

答案仅供参考!

Homework 1

(1) 观察讲义 lec1 中 P4 和 P11 上的函数 fact 的 C 代码及其汇编代码，初步了解编译器的作用。你可以：

(a) 简要注释每条汇编代码；

```
.file    "fact.c"    //相当于函数声明
```

```
.text
```

```
.globl fact
```

```
.type    fact, @function
```

```
fact:
```

```
.LFB0:
```

```
.cfi_startproc
```

```
pushl    %ebp    //保存主调函数的 frame 栈帧
```

```
.cfi_def_cfa_offset 8
```

```
.cfi_offset 5, -8
```

```
movl %esp, %ebp
```

```
.cfi_def_cfa_register 5
```

```
subl $24, %esp //将栈空出 24byte, 用于存放局部变量
```

```
cmpl $0, 8(%ebp) //将传入参数 n 与 0 进行比较
```

```
jg .L2 //n > 0
```

```
movl $1, %eax
```

```
jmp .L3 // n <= 0
```

```
.L2:
```

```
movl 8(%ebp), %eax
```

```
subl $1, %eax // n - 1
```

```
movl %eax, (%esp)
```

```
call fact // 递归调用
```

```
imull 8(%ebp), %eax // n * fact(n - 1)
```

```
.L3:
```

```
leave //函数开始返回
```

```
.cfi_restore 5
```

```
.cfi_def_cfa 4, 4
```

```
ret
```

```
.cfi_endproc
```

```
.LFE0:
```

```
.size fact, .-fact
```

```
.ident "GCC: (Ubuntu 4.8.4-2ubuntu1~14.04.4) 4.8.4"
```

```
.section .note.GNU-stack,"",@progbits
```

(b) 尝试指出 C 程序与汇编代码间的联系,

比如, C 程序中的参数 n 在汇编中是如何表示的; if 语句对应哪几条汇编代码.....

8(%ebp) 为 n

if 条件判别由 `cmpl` 指令实现

(2) 针对以下 C 程序, 给出标号 L 处变量 j 可能的值集合。

```
int main()
```

```
{int i,j = 0;
```

```
for(i=0;i<10;i++)
{ switch(i)
{
case 0:case 2: break;
case 3:case 5: continue;
    default:  j = i;
}
L: j += i * 2;
}
}
```

i = 0, j = 0

i = 1, j = 1 + 2 = 3

i = 2, j = 3 + 4 = 7

i = 3, 语句不执行

i = 4, j = 4 + 8 = 12

i = 5, 语句不执行

i = 6, j = 6 + 12 = 18

i = 7, j = 7 + 14 = 21

$i = 8, j = 8 + 16 = 24$

$i = 9, j = 9 + 18 = 27$

故取值集合为{0, 3, 7, 12, 18, 24, 27}

(3) 针对以下 C/C++ 程序：

(3.1) 补全相关代码

(3.2) 用文字简要描述变量 b 和 p 的类型信息。

如变量 a 的类型信息描述如下：变量 a 是一个含 10 个元素的数组，每个元素是指向一个整型变量的指针。

```
int main()  
  
{  
  
    int i;  
  
    int* q;  
  
    int* a[10];  
  
    int* (*b[10])[10];  
  
    int* ((*p)[10])[10];  
  
  
    i = 100; q = &i; a[1] = q; b[1] = &a; p = &b;  
  
    cout << *(*p[0][1])[1] << endl; //输出 100, 待补全
```

```
cout <<  *(*p)[1]  << endl; //输出 100, 待补全
```

```
cout <<  *p[0][1][0][1]  << endl; //输出 100, 待补全
```

