编译原理作业 1 参考答案

第二章作业

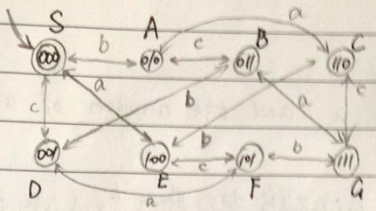
a) {w | w ∈ (a, b, c)* and the number of a's and b's and c's occurred in w are even}

解：(i) 画出电路状态转换图　(ii) 确定起始、终止状态

start : 000

end : 000

(iii) 为各状态命名 S, A~G



(iv) 写出产生式

$S \to \varepsilon \mid aE \mid bA \mid cD$ ， $A \to aC \mid bS \mid cB$ ， $B \to aG \mid bD \mid cA$

$C \to aA \mid bE \mid cG$ ， $D \to aF \mid bB \mid cS$ ， $E \to aS \mid bC \mid cF$

$F \to aD \mid bG \mid cE$ ， $G \to aB \mid bF \mid cC$

b) $\{a^i b^j \mid i \geq 2j+1 \text{ and } j \geq 0\}$

解：$a^i b^j = a^{i-2j} a^{2j} b^j$ 令 $i-2j=k$，则 $k \geq 1$，原式 $= \underbrace{a^k}_{A} \underbrace{a^{2j} b^j}_{B}$

即有 $S \to AB$

$A \to aA \mid a$ ， $B \to aaBb \mid \varepsilon$

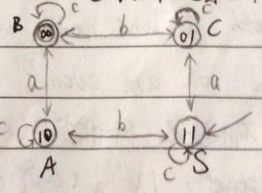
c) {w | w ∈ (a, b, c)* and the numbers of a's and b's occurred in w are odd}

解：(i) 画出电路状态转换图　(ii) 确定起、终状态

start : 11

end : 00

(iii) 为各状态命名 S, A~C



(iv) 写出产生式

$S \to aC \mid bA \mid cS$ ， $A \to aB \mid bS \mid cA$ ， $B \to \varepsilon \mid aA \mid bC \mid cB$ ， $C \to aS \mid bB \mid cC$

d) $\{a^i b^j \mid i \geq j+1, \text{ and } j \geq 0\}$

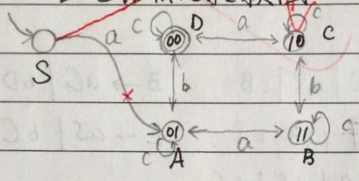
解：原式 $= a^{i-j}\, a^j b^j$　令 $i-j=k$，则 $k \geq 1$，原式 $= \underbrace{a^k}_{A}\, \underbrace{a^j b^j}_{B}$

即有　$S \to AB$

$A \to aA \mid a$，　$B \to aBb \mid \varepsilon$

e) $\{w \mid w \in (a,b,c)^*$, w is lead by a and the numbers of a's and b's occured in w are even$\}$

解：(i) 画电路状态转换图



(ii) 确定起、终状态（入口、出口）

start : 10　（a奇b偶）

end : 00

(iii) 给各状态命名 S, A~D

(iv) 写出产生式

$S \to aA$，　$A \to aB \mid bD \mid cA$，　$B \to aA \mid bC \mid cB$，

$C \to aD \mid bB \mid cC$，　$D \to aC \mid bA \mid cD \mid \varepsilon$

f) $\{a^{2i} b^{2j} \mid j \geq i \geq 1\}$

解：原式 $= b^{2j-2i}\, b^{2i} a^{2i}$　令 $k = j-i$，则 $k \geq 0$

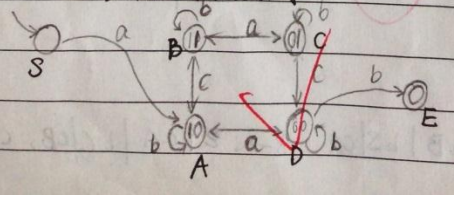
原式 $= \underbrace{b^{2k}}_{A}\, \underbrace{a^{2i} b^{2i}}_{B}$

