问题

化简如下给出的确定型有穷接受器，其中初始状态为A，终止状态为C。

δ	A	B	C	D	E	F	G	H
0	B	G	A	C	H	C	G	G
1	F	C	C	G	F	G	E	C

解答

根据列表，可以得到有穷接受器的状态迁移图，在迁移图中，可以看出D是不可达状态，将不可达状态D去掉。

应用Mark程序，由于C是终止状态，直接可区分的状态对(A,C)，(B,C)，(C,E)，(C,F)，(C,G)，(C,H)；

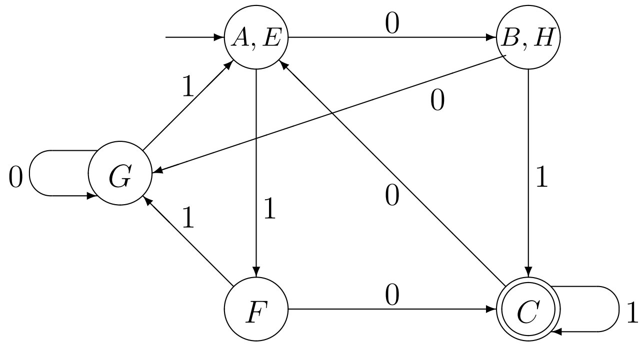
输入符号0，可得可区分状态对(A,F)，(B,F)，(F,G)，(E,F)，(F,H)，(G,H)，(H,E)，(H,A)；

输入符号1，可得可区分状态对(B,A)，(B,G)，(B,E)；

输入符号10，可得可区分状态对(G,A)，(E,G)。

之后，不会有新的可区分状态对了，因此可得不可区分状态对为(E,A)，(B,H)。

合并不可区分状态后，有穷接受器的示意图如下：



问题

设Σ是一个字母表，K是由字母表Σ上的符号串组成的有限集合。令语言

$$L = \{w \in \Sigma^* : \text{集合}K\text{中的符号串不是}w\text{的子串}\},$$

判定语言L是否是正则语言。

解答

假设字母表$\Sigma = \{a_1, a_2, \cdots, a_m\}$，则集合$K$可以表示为$K = \{w_1, w_2, \cdots, w_n\}$，其中$w_i$是字母表$\Sigma$上的符号串。令语言

$$L_1 = \{w \in \Sigma^*\text{，集合}K\text{中某个符号串是}w\text{的子串}\},$$

即$w \in L_1$，则w中包含$w_1, w_2, \cdots, w_n$中的某一个。因此，语言L_1可以用如下正则表达式表示：

$$(a_1 + a_2 + \cdots + a_m)^*(w_1 + w_2 + \cdots + w_n)(a_1 + a_2 + \cdots + a_m)^*,$$

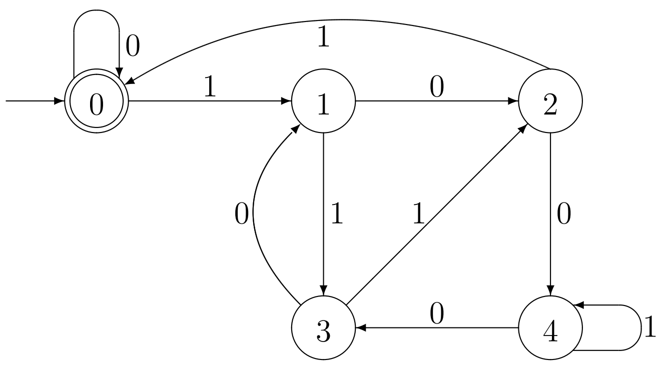
由此可得语言L_1是正则的。而$L = \overline{L_1}$，由正则语言的性质可得语言L是正则语言。

问题

在$\{0,1\}$上构造完成下面功能的有限自动机：把输入的符号串看成二进制数，如果这个二进制数能够被5整除，那么就接受这个符号串。例如0101和11001分别表示5和25，可以被设计的有限自动机接受。然后给出生成该语言的正则文法。

解答

构造有穷接受器，以模5的余数为状态，当前状态为q_i，输入为a，下一时刻状态为$2 * q_i + a$。示意图如下：



由此有穷接受器构造正则文法，可得文法

$$G = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, q_0, P),$$

其中产生式集合为：

$$\begin{array}{lll} q_0 \to 0q_0|1q_1|\varepsilon & q_1 \to 0q_2|1q_3 & q_2 \to 0q_4|1q_0 \\ q_3 \to 0q_1|1q_2 & q_4 \to 0q_3|1q_4 & \end{array}$$

问题

假设x和y是符号串，定义运算

$$S(x,y)=\{w|存在n\geq 1,使得w=x_1y_1x_2y_2\cdots x_ny_n,$$

$$其中x_i和y_i是符号串，并且x=x_1x_2\cdots x_n,y=y_1y_2\cdots y_n\}。$$

对于语言L_1和L_2，定义$S(L_1,L_2)=\{S(x,y):x\in L_1,y\in L_2\}$。证明：如果语言$L_1$和$L_2$是正则语言，则$S(L_1,L_2)$是正则语言。

解答

我们假设L_1由确定型有穷接受器$M_1=(Q_1,\Sigma_1,\delta_1,q_{01},F_1)$接受，$L_2$由确定型有穷接受器$M_2=(Q_2,\Sigma_2,\delta_2,q_{02},F_2)$接受。我们构造有穷接受器

$$M=(Q_1\times Q_2,\Sigma_1\cup\Sigma_2,\delta,(q_{01},q_{02}),F_1\times F_2),$$

其中转移函数δ如下定义：

$$\delta((q_{1i},q_{2j}),x)=\{(\delta_1(q_{1i},x),q_{2j}),(q_{1i},\delta_2(q_{2j},x))\}。$$