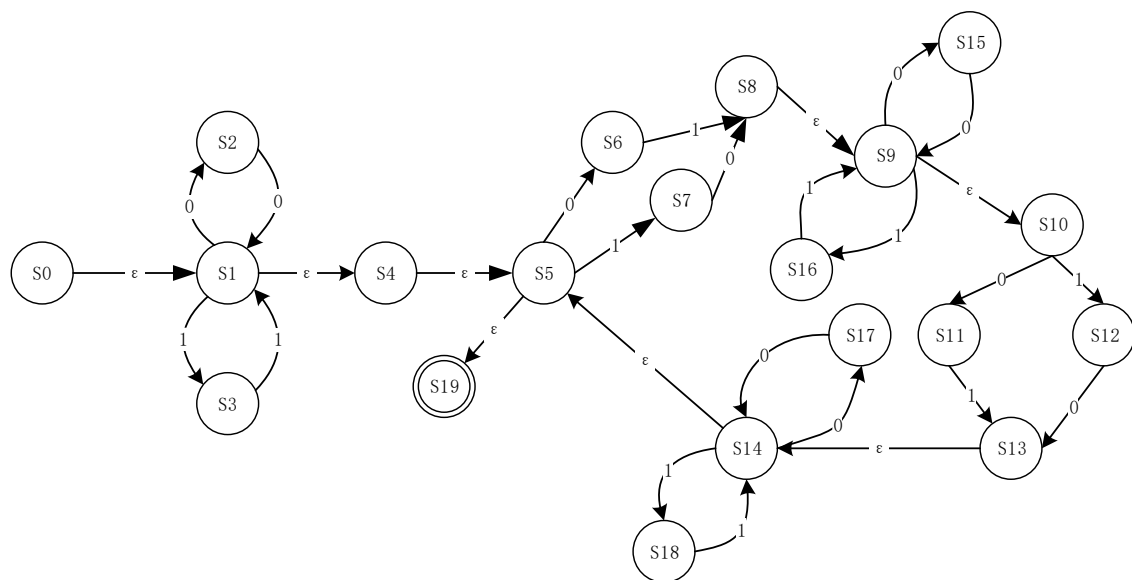
```
cout <<  p[0][1][0][1][0]  << endl; //输出 100, 待补全}
```

b 类型： b 是一个含有 10 个元素的数组，每个元素是一个指针，指向一个含有 10 个元素的数组，该数组元素为整型指针

p 类型： p 是一个指向 b 类型的指针

Homework 2

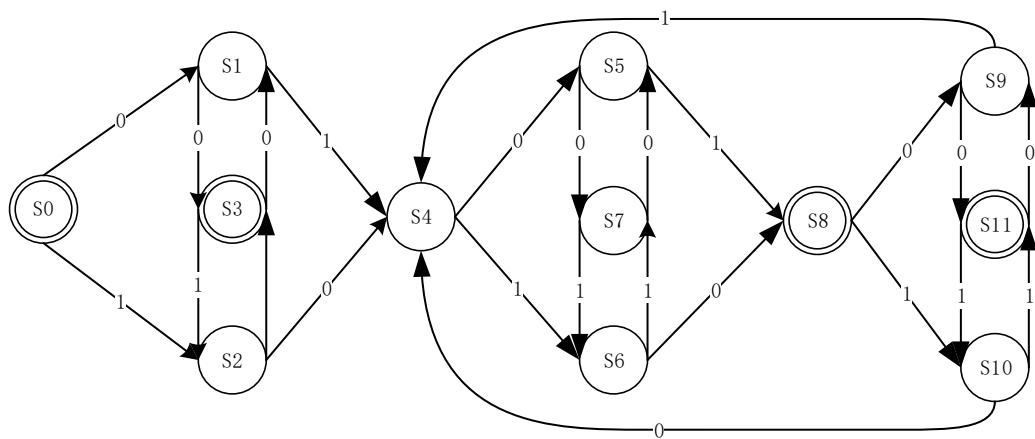
(1)采用 lec3 P38 所列方法,给出习题 2.3 中(e)NFA。
并对此 NFA 进行确定化和极小化。