即有　$S \to AB \mid BA$

$A \to bbA \mid \varepsilon$，　$B \to aaBbb \mid aabb$

g) $\{w \mid w \in (a,b,c)^*$ and w starts with a and ends with b, the numbers of a's and c's occured in w are even$\}$

解：(i) 画出状态转换图



(ii) 确定起、终状态（入口、出口）

start : 10　（a奇C偶）

end : 00

(iii) 为各状态命名 S, A~E

iv) 写出产生式

$S \to aA$, $A \to aD \mid bA \mid cB$, $B \to aC \mid bB \mid cA$.

$C \to aB \mid bC \mid cD$, $D \to aA \mid bD \mid cC \mid bE$, $E \to \varepsilon$

h) $\{a^i b^j c^k \mid j \geq i+k+1$, and $i \geq 0$, $k \geq 1\}$

解: 原式 $= a^i b^{j-i-k} b^i b^k c^k$ 令 $m = j-i-k$, 则 $m \geq 1$

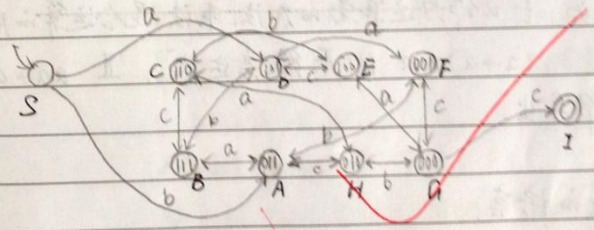
原式 $= \underbrace{a^i b^i}_{A} \underbrace{b^m}_{B} \underbrace{b^k c^k}_{C}$

即有 $S \to ABC$,

$A \to aAb \mid \varepsilon$, $B \to bB \mid b$, $C \to bCc \mid bc$

j) $\{w \mid w \in (a,b,c)^*$ and w starts with a or b, ends with c, and the numbers of a's and b's and c's occurred in w are even$\}$

解: (i) 画出确定状态转换图



(ii) 确定入口、出口状态

start with a: 101 (a奇b偶c奇)

start with b: 011 (a偶b奇c奇)

end: 000

(iii) 为各状态命名 S, A~I

(iv) 写出产生式

$S \to aD \mid bA$, $A \to aB \mid bF \mid cH$, $B \to aA \mid bD \mid cC$.

$C \to aH \mid bE \mid cB$, $D \to aF \mid bB \mid cE$, $E \to aG \mid bC \mid cD$

$F \to aD \mid bA \mid cG$, $G \to aE \mid bH \mid cF \mid cI$, $H \to aC \mid bG \mid cA$.

$I \to \varepsilon$

k) $a^{2i-1} b^{2j-1} c^{2k-1}$ $(i \geq 1, j \geq i+k, k \geq 1)$

解: 原式 $= a^{2i-1} b^{2j-2i-2k-1} b^{2i} b^{2k} c^{2k-1}$ 令 $m = j-i-k$ 则 $m \geq 0$

原式 = $\underbrace{a^{2i-1}\,b^{2i-1}}_{A}\ \underbrace{b^{2m+1}}_{B}\ \underbrace{b^{2k-1}\,c^{2k-1}}_{C}$

即有：$S \to ABC$

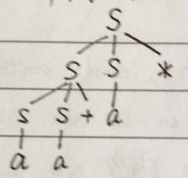
$A \to aaAbb \mid ab$ ， $B \to bBb \mid b$ ， $C \to bbCcc \mid bc$

