。

例11 (输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作)

例11 (输入为1111时$\{ww^{\mathcal{R}} | w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$

例11（输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作）

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$

例11 (输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$ $(q_3, 1111, \$)$

例11 (输入为1111时$\{ww^R|w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$ $(q_3, 1111, \$)$
- $(q_2, 11, 11\$)$

例11 (输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$　$(q_3, 1111, \$)$
- $(q_2, 11, 11\$)$　$(q_3, 111, 1\$)$

例11 (输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$ $(q_3, 1111, \$)$
- $(q_2, 11, 11\$)$ $(q_3, 111, 1\$)$ $(q_3, 11, \$)$

例11 (输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$ $(q_3, 1111, \$)$
- $(q_2, 11, 11\$)$ $(q_3, 111, 1\$)$ $(q_3, 11, \$)$
- $(q_2, 1, 111\$)$

例11 (输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$ $(q_3, 1111, \$)$
- $(q_2, 11, 11\$)$ $(q_3, 111, 1\$)$ $(q_3, 11, \$)$
- $(q_2, 1, 111\$)$ $(q_3, 11, 11\$)$

例11 (输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$ $(q_3, 1111, \$)$
- $(q_2, 11, 11\$)$ $(q_3, 111, 1\$)$ $(q_3, 11, \$)$
- $(q_2, 1, 111\$)$ $(q_3, 11, 11\$)$
- $(q_2, \varepsilon, 1111\$)$

例11 (输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$ $(q_3, 1111, \$)$
- $(q_2, 11, 11\$)$ $(q_3, 111, 1\$)$ $(q_3, 11, \$)$
- $(q_2, 1, 111\$)$ $(q_3, 11, 11\$)$
- $(q_2, \varepsilon, 1111\$)$ $(q_3, 1, 111\$)$

例11 (输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$ $(q_3, 1111, \$)$
- $(q_2, 11, 11\$)$ $(q_3, 111, 1\$)$ $(q_3, 11, \$)$
- $(q_2, 1, 111\$)$ $(q_3, 11, 11\$)$
- $(q_2, \varepsilon, 1111\$)$ $(q_3, 1, 111\$)$ $(q_3, 1, 1\$)$

例11 (输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$ $(q_3, 1111, \$)$
- $(q_2, 11, 11\$)$ $(q_3, 111, 1\$)$ $(q_3, 11, \$)$
- $(q_2, 1, 111\$)$ $(q_3, 11, 11\$)$
- $(q_2, \varepsilon, 1111\$)$ $(q_3, 1, 111\$)$ $(q_3, 1, 1\$)$
- $(q_3, \varepsilon, 1111\$)$

例11 (输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$ $(q_3, 1111, \$)$
- $(q_2, 11, 11\$)$ $(q_3, 111, 1\$)$ $(q_3, 11, \$)$
- $(q_2, 1, 111\$)$ $(q_3, 11, 11\$)$
- $(q_2, \varepsilon, 1111\$)$ $(q_3, 1, 111\$)$ $(q_3, 1, 1\$)$
- $(q_3, \varepsilon, 1111\$)$ $(q_3, \varepsilon, 11\$)$

例11 (输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$ $(q_3, 1111, \$)$
- $(q_2, 11, 11\$)$ $(q_3, 111, 1\$)$ $(q_3, 11, \$)$
- $(q_2, 1, 111\$)$ $(q_3, 11, 11\$)$
- $(q_2, \varepsilon, 1111\$)$ $(q_3, 1, 111\$)$ $(q_3, 1, 1\$)$
- $(q_3, \varepsilon, 1111\$)$ $(q_3, \varepsilon, 11\$)$ $(q_3, \varepsilon, \$)$

例11 (输入为1111时$\{ww^{\mathcal{R}}|w \in \{0,1\}^*\}$的动作)

- $(q_2, 1111, \$)$
- $(q_2, 111, 1\$)$ $(q_3, 1111, \$)$
- $(q_2, 11, 11\$)$ $(q_3, 111, 1\$)$ $(q_3, 11, \$)$
- $(q_2, 1, 111\$)$ $(q_3, 11, 11\$)$
- $(q_2, \varepsilon, 1111\$)$ $(q_3, 1, 111\$)$ $(q_3, 1, 1\$)$
- $(q_3, \varepsilon, 1111\$)$ $(q_3, \varepsilon, 11\$)$ $(q_3, \varepsilon, \$)$ $(q_4, \varepsilon, \$)$

定理3

定理3

- 若$(q,x,\alpha)\vdash^* (p,y,\beta)$，则$\forall w\in\sum^*,\gamma\in\Gamma^*$，有

$$(q,xw,\alpha\gamma)\vdash^* (p,yw,\beta\gamma).$$

定理3

- 若$(q,x,\alpha)\vdash^* (p,y,\beta)$，则$\forall w\in \sum^*, \gamma\in\Gamma^*$，有
$$(q,xw,\alpha\gamma)\vdash^* (p,yw,\beta\gamma).$$
- 若$(q,xw,\alpha)\vdash^* (p,yw,\beta)$，则$(q,x,\alpha)\vdash^* (p,y,\beta)$。

定义6 (以接受状态接受的语言$L(P)$)

设$P=(Q,\sum,\Gamma,\delta,q_0,\$,F)$是某个$PDA$，则以接受状态接受的语言$L(P)$是

$$\{w|(q_0,w,\$)\vdash^*(q,\varepsilon,\alpha),q\in F\}.$$

定义6 (以接受状态接受的语言$L(P)$)