确定化

Σ	0	1
状态 S_d	$\delta_d: S_d1 \rightarrow 0 \ S_d2$	$\delta_d: S_d1 \rightarrow 1 \ S_d2$
$\{0,1,4,5,19\}$	$\{2,6\}$	$\{3,7\}$
$\{2,6\}$	$\{1,4,5,19\}$	$\{8,9,10\}$
$\{3,7\}$	$\{8,9,10\}$	$\{1,4,5,19\}$
$\{1,4,5,19\}$	$\{2,6\}$	$\{3,7\}$
$\{8,9,10\}$	$\{11,15\}$	$\{12,16\}$
$\{11,15\}$	$\{9,10\}$	$\{13,14,5,19\}$
$\{12,16\}$	$\{13,14,5,19\}$	$\{9,10\}$
$\{9,10\}$	$\{11,15\}$	$\{12,16\}$

{13,14,5,19}	{6,17}	{7,18}
{6,17}	{14,5,19}	{8, 9,10}
{7,18}	{8,9,10}	{14,5,19}
{14,5,19}	{6,17}	{7,18}

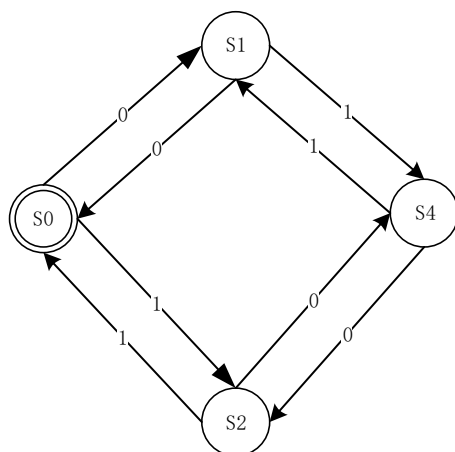


极小化

初始划分 $L_0 = \{0,3,8,11\}$, $L_1 = \{1,2,4,5,6,7,9,10\}$

a: $\rightarrow 0$ $L_0 = \{0,3,8,11\}$, $L_1 = \{1,6,9\}$, $L_2 = \{2,4,5,7,10\}$

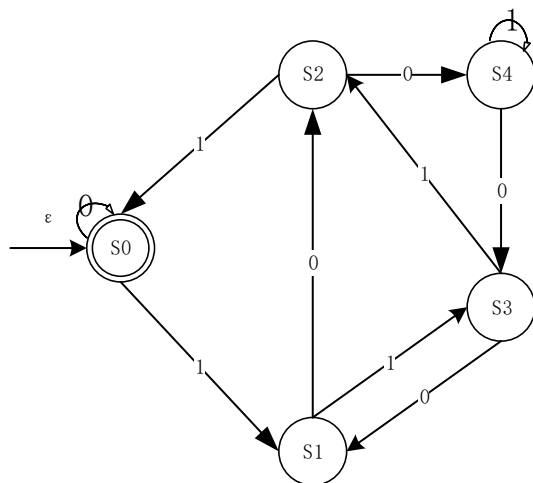
b: $\rightarrow 1$ $L_0 = \{0,3,8,11\}$, $L_1 = \{1,6,9\}$, $L_2 = \{2,5,10\}$, $L_3 = \{4,7\}$



习题 2.14

一个数 mod 5 结果为 0,1,2,3,4, 以此为 5 种状态。由于要求是能被 5 整除的数, $0 \bmod 5 = 0$ 满足, 故状态 0 为初始 & 最终状态, 状态表如下:

	+0	+1
0	0	1
1	2	3
2	4	0
3	1	2
4	3	4



算法:

$0^* ((1 [(10)^* (11|0) \{01^*0(01)^*(00|1)\}^*] 1)^* 0^*)^*$

第一个 0^* 为只进入 S_0 状态时的表达

从 S_0 进入 S_1 后, 会形成一个循环, 故有这个 $*$

第二个 0^* 为从 S_2 返回后, 可以在 S_0 处循环, 故有之

从 S_0 进入循环后，只有一条路径，先进入 S_1 ，故有前面的 1，也只有一条路径回到 S_0 ，即从 S_2 返回，故有后面的 1。又可以自然形成圈，故有一个 $*$ 。