☆下面是书本练习

2.2.1 $S \to SS+ \mid SS* \mid a$

a) $S \Rightarrow \underline{S}S* \Rightarrow S\underline{S}+S* \Rightarrow a\underline{S}+S* \Rightarrow aa+\underline{S}* \Rightarrow aa+a*$

b) 分析树构造如下



c) 该文法生成的语言为：{ 以a为基本操作数，以加和乘为基本操作符 ~~a作为运算数而加法、乘法混合运算的后缀表达~~

例如：$aa+a*$ 是 $(a+a)*a$ 的后缀表达式，且 $aa+a*$ 可以由该文法推导出。

2.2.2 写出文法各自生成的语言。

a) $S \to 0S1 \mid 01$

答：$L = \{ 0^n 1^n \mid n \geq 1 \}$

由该文法的"对称"特性容易看出，0和1数量相等，且0在前，1在后

例如，01，0011，000111

b) $S \to +SS \mid -SS \mid a$

答：$L = \{$ ~~以a为基本操作数，以加、减为基本操作符~~ ~~a为运算数而加减运算的前缀表达式~~ $\}$

例如：$+-aaa$ 是 $(a-a)+a$ 的前缀表达式，且

$S \Rightarrow +\underline{S}S \Rightarrow +-\underline{S}SS \Rightarrow +-a\underline{S}S \Rightarrow +-aa\underline{S} \Rightarrow +-aaa$

c) $S \to S(S)S \mid \varepsilon$

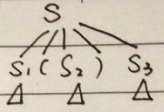
答：$L = \{$括号匹配的符号串，包括$\varepsilon\}$

证明如下：

(1) 先证该文法推导出的符号串都在 L 中。

(i) 考察最小语法树 $\overset{S}{\varepsilon}$，推导出的符号串 $\varepsilon \in L$.

(ii) 假设结点数 $< n$ 的语法树对应的符号串都在 L 中；

考察结点数 $= n$ 的语法树 S，其结构必为 

其子树 S_1, S_2, S_3 结点数 $< n$，因此对应的

符号串 $t_1, t_2, t_3 \in L$，S 对应的符号串 $t = t_1(t_2)t_3$，显然 $\in L$.

综(i)(ii)，(1)得证。

(2) 再证 L 中符号串都可由该文法推出。

(i) L 中最短符号串 ε，显然可由文法推导出；

(ii) 假定 L 中长度 $< n$ 的符号串都可由文法推导出，

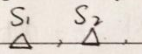
考虑长度 $= n$ 的符号串 $t = ((\ldots))\ldots((\ldots))$，寻找它的 $\in L$ 的

最短前缀 t'，则 $t = t't_1$，$t_1 \in L$（可能为 ε，显然可得）.

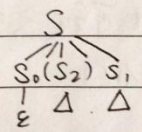
t' 的形式必为 $(\ )$ 或 $((\ldots))$，将它表示为 $t' = (t_2)$，无论

哪种情况，都有 $t_2 \in L$，

由于 t_1, t_2 长度 $< n$，因此可被文法推导出，对应语法树为



，则构造语法树



显然这是满足 t 的，即 t 可由文法推导出.

综(i)(ii)，(2)得证.

综(1)(2)，该文法描述语言就是 L.