设$P=(Q,\sum,\Gamma,\delta,q_0,\$,F)$是某个$PDA$，则以接受状态接受的语言$L(P)$是

$$\{w|(q_0,w,\$)\vdash^*(q,\varepsilon,\alpha),q\in F\}.$$

即以起始状态和空栈出发、以w为输入，如果在消耗完w后进入接受状态，则$w\in L(P)$。

定义6 (以接受状态接受的语言$L(P)$)

设$P=(Q,\sum,\Gamma,\delta,q_0,\$,F)$是某个$PDA$，则以接受状态接受的语言$L(P)$是

$$\{w|(q_0,w,\$)\vdash^*(q,\varepsilon,\alpha),q\in F\}.$$

即以起始状态和空栈出发、以w为输入，如果在消耗完w后进入接受状态，则$w\in L(P)$。

注意：与最终堆栈内容无关。

证明前述的PDA对应的语言就是$\{ww^R|w \in \{0,1\}^*\}$.

证明前述的PDA对应的语言就是$\{ww^R | w \in \{0,1\}^*\}$.

- 若$x = ww^R$，要证明该PDA必定接受x:

- PDA接受的字符串必定是ww^R形式:

证明前述的PDA对应的语言就是$\{ww^R | w \in \{0,1\}^*\}$.

- 若$x = ww^R$，要证明该PDA必定接受x:

$$(q_2, ww^R, \$)$$

- PDA接受的字符串必定是ww^R形式：

证明前述的PDA对应的语言就是$\{ww^R|w \in \{0,1\}^*\}$.

- 若$x = ww^R$，要证明该PDA必定接受x:

$$(q_2, ww^R, \$) \vdash^* (q_2, w^R, w^R\$)$$

- PDA接受的字符串必定是ww^R形式:

证明前述的PDA对应的语言就是$\{ww^R|w \in \{0,1\}^*\}$.

- 若$x = ww^R$，要证明该PDA必定接受x:

$$(q_2, ww^R, \$) \vdash^* (q_2, w^R, w^R\$) \vdash (q_3, w^R, w^R\$)$$

- PDA接受的字符串必定是ww^R形式:

证明前述的PDA对应的语言就是$\{ww^R|w \in \{0,1\}^*\}$.

- 若$x = ww^R$，要证明该PDA必定接受x:

$$(q_2, ww^R, \$) \vdash^* (q_2, w^R, w^R\$) \vdash (q_3, w^R, w^R\$)$$

$$\vdash^* (q_3, \varepsilon, \$)$$

- PDA接受的字符串必定是ww^R形式:

证明前述的PDA对应的语言就是$\{ww^R|w \in \{0,1\}^*\}$.

- 若$x = ww^R$，要证明该PDA必定接受x:

$$(q_2, ww^R, \$) \vdash^* (q_2, w^R, w^R\$) \vdash (q_3, w^R, w^R\$)$$

$$\vdash^* (q_3, \varepsilon, \$) \vdash (q_4, \varepsilon, \$)$$

- PDA接受的字符串必定是ww^R形式：

证明前述的PDA对应的语言就是$\{ww^R | w \in \{0,1\}^*\}$.

- 若$x = ww^R$，要证明该PDA必定接受x:

$$(q_2, ww^R, \$) \vdash^* (q_2, w^R, w^R\$) \vdash (q_3, w^R, w^R\$)$$

$$\vdash^* (q_3, \varepsilon, \$) \vdash (q_4, \varepsilon, \$)$$

- PDA接受的字符串必定是ww^R形式：只要证明满足

$$(q_2, x, \$) \vdash^* (q_3, \varepsilon, \$)$$

的x必为ww^R型即可。

定理4 (若$(q_2, x, \alpha) \vdash^* (q_3, \varepsilon, \alpha)$，则$x$必为$ww^R$形式)

证明.

□

定理4 (若$(q_2, x, \alpha) \vdash^* (q_3, \varepsilon, \alpha)$，则$x$必为$ww^R$形式)

证明.

- 若$|x| = 0$，

$\square$

定理4（若$(q_2, x, \alpha) \vdash^* (q_3, \varepsilon, \alpha)$，则$x$必为$ww^R$形式）

证明.

- 若$|x| = 0$，即$x = \varepsilon$，

□

定理4 (若$(q_2, x, \alpha) \vdash^* (q_3, \varepsilon, \alpha)$，则$x$必为$ww^R$形式)

证明.

- 若$|x| = 0$，即$x = \varepsilon$，显然成立。

□

定理4 (若$(q_2, x, \alpha) \vdash^* (q_3, \varepsilon, \alpha)$，则$x$必为$ww^R$形式)

证明.

- 若$|x| = 0$，即$x = \varepsilon$，显然成立。
- 假设当$|x| < n$时，结论成立

□

定理4（若$(q_2, x, \alpha) \vdash^* (q_3, \varepsilon, \alpha)$，则$x$必为$ww^R$形式）

证明.

- 若$|x| = 0$，即$x = \varepsilon$，显然成立。
- 假设当$|x| < n$时，结论成立
- 当$x = a_1 a_2 \cdots a_n, n > 0$时，

定理4 (若$(q_2, x, \alpha) \vdash^* (q_3, \varepsilon, \alpha)$，则$x$必为$ww^R$形式)

证明.

- 若$|x| = 0$，即$x = \varepsilon$，显然成立。
- 假设当$|x| < n$时，结论成立
- 当$x = a_1 a_2 \cdots a_n, n > 0$时，
  - 若$(q_2, x, \alpha) \vdash (q_3, x, \alpha)$，

定理4 (若$(q_2, x, \alpha) \vdash^* (q_3, \varepsilon, \alpha)$，则$x$必为$ww^R$形式)

证明.