下面证明$S(L_1,L_2)=L(M)$。对任意$w\in L(M)$，当M输入w后，状态从(q_{01},q_{02})迁移到(q_{f1},q_{f2})，我们把w中会引发第一个状态分量迁移的符号依次连接在一起为x，把w中会引发第二个状态分量迁移的符号依次连接在一起为y，则有$w=S(x,y)$。我们将x输入到M_1中，状态会由q_{01}迁移到q_{f1}，从而$x\in L(M_1)=L_1$。同样有$y\in L(M_2)=L_2$，从而有$w\in S(L_1,L_2)$。

反之，对任意$w\in S(L_1,L_2)$，则存在$x\in L_1$，$y\in L_2$，使得$w=S(x,y)$。由于$x\in L_1$，我们将x输入到M_1中，状态会由q_{01}迁移到q_{f1}；由于$y\in L_2$，我们将y输入到M_2中，状态会由q_{02}迁移到q_{f2}。如果把w输入到M中，其中属于x的部分会使得状态的第一个分量由q_{01}迁移到q_{f1}，属于y的部分会使得状态的第一个分量由q_{02}迁移到q_{f2}，从而，M的状态从(q_{01},q_{02})迁移到(q_{f1},q_{f2})，即$w\in L(M)$。

问题

构造一个算法，判断正则语言L是否包含无穷多个长度为偶数的符号串。

解答

假设字母表$\Sigma = \{a_1, a_2, \cdots, a_m\}$，并且设语言$L_1 = \{$长度为偶数的符号串$\}$，有

$$L_1 = L(((a_1 + a_2 + \cdots + a_m)(a_1 + a_2 + \cdots + a_m))^*)。$$

由此可得，语言L_1是正则语言。又语言L是正则的，则$L_1 \cap L$是正则语言。

容易得出：语言L包含无穷多个长度为偶数的符号串当且仅当语言$L_1 \cap L$包含无穷多个符号串。语言$L_1 \cap L$是正则语言，存在算法判定语言$L_1 \cap L$是否包含无穷多个符号串。因此，存在算法，判断正则语言L是否包含无穷多个长度为偶数的符号串。

具体算法：首先，构造一个接受正则语言$L_1 \cap L$的确定型有穷接受器，并给出它的状态转移图；然后，找出状态转移图中可以构成某个回路的所有顶点；最后，判断这些顶点中是否有一个顶点位于从初始状态到终止状态的某条路径中。

如果有，则正则语言L包含无穷多个长度为偶数的符号串。

如果没有，则正则语言L仅包含有限多个长度为偶数的符号串。

问题

给定文法$G=(\{S,A,B,C,D,E\},\{a,b,c\},S,P)$，其产生式为

$$\begin{array}{lll} S\rightarrow A|aSaB|BSC, & A\rightarrow a|bS|Ca|B, & E\rightarrow bA|A, \\ B\rightarrow b|\varepsilon, & C\rightarrow BB|aAS, & D\rightarrow A|aD|\varepsilon, \end{array}$$

试消除文法中的ε-产生式、单位产生式和无用产生式。

解答

注意到符号串$\varepsilon\in L(G)$，在去除ε-产生式，新文法将无法由S推导出ε。

1、去除ε-产生式

计算可空变量有$V_N=\{B,D,\quad A,C,\quad S,E\}$。

依据去除ε-产生式的方法，可得去除ε-产生式后文法的产生式为

$$\begin{array}{lcll} S & \rightarrow & A|aSaB|BSC & |aaB|aSa|aa|BS|SC|BC|B|S|C \\ A & \rightarrow & a|bS|Ca|B & |b \\ B & \rightarrow & b & \\ C & \rightarrow & BB|aAS & |B|aA|aS|a \\ D & \rightarrow & A|aD & |a \\ E & \rightarrow & bA|A & |b \end{array}$$

2、去除单位产生式

根据产生式画出单位推导的依赖图，可得单位推导有$S\overset{*}{\Rightarrow}S$，$S\overset{*}{\Rightarrow}A$，$S\overset{*}{\Rightarrow}B$，$S\overset{*}{\Rightarrow}C$，$A\overset{*}{\Rightarrow}B$，$C\overset{*}{\Rightarrow}B$，$D\overset{*}{\Rightarrow}A$，$D\overset{*}{\Rightarrow}B$，$E\overset{*}{\Rightarrow}A$，$E\overset{*}{\Rightarrow}B$。

依据去除单位产生式的方法，可得去除单位产生式后文法的产生式为

$$\begin{array}{lcll} S & \rightarrow & aSaB|BSC|aaB|aSa|aa|BS|SC|BC & |a|bS|Ca|b|BB|aAS|aA|aS \\ A & \rightarrow & a|bS|Ca|b & \\ B & \rightarrow & b & \\ C & \rightarrow & BB|aAS|aA|aS|a & |b \\ D & \rightarrow & aD|a & |bS|Ca|b \\ E & \rightarrow & bA|b & |a|bS|Ca \end{array}$$

3、去除无用产生式

计算可以推导出终结符号串的变量构成的集合：通过产生式集合可得

$$V_1=\{S,A,B,C,D,E\},$$

即所有变量都可以推导出终结符号串。

计算无法从S到达的变量：根据产生式集合可得变量依赖图，通过依赖图可知，变量D和E不可达，是无用变量。

去除包含无用变量的产生式，可得去除无用产生式后文法的产生式为

$$
\begin{aligned}
S &\rightarrow aSaB|BSC|aaB|aSa|aa|BS|SC|BC|a|bS|Ca|b|BB|aAS|aA|aS \\
A &\rightarrow a|bS|Ca|b \\
B &\rightarrow b \\
C &\rightarrow BB|aAS|aA|aS|a|b
\end{aligned}
$$