d) $S \to aSbS \mid bSaS \mid \varepsilon$

答：$L = \{$相同个数的a、b组成的符号串，包括$\varepsilon\}$

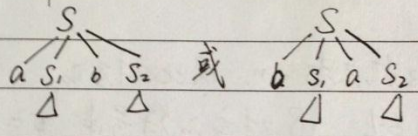
证明如下：

(1) 先证该文法推导出的符号串都在L中。

(i) 考虑最小语法树 $S \xrightarrow{\varepsilon}$，推导出的符号串 $\varepsilon \in L$ 成立。

(ii) 假设结点数 $< n$ 的语法树对应的符号串都 $\in L$，

那么考虑结点数 $= n$ 的语法树 S，其结构可能为



子树 S_1、S_2 结点数 $< n$，因此对应的符号串 $t_1, t_2 \in L$，

S 对应符号串 $t = at_1bt_2$ 或 bt_1at_2，显然 $\in L$。

综(i)(ii)，(1)得证；

(2) 再证L中符号串都可由该文法推导出。

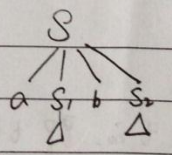
(i) L中最短符号串 ε，显然可由文法推导出；

(ii) 假设L中长度 $< n$ 的符号串都可由文法推导出。

考虑长度为 n 的符号串 t，考虑以a开头的情况，则 $t = a\ldots$，找出其中 $\in L$ 的最短前缀 t'，t'形式为 $a\ldots b$

将它表示为 $t' = at_1b$，显然 $t_1 \in L$（t_1可能为ε）

而 $t = t't_2$，$t_2 \in L$ 成立，则 $t = at_1bt_2$，即语法树



可推出t，即t可被文法推导出。（以b开头的情况类似）

综(i)(ii)，(2)得证。

综(1)(2)，该文法生成的语言就是L。

e) $S \to a \mid S+S \mid SS \mid S^* \mid (S)$ 以a为基本符号的正规表达式

答：$L = \{$ ~~支持加法、乘法、幂运算和括号的表达式~~ $\}$

例如：$S \Rightarrow \underline{S}+S \Rightarrow \underline{SS}+S \Rightarrow a\underline{S}+S \Rightarrow a\underline{S}^*+S \Rightarrow a(\underline{S})^*+S$

$\Rightarrow a(a)^*+\underline{S} \Rightarrow a(a)^*+a$

符号串 $a(a)^*+a \in L$ 显然成立

2.2.4 为下面的语言构造无二义性的上下文无关文法.

a) 算术表达式的后缀形式

解：$S \to num \mid SS+ \mid SS- \mid SS* \mid SS/$

例如：$num*num+num/num-num$ 可由如下方式推出其后缀形式

$S \Rightarrow \underline{SS}- \Rightarrow \underline{SS}+S- \Rightarrow \underline{SS}*S+S- \Rightarrow num\underline{S}*S+S-$

$\Rightarrow num\ num*\underline{S}+S- \Rightarrow num\ num*\underline{SS}/+S-$

$\Rightarrow num\ num*num\underline{S}/+S- \Rightarrow num\ num*num\ num/+\underline{S}-$

$\Rightarrow num\ num*num\ num/+num-$

正是该表达式的后缀形式. (postfix一定是无二义的)

b) 以“,”分隔的左结合的标识符列表

解：$IDS \to IDS, id \mid id$

由“左结合”性质的定义可得.

c) 以“,”分隔的右结合的标识符列表

解：$IDS \to id, IDS \mid id$

由“右结合”性质的定义可得.

(d) 含有整数和标识符、支持加减乘除运算的算术表达式.

解：考虑到运算的优先级不同，文法形式如下

$expr \to expr + term \mid expr - term \mid term$

note: 越靠近开始符，优先级越低. 例：$E \to E+F$

左递归(左结合)

term → term * factor | term / factor | factor

factor → id | num | (expr)

其中，"()"的引入是为了解决优先级问题.

e) 在此基础上，增加一元+，-运算符

解：类似(d)，文法格式如下：

expr → expr + term | expr - term | term

term → term * unary | term / unary | unary

unary → (+) factor | - factor

factor → id | num | (expr)

其中"(+)"表示"一元+"可以省略不写.

4.2.1 S → SS+ | SS* | a , string aa+a*.

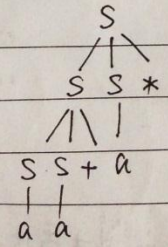
(a) 给出该字符串的最左推导：

S ⇒ SS* ⇒ SS+S* ⇒ aS+S* ⇒ aa+S* ⇒ aa+a*

(b) 给出该字符串的最右推导：

S ⇒ SS* ⇒ Sa* ⇒ SS+a* ⇒ Sa+a* ⇒ aa+a*

(c) 给出该字符串的分析树：



4.2.3 为下列语言设计文法

a) 含有0,1的所有字符串，且每个0后面跟着至少一个1.

解：$S \to 1S \mid 0A \mid \varepsilon$ 或 $S \to SS \mid 1 \mid 01 \mid \varepsilon$

$A \to 1S$

b) 0,1组成的回文字符串

解：使用对称法，其对称轴可能为0或1或ε.

那么有

$S \to 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon$

c) 0个数与1个数相同的字符串.