- 若$|x| = 0$，即$x = \varepsilon$，显然成立。
- 假设当$|x| < n$时，结论成立
- 当$x = a_1a_2 \cdots a_n, n > 0$时，
  - 若$(q_2, x, \alpha) \vdash (q_3, x, \alpha)$，不可能!

定理4 (若$(q_2, x, \alpha) \vdash^* (q_3, \varepsilon, \alpha)$，则$x$必为$ww^R$形式)

证明.

- 若$|x| = 0$，即$x = \varepsilon$，显然成立。
- 假设当$|x| < n$时，结论成立
- 当$x = a_1a_2 \cdots a_n, n > 0$时，
  - 若$(q_2, x, \alpha) \vdash (q_3, x, \alpha)$，不可能!
  - 若$(q_2, a_1a_2 \cdots a_n, \alpha) \vdash (q_2, a_2 \cdots a_n, a_1\alpha)$，

定理4 (若$(q_2, x, \alpha) \vdash^* (q_3, \varepsilon, \alpha)$，则$x$必为$ww^R$形式)

证明.

- 若$|x| = 0$，即$x = \varepsilon$，显然成立。
- 假设当$|x| < n$时，结论成立
- 当$x = a_1 a_2 \cdots a_n, n > 0$时，
  - 若$(q_2, x, \alpha) \vdash (q_3, x, \alpha)$，不可能!
  - 若$(q_2, a_1 a_2 \cdots a_n, \alpha) \vdash (q_2, a_2 \cdots a_n, a_1\alpha)$，则由于$(q_2, a_2 \cdots a_n, a_1\alpha) \vdash^* (q_3, \varepsilon, \alpha)$，

定理4 (若$(q_2, x, \alpha) \vdash^* (q_3, \varepsilon, \alpha)$，则$x$必为$ww^R$形式)

证明.

- 若$|x| = 0$，即$x = \varepsilon$，显然成立。
- 假设当$|x| < n$时，结论成立
- 当$x = a_1a_2 \cdots a_n, n > 0$时，
  - 若$(q_2, x, \alpha) \vdash (q_3, x, \alpha)$，不可能!
  - 若$(q_2, a_1a_2 \cdots a_n, \alpha) \vdash (q_2, a_2 \cdots a_n, a_1\alpha)$，则由于$(q_2, a_2 \cdots a_n, a_1\alpha) \vdash^* (q_3, \varepsilon, \alpha)$， 因此最后一步移动必有$(q_3, a_n, a_1\alpha) \vdash (q_3, \varepsilon, \alpha)$，

□

定理4（若$(q_2, x, \alpha) \vdash^* (q_3, \varepsilon, \alpha)$，则$x$必为$ww^R$形式）

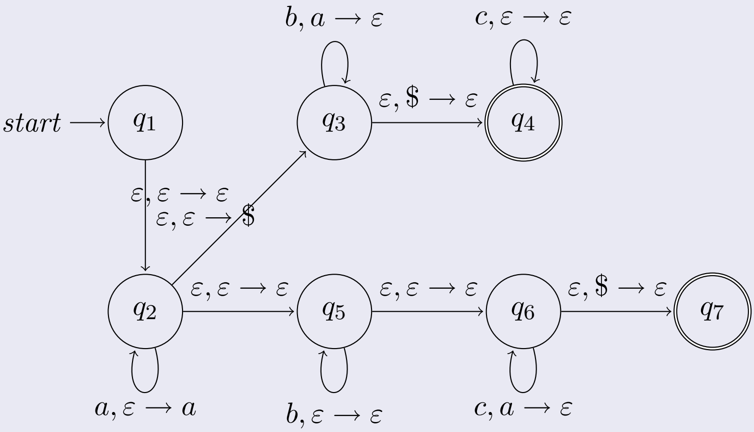
证明.

- 若$|x| = 0$，即$x = \varepsilon$，显然成立。
- 假设当$|x| < n$时，结论成立
- 当$x = a_1 a_2 \cdots a_n, n > 0$时，
  - 若$(q_2, x, \alpha) \vdash (q_3, x, \alpha)$，不可能!
  - 若$(q_2, a_1 a_2 \cdots a_n, \alpha) \vdash (q_2, a_2 \cdots a_n, a_1\alpha)$，则由于$(q_2, a_2 \cdots a_n, a_1\alpha) \vdash^* (q_3, \varepsilon, \alpha)$， 因此最后一步移动必有$(q_3, a_n, a_1\alpha) \vdash (q_3, \varepsilon, \alpha)$，从而$a_1 = a_n$。

□

例12 $(\{a^ib^jc^k|i,j,k \geq 0, i=j$ 或 $i=k\})$

例12 ($\{a^i b^j c^k | i,j,k \geq 0, i=j$ 或 $i=k\}$)



定理5 (语言w是CFL的充要条件是存在识别w的PDA.)