问题

给定语言

$$L = \{a^n b^n : n \geq 0\},$$

试构造一个上下文无关文法G，使得文法G生成语言$\overline{L}$。

解答

在语言L中的句子，符号a在符号b之前，并且两者数目相同。因此，语言$\overline{L}$中的句子，要么有符号b出现在符号a之前，要么符号a全在符号b之前，但两者数目不同。

基于此，构造文法$G = (\{S, A, B, C, D, E\}, \{a, b\}, S, P)$，其中产生式$P$包括：

$$\begin{array}{rcl} S & \rightarrow & A|B|D \\ A & \rightarrow & aA|aC \\ B & \rightarrow & Bb|Cb \\ C & \rightarrow & aCb|\varepsilon \\ D & \rightarrow & EbEaE \\ E & \rightarrow & aE|bE|\varepsilon \end{array}$$

注：变量D生成有符号b出现在符号a之前的符号串，其中变量E生成$(a+b)^*$；变量A和变量B生成的符号串满足符号a全在符号b之前，但两者数目不同，其中变量A生成符号a多的符号串，其中变量B生成符号b多的符号串。

问题

给定下推自动机

$$M = (\{q_0, q_1, q_2\}, \{a, b\}, \{c, z\}, \delta, q_0, z, \{q_2\}),$$

其中转移函数为

$$\begin{array}{ll} \delta(q_0, b, z) = \{(q_1, ccz)\}, & \delta(q_0, a, c) = \{(q_1, \varepsilon)\}, \\ \delta(q_1, a, c) = \{(q_1, \varepsilon)\}, & \delta(q_1, a, z) = \{(q_1, z), (q_2, \varepsilon)\}, \end{array}$$

试给出下推自动机M对应的上下文无关文法。

解答

先将转移函数改造为满足定理7.2要求的形式，可得转移函数集合：

$$\begin{array}{ll} \delta(q_0, b, z) = \{(q_4, cz)\}, & \delta(q_4, \varepsilon, c) = \{(q_0, cc)\}, \\ \delta(q_0, a, c) = \{(q_1, \varepsilon)\}, & \delta(q_1, a, c) = \{(q_1, \varepsilon)\}, \\ \delta(q_1, a, z) = \{(q_3, cz)\}, & \delta(q_3, \varepsilon, c) = \{(q_1, \varepsilon)\}, \\ \delta(q_1, a, z) = \{(q_2, \varepsilon)\}。 & \end{array}$$

由第一个转移函数，可得产生式集合：

$$\begin{array}{lcl} q_0zq_0 & \to & b(q_4cq_0)(q_0zq_0)|b(q_4cq_1)(q_1zq_0)|b(q_4cq_2)(q_2zq_0)| \\ & & b(q_4cq_3)(q_3zq_0)|b(q_4cq_4)(q_4zq_0), \\ q_0zq_1 & \to & b(q_4cq_0)(q_0zq_1)|b(q_4cq_1)(q_1zq_1)|b(q_4cq_2)(q_2zq_1)| \\ & & b(q_4cq_3)(q_3zq_1)|b(q_4cq_4)(q_4zq_1), \\ q_0zq_2 & \to & b(q_4cq_0)(q_0zq_2)|b(q_4cq_1)(q_1zq_2)|b(q_4cq_2)(q_2zq_2)| \\ & & b(q_4cq_3)(q_3zq_2)|b(q_4cq_4)(q_4zq_2), \\ q_0zq_3 & \to & b(q_4cq_0)(q_0zq_3)|b(q_4cq_1)(q_1zq_3)|b(q_4cq_2)(q_2zq_3)| \\ & & b(q_4cq_3)(q_3zq_3)|b(q_4cq_4)(q_4zq_3), \\ q_0zq_4 & \to & b(q_4cq_0)(q_0zq_4)|b(q_4cq_1)(q_1zq_4)|b(q_4cq_2)(q_2zq_4)| \\ & & b(q_4cq_3)(q_3zq_4)|b(q_4cq_4)(q_4zq_4)。 \end{array}$$

由第二个转移函数，可得产生式集合：

$$\begin{array}{lcl} q_4cq_0 & \to & (q_0cq_0)(q_0cq_0)|(q_0cq_1)(q_1cq_0)|(q_0cq_2)(q_2cq_0)| \\ & & (q_0cq_3)(q_3cq_0)|(q_0cq_4)(q_4cq_0), \\ q_4cq_1 & \to & (q_0cq_0)(q_0cq_1)|(q_0cq_1)(q_1cq_1)|(q_0cq_2)(q_2cq_1)| \\ & & (q_0cq_3)(q_3cq_1)|(q_0cq_4)(q_4cq_1), \\ q_4cq_2 & \to & (q_0cq_0)(q_0cq_2)|(q_0cq_1)(q_1cq_2)|(q_0cq_2)(q_2cq_2)| \\ & & (q_0cq_3)(q_3cq_2)|(q_0cq_4)(q_4cq_2), \\ q_4cq_3 & \to & (q_0cq_0)(q_0cq_3)|(q_0cq_1)(q_1cq_3)|(q_0cq_2)(q_2cq_3)| \\ & & (q_0cq_3)(q_3cq_3)|(q_0cq_4)(q_4cq_3), \\ q_4cq_4 & \to & (q_0cq_0)(q_0cq_4)|(q_0cq_1)(q_1cq_4)|(q_0cq_2)(q_2cq_4)| \\ & & (q_0cq_3)(q_3cq_4)|(q_0cq_4)(q_4cq_4)。 \end{array}$$