解：可使用等价法

$S \to 0A \mid 1B \mid \varepsilon$

$A \to 0AA \mid 1S \mid 1$

$B \to 1BB \mid 0S \mid 0$

可化简为

$S \to 0A \mid 1B \mid \varepsilon$

$A \to 0AA \mid 1S$

$B \to 1BB \mid 0S$

另一种表达形式可以为：$S \to 0S1S \mid 1S0S \mid \varepsilon$

d) 0与1的个数不同的字符串.

解：关键考虑0多和1多的问题，根据c)的思路调整得（递归）

~~$S \to A \mid B$~~

~~$A \to S_0T_0A \mid T_0S_0$，$T_0 \to 0T_0 \mid 0$~~

~~$B \to S_0T_1B \mid T_1S_0$，$T_1 \to 1T_1 \mid 1$~~

~~$S_0 \to 0S_01S_0 \mid 1S_00S_0 \mid \varepsilon$~~

$S \to A' \mid B'$　　$S_1 \to 1A \mid 0B \mid \varepsilon$

$A' \to AA' \mid A$　　$A \to 0S_1 \mid 1AA$

$B' \to BB' \mid B$　　$B \to 1S_1 \mid 0BB$

e) 0和1组成的字符串，但不含011子字符串

解：正规表达式可写作 $1^*(0 \mid 01)^*$

按照逐步求精法，可写出如下文法：

$S \to AB$，$A \to 1A \mid \varepsilon$，$B \to CB \mid \varepsilon$，$C \to 0 \mid 01$

f) 形如xy的0,1组成的字符串，$x \neq y$ 且x,y等长.

解：$S \to S_1S_2 \mid S_2S_1$

$S_1 \to aS_1a \mid aS_1b \mid bS_1a \mid bS_1b \mid a$

$S_2 \to aS_2a \mid aS_2b \mid bS_2a \mid bS_2b \mid b$

4.2.7 符号是"无用的"定义为：如果不存在如下形式的推导 $S \Rightarrow^{*} wXy \Rightarrow^{*} wxy$，那么 X 是"无用的"。

a) 算法设计：树型算法

(i) 构造树算法

① 对应文法的开始符号 S 创建树的根结点；

② 从根结点开始，若以某个已被创建为结点的非终结符 N 为左部符号的产生式有若干个候选式，那么：

1° 若某个候选式只有一个单独符号（$V_n \cup V_t$），就为该符号对应地创建一个结点，并把这个结点作为结点 N 的孩子结点。

2° 若某个候选式包含两个或多于两个符号，就创建一个空白结点，并将其作为结点 N 的孩子结点。再为该候选式中的每个符号（$V_t \cup V_n$）分别创建一个结点，并作为空白结点的孩子结点。

3° 约束条件：从根结点开始到当前结点的通路上，产生式集中的每个产生式最多只能出现一次。

③ 重复②中的操作，直到树中的结点不再增加为止。

(ii) 修剪树算法

① 检查树的所有叶结点，若某个叶结点是非终结符，那么删除以该叶子的真正父结点为根的子树（若某结点的父结点为空白结点，则空白结点的父结点为真正父结点，否则真正父结点即为自身）。

② 重复①中的操作，直到树中的结点不再减少为止。

通过构造树和修剪树算法得到的树，称作"干净树"。干净树的根结点是新文法的开始符号，干净树的所有终结符构成新文法的终结符集，所有非终结符构成新文法的非终结符集，出现在干净树中的所有产生式构成新文法的产生式集。

《参考文献：《一种新的文法化简方法——树型算法》郭新明，唐高峰，李长生，《咸阳师范学院学报》》

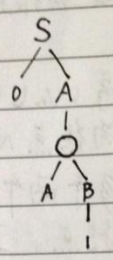
b) 应用示例

$S \to 0|A$

$A \to AB$

$B \to 1$

(i) 构造树



(ii) 修剪树

树中最下面的叶节点 A，是非终结符，因此要删除其真正父节为根的子树，如下

因此，最终的文法变为：$S \to 0$.