定理5 (语言w是CFL的充要条件是存在识别w的PDA.
正则语言⊂上下文无关语言)

定理5 (语言w是CFL的充要条件是存在识别w的PDA. 正则语言$\subset$上下文无关语言)

1.识别CFL的PDA的构造:

定理5 (语言w是CFL的充要条件是存在识别w的PDA. 正则语言$\subset$上下文无关语言)

1.识别CFL的PDA的构造：

1. 将标记符\$和起始变元推入栈中；

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

定理5 (语言w是CFL的充要条件是存在识别w的PDA. 正则语言⊂上下文无关语言)

1.识别CFL的PDA的构造：

1 将标记符\$和起始变元推入栈中；

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

1 设置新状态q_1

定理5 (语言w是CFL的充要条件是存在识别w的PDA. 正则语言⊂上下文无关语言)

1.识别CFL的PDA的构造：

1. 将标记符$和起始变元推入栈中；

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

1. 设置新状态q_1
2. 设置新转移函数

定理5 (语言w是CFL的充要条件是存在识别w的PDA. 正则语言⊂上下文无关语言)

1.识别CFL的PDA的构造：

1. 将标记符$和起始变元推入栈中；

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

1. 设置新状态q_1
2. 设置新转移函数

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

定理5 (语言w是CFL的充要条件是存在识别w的PDA. 正则语言⊂上下文无关语言)

1.识别CFL的PDA的构造:

1. 将标记符\$和起始变元推入栈中;

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

1. 设置新状态q_1
2. 设置新转移函数

$$\delta(q_{start},\varepsilon,\varepsilon)=\{(q_1,\$)\}$$

定理5 (语言w是CFL的充要条件是存在识别w的PDA. 正则语言⊂上下文无关语言)

1.识别CFL的PDA的构造：

1. 将标记符$和起始变元推入栈中；

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

1. 设置新状态q_1
2. 设置新转移函数

$$\delta(q_{start}, \varepsilon, \varepsilon) = \{(q_1, \$)\}$$

$$\delta(q_1, \varepsilon, \varepsilon) =$$

定理5 (语言w是CFL的充要条件是存在识别w的PDA. 正则语言⊂上下文无关语言)

1.识别CFL的PDA的构造:

1. 将标记符$和起始变元推入栈中;

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

1. 设置新状态q_1
2. 设置新转移函数

$$\delta(q_{start},\varepsilon,\varepsilon)=\{(q_1,\$)\}$$

$$\delta(q_1,\varepsilon,\varepsilon)=\{(q_{loop},S)\}$$

2 重复以下操作：

2 重复以下操作：
   1. 若栈顶为变元A，则非确定地选取一个关于A的规则，并将A替换为该规则右侧的字符串；

2 重复以下操作：

1. 若栈顶为变元A，则非确定地选取一个关于A的规则，并将A替换为该规则右侧的字符串；

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

2 重复以下操作：

1. 若栈顶为变元A，则非确定地选取一个关于A的规则，并将A替换为该规则右侧的字符串；

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

2. 若栈顶为终结符a，则读取输入中的下一符号并将其与a比较。若其匹配，则重复以上操作；否则拒绝；

2 重复以下操作：

1. 若栈顶为变元A，则非确定地选取一个关于A的规则，并将A替换为该规则右侧的字符串；

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

2. 若栈顶为终结符a，则读取输入中的下一符号并将其与a比较。若其匹配，则重复以上操作；否则拒绝；

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

2 重复以下操作：

1. 若栈顶为变元A，则非确定地选取一个关于A的规则，并将A替换为该规则右侧的字符串；

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

2. 若栈顶为终结符a，则读取输入中的下一符号并将其与a比较。若其匹配，则重复以上操作；否则拒绝；

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

3. 若栈顶是\$，则进入接受状态，如果此时输入已经读完，则接受该输入串。

2 重复以下操作：

1. 若栈顶为变元A，则非确定地选取一个关于A的规则，并将A替换为该规则右侧的字符串；

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

2. 若栈顶为终结符a，则读取输入中的下一符号并将其与a比较。若其匹配，则重复以上操作；否则拒绝；

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

3. 若栈顶是\$，则进入接受状态，如果此时输入已经读完，则接受该输入串。

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

例13

将如下的CFG G转化为PDA：$S \to aTb \mid b, \quad T \to Ta \mid \varepsilon$

例13

将如下的$CFG\ G$转化为PDA：$S \to aTb \mid b, \quad T \to Ta \mid \varepsilon$

- 将起始变元和空栈标识压入栈中：

例13

将如下的CFG G转化为PDA：$S \to aTb \mid b, \quad T \to Ta \mid \varepsilon$

- 将起始变元和空栈标识压入栈中：

$$\delta(q_{start}, \varepsilon, \varepsilon) = \{(q_1, \$)\}$$

例13

将如下的CFG G转化为PDA：$S \to aTb \mid b, \quad T \to Ta \mid \varepsilon$

- 将起始变元和空栈标识压入栈中：

$$\delta(q_{start}, \varepsilon, \varepsilon) = \{(q_1, \$)\} \quad \delta(q_1, \varepsilon, \varepsilon) = \{(q_{loop}, S)\}$$

例13

将如下的CFG G转化为PDA：$S \rightarrow aTb \mid b, \quad T \rightarrow Ta \mid \varepsilon$

- 将起始变元和空栈标识压入栈中：

$$\delta(q_{start}, \varepsilon, \varepsilon) = \{(q_1, \$)\} \quad \delta(q_1, \varepsilon, \varepsilon) = \{(q_{loop}, S)\}$$

- 当栈顶为变元T时：

例13

将如下的CFG G转化为PDA：$S \to aTb \mid b, \quad T \to Ta \mid \varepsilon$

- 将起始变元和空栈标识压入栈中：

$$\delta(q_{start}, \varepsilon, \varepsilon) = \{(q_1, \$)\} \quad \delta(q_1, \varepsilon, \varepsilon) = \{(q_{loop}, S)\}$$