由第三个转移函数，可得产生式集合：

$$(q_0cq_1) \to a。$$

由第四个转移函数，可得产生式集合：

$$(q_1cq_1) \to a。$$

由第五个转移函数，可得产生式集合：

$$\begin{array}{lcl}
q_1zq_0 & \to & a(q_3cq_0)(q_0zq_0)|a(q_3cq_1)(q_1zq_0)|a(q_3cq_2)(q_2zq_0)| \\
 & & a(q_3cq_3)(q_3zq_0)|a(q_3cq_4)(q_4zq_0), \\
q_1zq_1 & \to & a(q_3cq_0)(q_0zq_1)|a(q_3cq_1)(q_1zq_1)|a(q_3cq_2)(q_2zq_1)| \\
 & & a(q_3cq_3)(q_3zq_1)|a(q_3cq_4)(q_4zq_1), \\
q_1zq_2 & \to & a(q_3cq_0)(q_0zq_2)|a(q_3cq_1)(q_1zq_2)|a(q_3cq_2)(q_2zq_2)| \\
 & & a(q_3cq_3)(q_3zq_2)|a(q_3cq_4)(q_4zq_2), \\
q_1zq_3 & \to & a(q_3cq_0)(q_0zq_3)|a(q_3cq_1)(q_1zq_3)|a(q_3cq_2)(q_2zq_3)| \\
 & & a(q_3cq_3)(q_3zq_3)|a(q_3cq_4)(q_4zq_3), \\
q_1zq_4 & \to & a(q_3cq_0)(q_0zq_4)|a(q_3cq_1)(q_1zq_4)|a(q_3cq_2)(q_2zq_4)| \\
 & & a(q_3cq_3)(q_3zq_4)|a(q_3cq_4)(q_4zq_4)。
\end{array}$$

由第六个转移函数，可得产生式集合：

$$(q_3cq_1) \to \varepsilon。$$

由第七个转移函数，可得产生式集合：

$$(q_1zq_2) \to a。$$

文法的初始变量为(q_0zq_2)。

问题

给定集合$\Sigma = \{0, 1, c\}$上的语言$L = \{0^i 1^i c 0^k 1^k | k = i + 1\}$，试给出生成语言$L$的上下文无关文法。

解答

不存在生成语言L的上下文无关文法。如果存在上下文无关文法生成语言L，则语言L是上下文无关语言。可以用泵引理证明语言L 不是上下文无关语言。

用反证法证明，假设L是上下文无关语言，则存在正整数m，满足上下文无关语言的泵引理。取符号串

$$w = 0^m 1^m c 0^{m+1} 1^{m+1},$$

其长度大于m，从而w 可以分解为

$$w = uvxyz$$

的形式，其中$|vxy| \le m$，$|vy| \ge 1$。根据泵引理，$w_0 = uxz \in L$。

由于$|vxy| \le m$，$|vxy|$有以下情形：

1. 如果vxy全在0^m中，则v和y全是0，易知$w_0 = 0^{m-i-j} 1^m c 0^{m+1} 1^{m+1}$，显然有$w_0$不属于语言$L$，与$w_0 \in L$矛盾；
2. 如果vxy全在1^m或者0^{m+1}或者1^{m+1}中，类似上面分析，有w_0不属于语言L，与$w_0 \in L$矛盾；
3. 如果vxy一部分在0^m中，一部分在1^m中，则v和y含有部分0和部分1，由泵引理，易知$w_0 = 0^{m-i} 1^{m-j} c 0^{m+1} 1^{m+1} \in L$，从而在$w_0$中，前半部分0的个数和后半部分0的个数不是相差1个，前半部分1的个数和后半部分1的个数也不是相差1个，从而有w_0不属于语言L，与$w_0 \in L$矛盾；
4. 如果vxy一部分在1^m中，一部分在0^{m+1}中，或者一部分在0^{m+1}中，一部分在1^{m+1}中，类似上面分析，有w_0不属于语言L，与$w_0 \in L$矛盾。

由上面分析，无论vxy如何取值，均与泵引理的结果矛盾，从而语言L不是上下文无关语言。

问题

给定语言

$$L = \{a^n b^m c^k : k, m, n \geq 0, n = m\text{或者}k = m\},$$

1. 试构造一个上下文无关文法G，使得文法G生成语言L。
2. 依据文法G，给出生成语言$L' = L - \{\varepsilon\}$的上下文无关文法G'，使其具有乔姆斯基范式形式。
3. 依据文法G'，使用CYK算法给出符号串$aabbcc$的分析，并列出所有可能的推导树。

解答

1、构造文法$G = (\{S, A, B, C, D\}, \{a, b, c\}, S, P)$，其中产生式$P$包括：

$$\begin{array}{ll} S \to A|B, & \\ A \to C|Ac, & C \to \varepsilon|aCb, \\ B \to D|aB, & D \to \varepsilon|bDc。 \end{array}$$

2、去ε产生式：

$$\begin{array}{ll} S \to A|B, & \\ A \to C|Ac|c, & C \to ab|aCb, \\ B \to D|aB|a, & D \to bc|bDc。 \end{array}$$

去单位产生式：

$$\begin{array}{ll} S \to ab|aCb|Ac|c|bc|bDc|aB|a, & \\ A \to ab|aCb|Ac|c, & C \to ab|aCb, \\ B \to bc|bDc|aB|a, & D \to bc|bDc。 \end{array}$$

转化为乔姆斯基范式形式：

$$\begin{array}{ll} S \to XY|EY|AZ|c|YZ|FZ|XB|a, & \\ A \to XY|EY|AZ|c, & C \to XY|EY, \\ B \to YZ|FZ|XB|a, & D \to YZ|FZ, \\ X \to a, & Y \to b, \\ Z \to c, & \\ E \to XC, & F \to YD。 \end{array}$$

3、使用CYK算法分析符号串$aabbcc$，第一步可得：

$$V_{11} = V_{22} = \{X, S, B\}，V_{33} = V_{44} = \{Y\}，V_{11} = V_{22} = \{Z, S, A\},$$

第二步可得：

$$V_{12}=\{S,B\}，V_{23}=\{S,C,A\}，V_{34}=\emptyset，V_{45}=\{S,B,D\}，V_{56}=\{S,A\}，$$