4.4.6 如果每个产生式的产生式体都不为 ε，那么该文法是"无 ε 产生式"

a) 算法设计

(i) 设 $G=(V_N, V_T, P, S)$ 是一文法，且 $\varepsilon \notin L(G)$.

这时可以按照以下算法构造一个文法 $G'=(V_N, V_T, P', S)$，使 $L(G')$ 且 G 中不含任何 ε-产生式.

① 将原文法 G 中的 ε-产生式之外的所有产生式放入 P' 中；

② 对 P' 中每个产生式的右部符号串，若含有 P 中的 ε-产生式左部非终结符号，则从中分别删除 ε-产生式的左部的非终结符形成新的产生式，加入 P' 中，

③ 若由(2)得到的 P' 中仍含有 ε-产生式，则将其看作 P，再重复前两步，直到全部 ε-产生式被消除为止。

此时，P′中不含任何ε-产生式。

(ii) 设 $G=(V_N, V_T, P, S)$，且 $\varepsilon \in L(G)$.

这时按以下算法构造一个文法 $G_1=(V_{N_1}, V_T, P_1, S_1)$，使 $L(G_1)=L(G)$ 且除了 $S_1\to\varepsilon$ 外，P_1 中不再含有其他 ε-产生式，此外，S_1 不出现在任何产生式的右边.

如果在原文法 G 中，S 不出现在任何产生式的右边，则可直接对 G 执行 (i) 中算法，消去 P 中的所有 ε-产生式。设所得的产生式集为 P′，不过，由于 $\varepsilon\in L(G)$，故不论原文法 G 中是否含有 $S\to\varepsilon$，都在使所得文法中含有此产生式，因此可令 $P_1=P'\cup\{S\to\varepsilon\}$，$V_{N_1}=V_N$，$S_1=S$，则 $G_1=(V_{N_1}, V_T, P_1, S_1)$ 即为所求。

如果文法的开始符S出现在某些产生式的右部，则

① 引入新的符号 S_1（$S_1\notin V_N\cup V_T$）作为 G′的开始符号，并令 $V_{N_1}=V_N\cup\{S_1\}$；

② 作产生式集

$$P'=P\cup\{S_1\to\alpha \mid \text{产生式 } S\to\alpha\in P\}$$

③ 对文法 $G'=(V_{N_1}, V_T, P_1, S_1)$ 执行 (i) 中算法，消去 P′中的全部ε-产生式，再把 $S_1\to\varepsilon$ 添加到所得的产生式集中，设所得的产生式集为 P_1，则 $G_1=(V_{N_1}, V_T, P_1, S_1)$ 即为所求。

b) 应用示例

$$S\to aSbS \mid bSaS \mid \varepsilon$$

∵ $\varepsilon\in L(G)$ 且 S 出现在某些产生式的右部

∴ 使用 (ii) 算法的第二种情况

步骤 (1) 引入新符号 S_1 作为新文法 G_1 的开始符号，且令 $V_{N_1}=\{S\}\cup\{S_1\}=\{S, S_1\}$

(2) $P'=P\cup\{S_1\to\varepsilon\}=\{S\to aSbS,\ S\to bSaS,\ S\to\varepsilon,\ S_1\to\varepsilon\}$

$S_1\to S$

(3) 对 G′消去 P′中全部ε-产生式之后，$P'=\{S\to aSbS,\ S\to bSaS,\ S\to ab,\ S\to ba,\ S_1\to aSbS,\ S_1\to bSaS,\ S_1\to ab,\ S_1\to ba\}$.

然后 $P_1=P'\cup\{S_1\to\varepsilon\}=\{S\to aSbS,\ S\to bSaS,\ S\to ab,\ S\to ba,\ S_1\to aSbS,\ S_1\to bSaS,\ S_1\to ab,\ S_1\to ba,\ S_1\to\varepsilon\}$.

此时 $G_1=(V_{N_1}, V_T, P_1, S_1)$ 中不含 $S_1\to\varepsilon$ 以外的ε-产生式，且 S_1 不出现在任何产生式的右部。