- 当栈顶为变元T时：

$$\delta(q_{loop}, \varepsilon, T) = \{(q_2, a)\}$$

例13

将如下的CFG G转化为PDA：$S \to aTb \mid b, \quad T \to Ta \mid \varepsilon$

- 将起始变元和空栈标识压入栈中：

$$\delta(q_{start}, \varepsilon, \varepsilon) = \{(q_1, \$)\} \quad \delta(q_1, \varepsilon, \varepsilon) = \{(q_{loop}, S)\}$$

- 当栈顶为变元T时：

$$\delta(q_{loop}, \varepsilon, T) = \{(q_2, a)\} \quad \delta(q_2, \varepsilon, \varepsilon) = \{(q_{loop}, T)\}$$

例13

将如下的CFG G转化为PDA：$S \rightarrow aTb \mid b, \quad T \rightarrow Ta \mid \varepsilon$

- 将起始变元和空栈标识压入栈中：

$$\delta(q_{start}, \varepsilon, \varepsilon) = \{(q_1, \$)\} \quad \delta(q_1, \varepsilon, \varepsilon) = \{(q_{loop}, S)\}$$

- 当栈顶为变元T时：

$$\delta(q_{loop}, \varepsilon, T) = \{(q_2, a)\} \quad \delta(q_2, \varepsilon, \varepsilon) = \{(q_{loop}, T)\}$$

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

- 当栈顶为变元S时：

- 当栈顶为变元S时：

$$\delta(q_{loop}, \varepsilon, S) = (q_3, b)$$

- 当栈顶为变元S时：

$$\delta(q_{loop}, \varepsilon, S) = (q_3, b) \quad \delta(q_3, \varepsilon, \varepsilon) = \{(q_4, T)\}$$

- 当栈顶为变元S时：

$$\delta(q_{loop},\varepsilon,S)=(q_3,b)\quad\delta(q_3,\varepsilon,\varepsilon)=\{(q_4,T)\}$$

$$\delta(q_4,\varepsilon,\varepsilon)=\{(q_{loop},a)\}$$

- 当栈顶为变元S时：

$$\delta(q_{loop}, \varepsilon, S) = (q_3, b) \quad \delta(q_3, \varepsilon, \varepsilon) = \{(q_4, T)\}$$

$$\delta(q_4, \varepsilon, \varepsilon) = \{(q_{loop}, a)\}\delta(q_{loop}, \varepsilon, S) = \{(q_{loop}, b)\}$$

- 当栈顶为变元S时：

$$\delta(q_{loop},\varepsilon,S)=(q_3,b)\quad\delta(q_3,\varepsilon,\varepsilon)=\{(q_4,T)\}$$

$$\delta(q_4,\varepsilon,\varepsilon)=\{(q_{loop},a)\}\delta(q_{loop},\varepsilon,S)=\{(q_{loop},b)\}$$

- 当栈顶是终止符时

- 当栈顶为变元S时：

$$\delta(q_{loop}, \varepsilon, S) = (q_3, b) \quad \delta(q_3, \varepsilon, \varepsilon) = \{(q_4, T)\}$$

$$\delta(q_4, \varepsilon, \varepsilon) = \{(q_{loop}, a)\}\delta(q_{loop}, \varepsilon, S) = \{(q_{loop}, b)\}$$

- 当栈顶是终止符时

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

- 当栈顶为变元S时：

$$\delta(q_{loop}, \varepsilon, S) = (q_3, b) \quad \delta(q_3, \varepsilon, \varepsilon) = \{(q_4, T)\}$$

$$\delta(q_4, \varepsilon, \varepsilon) = \{(q_{loop}, a)\}\delta(q_{loop}, \varepsilon, S) = \{(q_{loop}, b)\}$$

- 当栈顶是终止符时

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

- 当栈顶为变元S时：

$$\delta(q_{loop}, \varepsilon, S) = (q_3, b) \quad \delta(q_3, \varepsilon, \varepsilon) = \{(q_4, T)\}$$

$$\delta(q_4, \varepsilon, \varepsilon) = \{(q_{loop}, a)\}\delta(q_{loop}, \varepsilon, S) = \{(q_{loop}, b)\}$$

- 当栈顶是终止符时

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

- 接受状态

- 当栈顶为变元S时：

$$\delta(q_{loop}, \varepsilon, S) = (q_3, b) \quad \delta(q_3, \varepsilon, \varepsilon) = \{(q_4, T)\}$$

$$\delta(q_4, \varepsilon, \varepsilon) = \{(q_{loop}, a)\}\delta(q_{loop}, \varepsilon, S) = \{(q_{loop}, b)\}$$

- 当栈顶是终止符时

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

- 接受状态

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

考虑输入为$aaab$的情况：

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$)$$

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$) \vdash (q_3, aaab, b\$)$$

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$) \vdash (q_3, aaab, b\$) \vdash (q_4, aaab, Tb\$)$$

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$) \vdash (q_3, aaab, b\$) \vdash (q_4, aaab, Tb\$)$$

$$\vdash (q_{loop}, aaab, aTb\$)$$

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$) \vdash (q_3, aaab, b\$) \vdash (q_4, aaab, Tb\$)$$

$$\vdash (q_{loop}, aaab, aTb\$) \vdash (q_{loop}, aab, Tb\$)$$

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$) \vdash (q_3, aaab, b\$) \vdash (q_4, aaab, Tb\$)$$

$$\vdash (q_{loop}, aaab, aTb\$) \vdash (q_{loop}, aab, Tb\$) \vdash (q_2, aab, ab\$)$$