第三步可得：

$$V_{13}=\{E\}，V_{24}=\emptyset，V_{35}=\{F\}，V_{46}=\emptyset，$$

第四步可得：

$$V_{14}=\{S,C,A\}，V_{25}=\emptyset，V_{36}=\{S,B,D\}，$$

第五步可得：

$$V_{15}=\{S,A\}，V_{26}=\{S,B\}，$$

最后可得：

$$V_{16}=\{S,A,B\}。$$

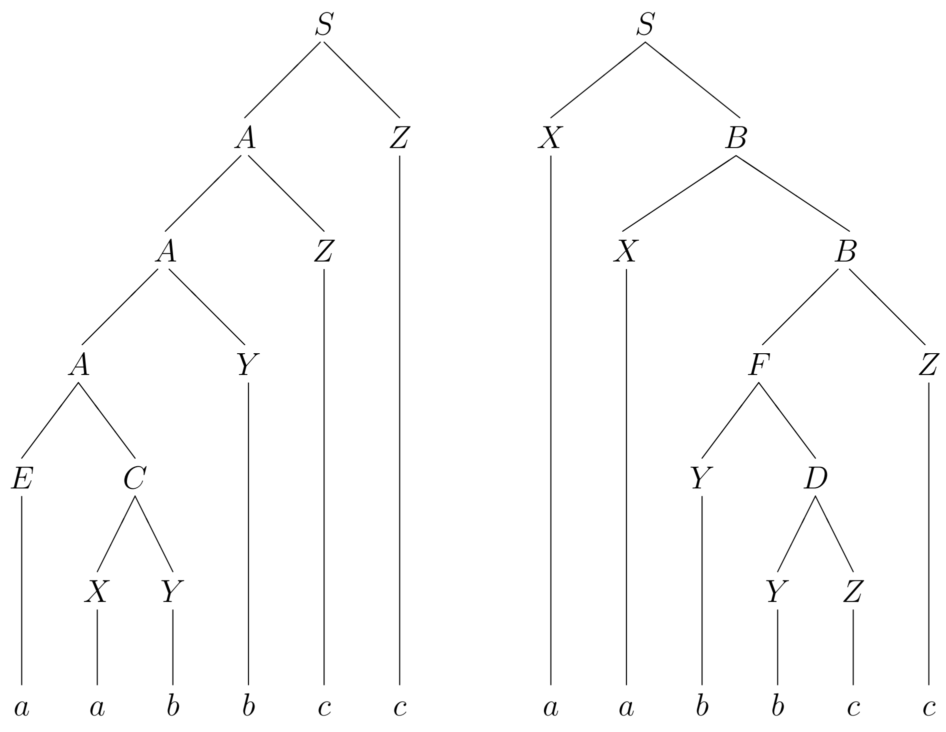
因此，可得符号串$aabbcc$的推导过程为：

$$S\Rightarrow AZ\Rightarrow AZZ\Rightarrow EYZZ\Rightarrow XCYZZ\Rightarrow XXYYZZ\Rightarrow\cdots\Rightarrow aabbcc，$$

和

$$S\Rightarrow XB\Rightarrow XXB\Rightarrow XXFZ\Rightarrow XXYDZ\Rightarrow XXYYZZ\Rightarrow\cdots\Rightarrow aabbcc。$$

推导所对应的推导树分别为：



问题

类似图灵机的编码过程，可以对下推自动机进行编码，并且对于任给的一个下推自动机的编码，可以按照唯一的方式解码。用$\langle M\rangle$表示下推自动机M的编码，并给定自然数n，记语言$L=\{\langle M\rangle : M$是下推自动机，并且$L(M)$至少包含一个长度不大于n的符号串$\}$，证明语言L是递归语言。

解答

为了证明语言L是递归语言，需要给出L的成员资格判定算法。

对于$\{0,1\}^+$上的符号串w，先检查该编码是否定义了一个下推自动机，如果不是输出一个“否”。

如果该编码定义了一个下推自动机，记为M。构造确定型有穷接受器M'，其接受的语言为字母表上长度不大于n的所有符号串。

我们构造一个下推自动机M_1，其接受的语言为$L(M)\cap L(M')$。这样，$L(M)$至少包含一个长度不大于n的符号串当且仅当$L(M_1)$非空。

将M_1转化为一个上下文无关文法G。存在一个算法，可以判定该上下文无关文法生成的语言是否为空。如果为空，输出一个“否”；如果不为空，输出一个“是”。

由上面的算法可知，语言L存在一个成员资格判定算法，从而语言L是递归语言。

问题

构造一个字母表$\{a,b\}$上的图灵机，其接受的语言所包含的符号串满足如下条件：符号串的任意前缀中(除去ε)，字母a的个数大于字母b的个数。然后根据所构造的图灵机，给出其接受符号串$aabab$的瞬时描述序列。

解答

构造满足条件的图灵机如下：

$$M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{a, b, x, \Box\}, \delta, q_0, \Box, \{q_3\}),$$

其中转移函数为

$\delta(q_0, a) = (q_1, \Box, R)$，　(去掉一个符号$a$)
$\delta(q_1, a) = (q_1, a, R)$，　($q_1$去找下一个$b$)
$\delta(q_1, x) = (q_1, x, R)$，　($q_1$遇到$a$和$x$不变)
$\delta(q_1, b) = (q_2, x, L)$，　(遇到一个$b$，将其标记为$x$)
$\delta(q_2, x) = (q_2, x, L)$，　($q_2$去找前面是否有一个$a$)
$\delta(q_2, a) = (q_1, x, R)$，　(找到一个与$b$匹配的$a$，去找下一个$b$)
$\delta(q_1, \Box) = (q_3, \Box, L)$。　(找不到$b$，遇到$\Box$进入终止状态)