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$) \vdash (q_3, aaab, b\$) \vdash (q_4, aaab, Tb\$)$$

$$\vdash (q_{loop}, aaab, aTb\$) \vdash (q_{loop}, aab, Tb\$) \vdash (q_2, aab, ab\$)$$

$$\vdash (q_{loop}, aab, Tab\$)$$

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$) \vdash (q_3, aaab, b\$) \vdash (q_4, aaab, Tb\$)$$

$$\vdash (q_{loop}, aaab, aTb\$) \vdash (q_{loop}, aab, Tb\$) \vdash (q_2, aab, ab\$)$$

$$\vdash (q_{loop}, aab, Tab\$) \vdash (q_2, aab, aab\$)$$

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$) \vdash (q_3, aaab, b\$) \vdash (q_4, aaab, Tb\$)$$

$$\vdash (q_{loop}, aaab, aTb\$) \vdash (q_{loop}, aab, Tb\$) \vdash (q_2, aab, ab\$)$$

$$\vdash (q_{loop}, aab, Tab\$) \vdash (q_2, aab, aab\$) \vdash (q_{loop}, aab, Taab\$)$$

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$) \vdash (q_3, aaab, b\$) \vdash (q_4, aaab, Tb\$)$$

$$\vdash (q_{loop}, aaab, aTb\$) \vdash (q_{loop}, aab, Tb\$) \vdash (q_2, aab, ab\$)$$

$$\vdash (q_{loop}, aab, Tab\$) \vdash (q_2, aab, aab\$) \vdash (q_{loop}, aab, Taab\$)$$

$$\vdash (q_{loop}, aab, aab\$)$$

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$) \vdash (q_3, aaab, b\$) \vdash (q_4, aaab, Tb\$)$$

$$\vdash (q_{loop}, aaab, aTb\$) \vdash (q_{loop}, aab, Tb\$) \vdash (q_2, aab, ab\$)$$

$$\vdash (q_{loop}, aab, Tab\$) \vdash (q_2, aab, aab\$) \vdash (q_{loop}, aab, Taab\$)$$

$$\vdash (q_{loop}, aab, aab\$) \vdash (q_{loop}, ab, ab\$)$$

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$) \vdash (q_3, aaab, b\$) \vdash (q_4, aaab, Tb\$)$$

$$\vdash (q_{loop}, aaab, aTb\$) \vdash (q_{loop}, aab, Tb\$) \vdash (q_2, aab, ab\$)$$

$$\vdash (q_{loop}, aab, Tab\$) \vdash (q_2, aab, aab\$) \vdash (q_{loop}, aab, Taab\$)$$

$$\vdash (q_{loop}, aab, aab\$) \vdash (q_{loop}, ab, ab\$) \vdash (q_{loop}, b, b\$)$$

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$) \vdash (q_3, aaab, b\$) \vdash (q_4, aaab, Tb\$)$$

$$\vdash (q_{loop}, aaab, aTb\$) \vdash (q_{loop}, aab, Tb\$) \vdash (q_2, aab, ab\$)$$

$$\vdash (q_{loop}, aab, Tab\$) \vdash (q_2, aab, aab\$) \vdash (q_{loop}, aab, Taab\$)$$

$$\vdash (q_{loop}, aab, aab\$) \vdash (q_{loop}, ab, ab\$) \vdash (q_{loop}, b, b\$)$$

$$\vdash (q_{loop}, \varepsilon, \$)$$

考虑输入为$aaab$的情况：

$$(q_{loop}, aaab, S\$) \vdash (q_3, aaab, b\$) \vdash (q_4, aaab, Tb\$)$$

$$\vdash (q_{loop}, aaab, aTb\$) \vdash (q_{loop}, aab, Tb\$) \vdash (q_2, aab, ab\$)$$

$$\vdash (q_{loop}, aab, Tab\$) \vdash (q_2, aab, aab\$) \vdash (q_{loop}, aab, Taab\$)$$

$$\vdash (q_{loop}, aab, aab\$) \vdash (q_{loop}, ab, ab\$) \vdash (q_{loop}, b, b\$)$$

$$\vdash (q_{loop}, \varepsilon, \$) \vdash (q_{accept}, \$)$$

定理6 (上下文无关语言的泵引理)

定理6 (上下文无关语言的泵引理)

如果A是上下文无关语言，则存在泵长度p，使得A中任何长度不小于p的字符串s都可以被划分为5段：$s = uvxyz$，满足：

定理6 (上下文无关语言的泵引理)

如果A是上下文无关语言，则存在泵长度p，使得A中任何长度不小于p的字符串s都可以被划分为5段：$s = uvxyz$，满足：

1. 对于每个$i \geq 0$，$uv^ixy^iz \in A$；

定理6 (上下文无关语言的泵引理)

如果A是上下文无关语言，则存在泵长度p，使得A中任何长度不小于p的字符串s都可以被划分为5段：$s = uvxyz$，满足：

1. 对于每个$i \geq 0$，$uv^ixy^iz \in A$;
2. $|vy| > 0$;

定理6 (上下文无关语言的泵引理)

如果A是上下文无关语言，则存在泵长度p，使得A中任何长度不小于p的字符串s都可以被划分为5段：$s = uvxyz$，满足：

1. 对于每个$i \geq 0$，$uv^ixy^iz \in A$；
2. $|vy| > 0$；
3. $|vxy| \leq p$。

证明基本思想：

例14 (证明$B = \{a^n b^n c^n | n \geq 0\}$不是上下文无关的)

证明.

$\square$

例14 (证明$B=\{a^nb^nc^n|n\geq 0\}$不是上下文无关的)

证明.

设泵长度为p，考虑字符串$s=a^pb^pc^p$

$\square$

例14 (证明$B = \{a^n b^n c^n | n \geq 0\}$不是上下文无关的)

证明.

设泵长度为p，考虑字符串$s = a^p b^p c^p = uvxyz$，其中v或y不是空串。

例14 (证明$B=\{a^nb^nc^n|n\geq 0\}$不是上下文无关的)

证明.

设泵长度为p，考虑字符串$s=a^pb^pc^p=uvxyz$，其中v或y不是空串。

考虑字符串uv^2xy^2z:

$\square$

例14 (证明$B = \{a^n b^n c^n | n \geq 0\}$不是上下文无关的)

证明.

设泵长度为p，考虑字符串$s = a^p b^p c^p = uvxyz$，其中v或y不是空串。

考虑字符串uv^2xy^2z:

- 若v, y都只包含一种符号，则uv^2xy^2z中a, b, c个数不可能相同，即其不属于B;

例14 (证明$B=\{a^nb^nc^n|n\geq 0\}$不是上下文无关的)

证明.

设泵长度为p，考虑字符串$s=a^pb^pc^p=uvxyz$，其中v或y不是空串。

考虑字符串uv^2xy^2z:

- 若v,y都只包含一种符号，则uv^2xy^2z中a,b,c个数不可能相同，即其不属于B;
- 若v或y中含有2个符号，则uv^2xy^2z中a,b,c的次序不对，即其不属于B;

□

例15 (证明$B=\{a^ib^jc^k|0\le i\le j\le k\}$不是上下文无关的)

证明.

例15 (证明$B = \{a^i b^j c^k | 0 \le i \le j \le k\}$不是上下文无关的)

证明.

考虑字符串uv^2xy^2z，其中$a^p b^p c^p = uvxyz$，v或y不是空串。

例15 (证明$B=\{a^ib^jc^k|0\le i\le j\le k\}$不是上下文无关的)

证明.

考虑字符串uv^2xy^2z，其中$a^pb^pc^p=uvxyz$，v或y不是空串。

- 若v,y都只包含一种符号，
- 若v或y中含有2个符号，则uv^2xy^2z中a,b,c的次序不对，

例15 (证明$B = \{a^i b^j c^k | 0 \le i \le j \le k\}$不是上下文无关的)

证明.

考虑字符串uv^2xy^2z，其中$a^p b^p c^p = uvxyz$，v或y不是空串。

- 若v, y都只包含一种符号，
  - 若$a \notin v, y$，
  - 若$b \notin v, y$，则
  - 若$c \notin v, y$，
- 若v或y中含有2个符号，则uv^2xy^2z中a, b, c的次序不对，

例15 (证明$B=\{a^ib^jc^k|0\le i\le j\le k\}$不是上下文无关的)

证明.

考虑字符串uv^2xy^2z，其中$a^pb^pc^p=uvxyz$，v或y不是空串。

- 若v,y都只包含一种符号，
  - 若$a\notin v,y$，则$uv^0xy^0z=uxz\notin B$;
  - 若$b\notin v,y$，则
  - 若$c\notin v,y$，
- 若v或y中含有2个符号，则uv^2xy^2z中a,b,c的次序不对，

例15 (证明$B=\{a^ib^jc^k|0\le i\le j\le k\}$不是上下文无关的)

证明.

考虑字符串uv^2xy^2z，其中$a^pb^pc^p=uvxyz$，v或y不是空串。

- 若v,y都只包含一种符号，
  - 若$a\notin v,y$，则$uv^0xy^0z=uxz\notin B$;
  - 若$b\notin v,y$，则
    - 若$a\in v$或y，则$uv^2xy^2z\notin B$;
  - 若$c\notin v,y$，
- 若v或y中含有2个符号，则uv^2xy^2z中a,b,c的次序不对，

例15 (证明$B=\{a^ib^jc^k|0\le i\le j\le k\}$不是上下文无关的)

证明.

考虑字符串uv^2xy^2z，其中$a^pb^pc^p=uvxyz$，v或y不是空串。

- 若v,y都只包含一种符号，
  - 若$a\notin v,y$，则$uv^0xy^0z=uxz\notin B$;
  - 若$b\notin v,y$，则
    - 若$a\in v$或y，则$uv^2xy^2z\notin B$;
    - 若$c\in v$或y，则$uv^0xy^0z\notin B$;
  - 若$c\notin v,y$，
- 若v或y中含有2个符号，则uv^2xy^2z中a,b,c的次序不对，

例15 (证明$B=\{a^ib^jc^k|0\le i\le j\le k\}$不是上下文无关的)

证明.

考虑字符串uv^2xy^2z，其中$a^pb^pc^p=uvxyz$，v或y不是空串。

- 若v,y都只包含一种符号，
  - 若$a\notin v,y$，则$uv^0xy^0z=uxz\notin B$;
  - 若$b\notin v,y$，则
    - 若$a\in v$或y，则$uv^2xy^2z\notin B$;
    - 若$c\in v$或y，则$uv^0xy^0z\notin B$;
  - 若$c\notin v,y$，则$uv^2xy^2z\notin B$;
- 若v或y中含有2个符号，则uv^2xy^2z中a,b,c的次序不对，

例16 (证明$B=\{ww|w\in\{0,1\}^*\}$不是CFL(提示：考虑$0^p1^p0^p1^p$))

定理7 (CFG在并运算、连接运算、星运算下封闭)

证明.

定理7 (CFG在并运算、连接运算、星运算下封闭)

证明.