(注：先去掉一个a，这样符号串的任意前缀中字母a的个数$\geq$字母b的个数，就不用考虑把如何等于的情形剔除掉)

该图灵机接受符号串$aabab$的瞬时描述序列如下：$q_0aabab \vdash \Box q_1abab \vdash aq_1bab \vdash q_2axab \vdash xq_1xab \vdash xxq_1ab \vdash xxaq_1b \vdash xxq_2ax \vdash xxxq_1x \vdash xxxxq_1\Box \vdash xxxq_3x$。

问题

设语言L_1和L_2是递归可枚举语言，语言$L_1 \cap L_2$和$L_1 \cup L_2$是递归语言。证明：语言L_1和L_2是递归语言。

解答

我们给出语言L_1的成员资格判定算法，根据对称性，可以给出语言L_2的成员资格判定算法。

语言$L_1 \cap L_2$和$L_1 \cup L_2$是递归语言，这两个语言存在成员资格判定算法。对于字母表上的任意符号串x，我们用如下算法判定它是否属于语言L_1：

1、将符号串x输入语言$L_1 \cup L_2$的成员资格判定算法，查看它是否属于语言$L_1 \cup L_2$。

- 如果符号串x不属于语言$L_1 \cup L_2$，算法输出“拒绝”并停止；
- 如果符号串x属于语言$L_1 \cup L_2$，算法进入下一步。

2、将符号串x输入语言$L_1 \cap L_2$的成员资格判定算法，查看它是否属于语言$L_1 \cap L_2$。

- 如果符号串x属于语言$L_1 \cap L_2$，算法输出“接受”并停止；
- 如果符号串x不属于语言$L_1 \cap L_2$，算法进入下一步。

3、此时，我们知道符号串x要么属于语言L_1，要么属于语言L_2，但不会同时属于这两个语言。我们需要判定符号串x具体属于哪一个语言。

语言L_1是递归可枚举语言，存在图灵机M_1接受语言L_1；语言L_2是递归可枚举语言，存在图灵机M_2接受语言L_2。将符号串x分别输入到图灵机M_1和图灵机M_2中，并执行i步迁移，$i = 1, 2, 3, \cdots$，直到其中一个图灵机接受符号串x。(由于符号串x要么属于语言L_1，要么属于语言L_2，这个事件一定会在某个时刻发生。)

- 如果图灵机M_1接受了符号串x，算法输出“接受”并停止；
- 如果图灵机M_2接受了符号串x，算法输出“拒绝”并停止。

这样，我们给出了语言L_1的成员资格判定算法，因此，语言L_1是递归语言。

同样，我们可以给出语言L_2的成员资格判定算法，因此，语言L_2是递归语言。

问题

设字母表$\Sigma=\{0,1,\#\}$上的语言L包含具有如下形式的符号串：

$$a_1a_2\cdots a_k\#b_1b_2\cdots b_k\#c_1c_2\cdots c_k,$$

其中$a_1,a_2,\cdots,a_k,b_1,b_2,\cdots,b_k,c_1,c_2,\cdots,c_k\in\{0,1\}$，并且满足条件，如果将$a_1a_2\cdots a_k$、$b_1b_2\cdots b_k$和$c_1c_2\cdots c_k$视为二进制数，则有

$$(c_1c_2\cdots c_k)_2=(a_1a_2\cdots a_k)_2+(b_1b_2\cdots b_k)_2。$$

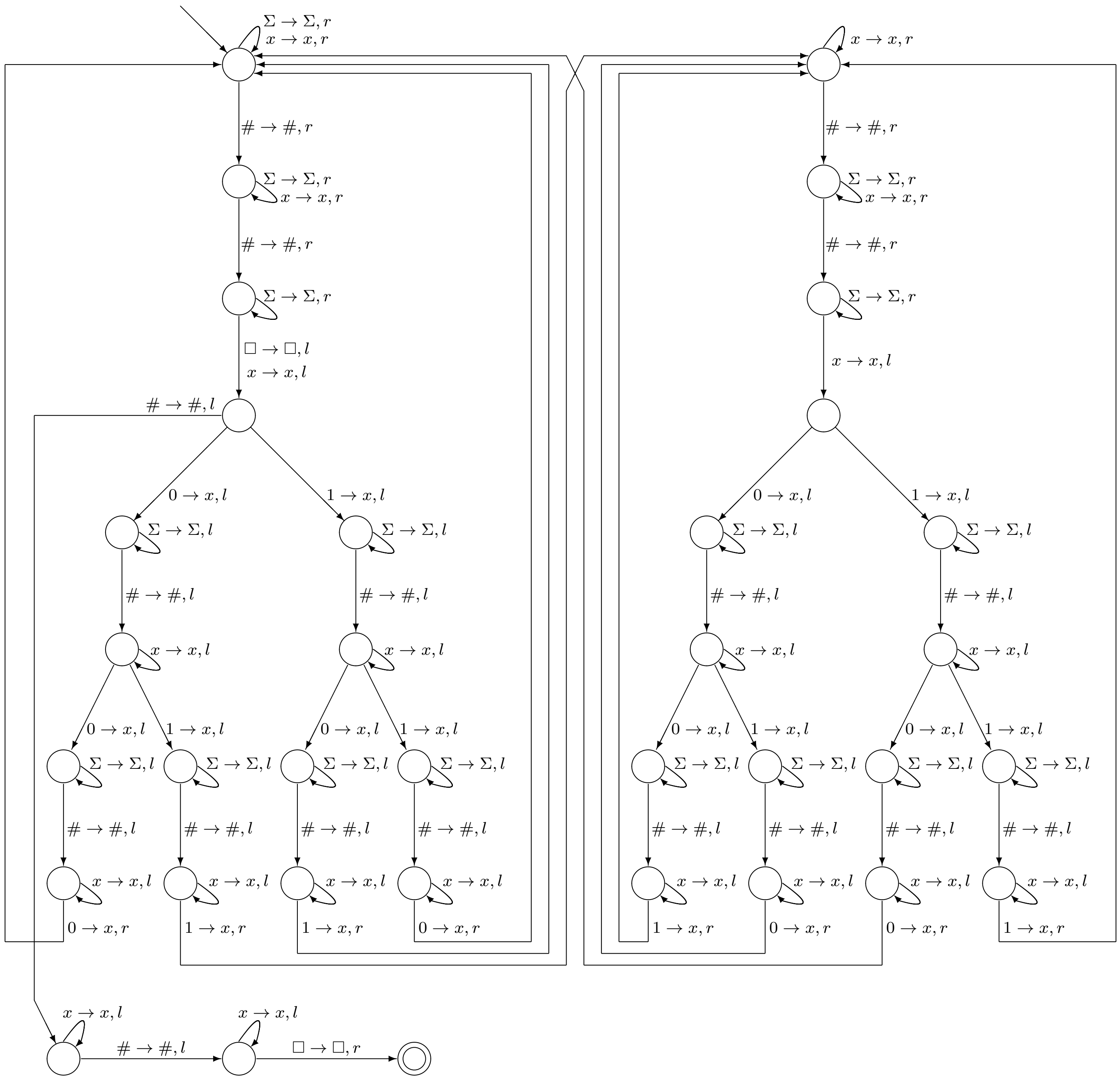
例如：由$(10010)_2=(01011)_2+(00111)_2$，可得$01011\#00111\#10010\in L$。判断语言$L$是否是递归语言，并证明你的结论。

解答

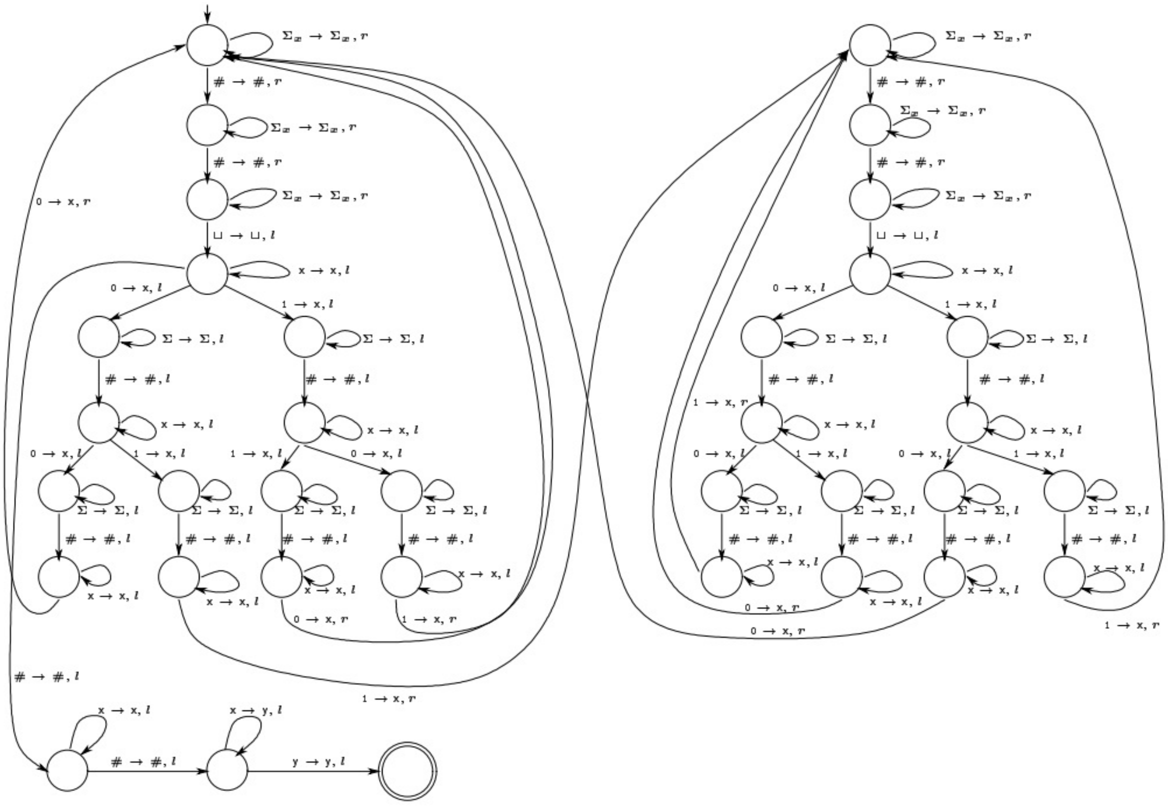
语言L是递归语言，为证明此结论，构造一个可以识别语言L的图灵机，并且对任意输入都可以停机。

图灵机的带字母表$\Gamma=\{0,1,\#,x,\Box\}$，它的运行方式是逐个比较w_i在二进制运算下是否真的是u_i和v_i的和。图灵机在检查了w_i之后，它将用符号x替换它。最后，当所有w的符号都被检查完毕时，图灵机检查u和v的所有符号是否都被检查到。