设L_1, L_2分别是由CFG

$$G_1 = (V_1, T_1, P_1, S_1), \quad G_2 = (V_2, T_2, P_2, S_2)$$

产生的语言，不妨设$V_1 \cap V_2 = \emptyset$。

定理7 (CFG在并运算、连接运算、星运算下封闭)

证明.

设L_1, L_2分别是由CFG

$$G_1 = (V_1, T_1, P_1, S_1), \quad G_2 = (V_2, T_2, P_2, S_2)$$

产生的语言，不妨设$V_1 \cap V_2 = \emptyset$。

- 对于$L_1 \cup L_2$，

定理7 (CFG在并运算、连接运算、星运算下封闭)

证明.

设L_1, L_2分别是由CFG

$$G_1 = (V_1, T_1, P_1, S_1), \quad G_2 = (V_2, T_2, P_2, S_2)$$

产生的语言，不妨设$V_1 \cap V_2 = \emptyset$。

- 对于$L_1 \cup L_2$，

$$G_\cup = (V_1 \cup V_2 \cup \{S_3\}, T_1 \cup T_2, P_3, S_3)$$

定理7 (CFG在并运算、连接运算、星运算下封闭)

证明.

设L_1, L_2分别是由CFG

$$G_1 = (V_1, T_1, P_1, S_1), \quad G_2 = (V_2, T_2, P_2, S_2)$$

产生的语言，不妨设$V_1 \cap V_2 = \emptyset$。

- 对于$L_1 \cup L_2$，

$$G_\cup = (V_1 \cup V_2 \cup \{S_3\}, T_1 \cup T_2, P_3, S_3)$$

其中$P_3 = P_1 \cup P_2 \cup \{S_3 \to S_1 | S_2\}$。

封闭性

- 对于L_1L_2:

- 对于L_1L_2:

$$G_| = (V_1 \cup V_2 \cup \{S_3\}, T_1 \cup T_2, P_3, S_3)$$

- 对于L_1L_2：

$$G_| = (V_1 \cup V_2 \cup \{S_3\}, T_1 \cup T_2, P_3, S_3)$$

其中$P_3 = P_1 \cup P_2 \cup \{S_3 \to S_1S_2\}$。

- 对于L_1L_2：

$$G_| = (V_1 \cup V_2 \cup \{S_3\}, T_1 \cup T_2, P_3, S_3)$$

其中$P_3 = P_1 \cup P_2 \cup \{S_3 \to S_1S_2\}$。

- 对于L_1^*：

- 对于L_1L_2：

$$G_| = (V_1 \cup V_2 \cup \{S_3\}, T_1 \cup T_2, P_3, S_3)$$

其中$P_3 = P_1 \cup P_2 \cup \{S_3 \to S_1S_2\}$。

- 对于L_1^*：

$$G_* = (V_1 \cup \{S_3\}, T_1, P_3, S_3)$$

- 对于L_1L_2：

$$G_| = (V_1 \cup V_2 \cup \{S_3\}, T_1 \cup T_2, P_3, S_3)$$

其中$P_3 = P_1 \cup P_2 \cup \{S_3 \to S_1S_2\}$。

- 对于L_1^*：

$$G_* = (V_1 \cup \{S_3\}, T_1, P_3, S_3)$$

其中$P_3 = P_1 \cup \{S_3 \to S_1S_3|\varepsilon\}$。

定理8 (CFG在交运算下不封闭)

证明.

$\square$

定理8 (CFG在交运算下不封闭)

证明.

- $L_1 = \{a^n b^n c^i | n \geq 1, i \geq 1\}, L_2 = \{a^i b^n c^n | n \geq 1, i \geq 1\}$是上下文无关的。

$\square$

定理8 (CFG在交运算下不封闭)

证明.

- $L_1=\{a^nb^nc^i|n\geq 1, i\geq 1\}, L_2=\{a^ib^nc^n|n\geq 1, i\geq 1\}$是上下文无关的。

$$S\rightarrow AB,\ A\rightarrow aAb|ab,\ B\rightarrow cB|c$$

□

定理8 (CFG在交运算下不封闭)

证明.

- $L_1 = \{a^n b^n c^i | n \geq 1, i \geq 1\}, L_2 = \{a^i b^n c^n | n \geq 1, i \geq 1\}$是上下文无关的。

$$S \to AB,\ A \to aAb|ab,\ B \to cB|c$$

- $L = L_1 \cap L_2$

□

定理8 (CFG在交运算下不封闭)

证明.

- $L_1 = \{a^n b^n c^i | n \geq 1, i \geq 1\}, L_2 = \{a^i b^n c^n | n \geq 1, i \geq 1\}$是上下文无关的。

$$S \rightarrow AB,\ A \rightarrow aAb|ab,\ B \rightarrow cB|c$$

- $L = L_1 \cap L_2 = \{a^n b^n c^n | n \geq 1\}$

$\square$

定理8 (CFG在交运算下不封闭)

证明.

- $L_1 = \{a^n b^n c^i | n \geq 1, i \geq 1\}, L_2 = \{a^i b^n c^n | n \geq 1, i \geq 1\}$是上下文无关的。

$$S \rightarrow AB,\ \ A \rightarrow aAb|ab,\ \ B \rightarrow cB|c$$

- $L = L_1 \cap L_2 = \{a^n b^n c^n | n \geq 1\}$不是上下文无关的。

□

定理9 (CFG在“补运算”下不封闭)

定理9 (CFG在“补运算”下不封闭)

定理10 (若L是CFL，R是正则语言，则$L \cap R$是CFL)