图灵机的状态转换图如下，图中的左半部对因无进位运算。右半部对应有进位运算：



Σ → Σ, r
x → x, r
x → x, r
→ #, r
→ #, r
Σ → Σ, r
x → x, r
Σ → Σ, r
x → x, r
→ #, r
→ #, r
Σ → Σ, r
Σ → Σ, r
□ → □, l
x → x, l
x → x, l
→ #, l
0 → x, l
1 → x, l
0 → x, l
1 → x, l
Σ → Σ, l
Σ → Σ, l
Σ → Σ, l
Σ → Σ, l
→ #, l
→ #, l
→ #, l
→ #, l
x → x, l
x → x, l
x → x, l
x → x, l
0 → x, l
1 → x, l
0 → x, l
1 → x, l
0 → x, l
1 → x, l
0 → x, l
1 → x, l
Σ → Σ, l
Σ → Σ, l
Σ → Σ, l
Σ → Σ, l
Σ → Σ, l
Σ → Σ, l
Σ → Σ, l
Σ → Σ, l
→ #, l
→ #, l
→ #, l
→ #, l
→ #, l
→ #, l
→ #, l
→ #, l
x → x, l
x → x, l
x → x, l
x → x, l
x → x, l
x → x, l
x → x, l
x → x, l
0 → x, r
1 → x, r
1 → x, r
0 → x, r
1 → x, r
0 → x, r
0 → x, r
1 → x, r
x → x, l
x → x, l
→ #, l
□ → □, r



问题

用$\langle M\rangle$表示图灵机M的编码，记语言$L=\{\langle M\rangle : M$是图灵机，并且$L(M)=\Sigma^*\}$，判定语言L是否是递归可枚举语言。

解答

(1) 语言$A_{TM}=\{\langle M,w\rangle : M$是图灵机，并且$w\in L(M)\}$是递归可枚举语言。

为了证明语言A_{TM}是递归可枚举语言，需要构造一个接受该语言的图灵机。实际上，通用图灵机M_u接受语言A_{TM}。

对于通用图灵机M_u，输入符号串为$\langle M,w\rangle$，其中M是一个(确定型)图灵机，w是M字母表上的符号串，通用图灵机M_u 模拟图灵机M作用在w上的动作：

- 如果图灵机M接受符号串w，通用图灵机M_u进入终止状态，接受符号串$\langle M,w\rangle$；
- 如果图灵机M拒绝符号串w，通用图灵机M_u进入非终止状态，拒绝符号串$\langle M,w\rangle$。

如果图灵机M作用在w上进入循环，通用图灵机M_u作用在$\langle M,w\rangle$上也进入循环。因此，通用图灵机M_u接受语言A_{TM}，即语言A_{TM}是递归可枚举语言。

(2) 语言A_{TM}不是递归语言，即$\overline{A_{TM}}=\{\langle M,w\rangle : M$是图灵机，并且$w$不属于语言$L(M)\}$不是递归可枚举语言。

用反证法，假设语言A_{TM}是递归语言，则语言A_{TM}存在成员资格判定算法，即存在图灵机H，判定符号串$\langle M,w\rangle$ 是否在语言A_{TM}中。我们由图灵机H构造如下图灵判定器D：

图灵机D的输入是符号串w。图灵机D将符号串w复制一次，使得存储带上为$\langle w,w\rangle$。如果$\langle w,w\rangle\in L(H)$(这里将第一个$w$视为一个图灵机的编码)，图灵机$D$拒绝符号串$w$，否则图灵机$D$接受符号串$w$。

由于图灵机H不会循环(H是判定算法)，图灵机D可以把H作为子程序使用。下面考虑图灵机D在输入符号串$w=\langle D\rangle$的动作：

- 如果$\langle\langle D\rangle,\langle D\rangle\rangle$在语言$L(H)$中，则图灵机$D$拒绝符号串$\langle D\rangle$，即符号串$\langle D\rangle$不在语言$L(D)$中，但是$\langle\langle D\rangle,\langle D\rangle\rangle$在语言$L(H)$ 中，图灵机D接受符号串$\langle D\rangle$，即符号串$\langle D\rangle$在语言$L(D)$中，矛盾；
- 如果$\langle\langle D\rangle,\langle D\rangle\rangle$不在语言$L(H)$中，则图灵机$D$接受符号串$\langle D\rangle$，即符号串$\langle D\rangle$在语言$L(D)$中，但是$\langle\langle D\rangle,\langle D\rangle\rangle$ 不在语言$L(H)$ 中，图灵机D不接受符号串$\langle D\rangle$，即符号串$\langle D\rangle$不在语言$L(D)$中，矛盾。

从而有假设错误，即语言A_{TM}不是递归语言。

(3) 语言$U_{TM}=\{\langle M\rangle : M$是图灵机，并且$L(M)=\Sigma^*\}$不是递归可枚举语言。

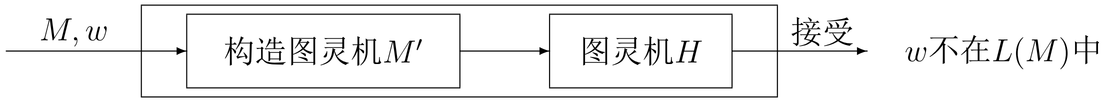
我们先构造如下图灵机M'，其输入是符号串w'：

- 图灵机M'记住$j=|w'|$，然后用符号串w重写w'；

- 图灵机M'模拟图灵机M作用在w上的动作，如果在j步内，M接受w，则图灵机M'拒绝符号串符号串w'；
- 如果在j步内，M没有接受w，则图灵机M'接受符号串符号串w'。

图灵机M'不会进入循环状态。如果符号串w在语言$L(M)$中，则存在一个整数j，使得图灵机M在j步内接受w，从而语言$L(M')$包含所有长度小于j的符号串；如果符号串w不在语言$L(M)$中，则不存在这样的整数j，即$L(M') = \Sigma^*$。($\overline{A_{TM}} \leq_m U_{TM}$)因此有语言$U_{TM}$不是递归可枚举语言。

注：用反证法证明。如果语言U_{TM}是递归可枚举语言，其可以被图灵机H接受，构造如下图灵机：



易知我们构造的图灵机可以识别语言$\overline{A_{TM}}$，但是语言$\overline{A_{TM}}$不是递归可枚举语言，矛盾。因此有语言U_{TM}不是递归可枚举语言。