现在已经可以去除 $S_0, S_0 \rightarrow S_1, S_2 \rightarrow S_0$. 以 S_1 为开始， S_2 为终点和中间点，

从 $S_1 \rightarrow S_2$, 有: $(10)^* (11|0)$, 单向无圈，故无 $*$

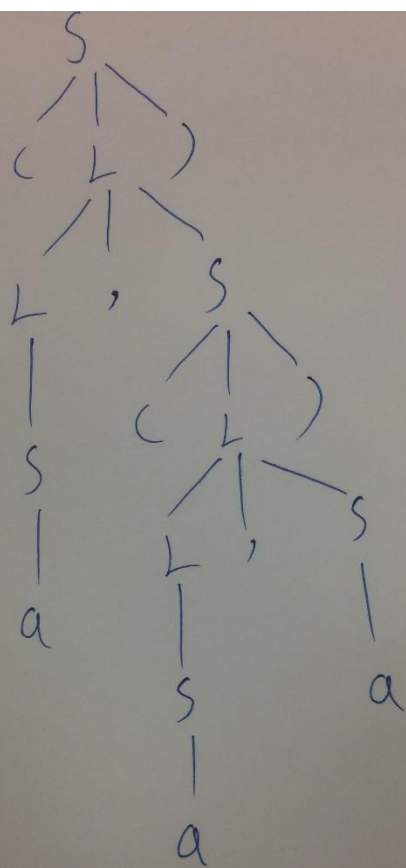
从 $S_2 \rightarrow S_2$, 可以形成圈，故有 $*$ ，有 $\{01^*0(01)^*(00|1)\}^*$

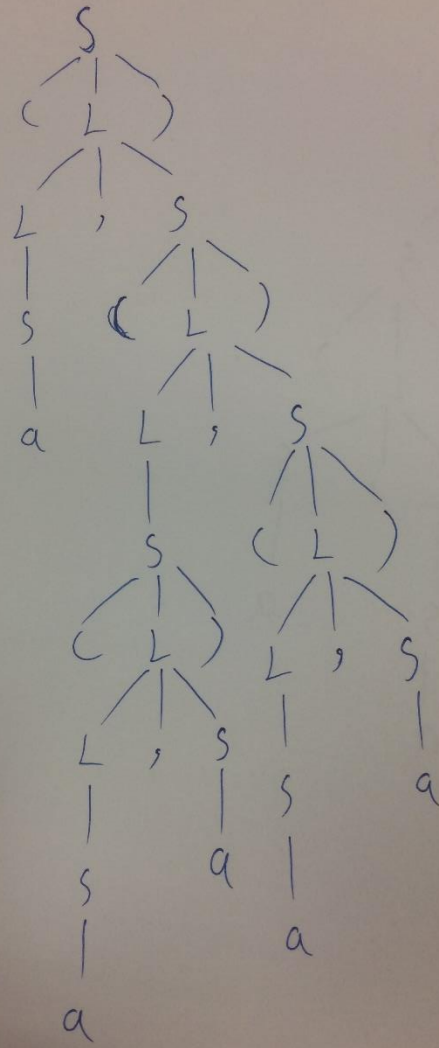
综上正规式可得。

另一种方法写 DNF 到 RE 的变换

$\textcircled{1} S_0 = \varepsilon + S_0 b + S_0 a$
 $\textcircled{2} S_1 = S_0 b + S_1 a$
 $\textcircled{3} S_2 = S_1 a + S_1 b$
 $\textcircled{4} S_3 = S_2 a + S_1 b$
 $\textcircled{5} S_4 = S_2 a + S_3 b$
 由⑤得
 $\textcircled{6} S_4 = S_2 a b^*$
 将⑥代入⑤得
 $\textcircled{7} S_3 = S_2 a b^* a + S_1 b$
 将⑦代入⑤得
 $S_2 = S_1 a + (S_2 a b^* a + S_1 b) \cdot b$
 $S_2 = S_1(a + b b) + S_2 a b^* a b$
 $\textcircled{8} S_2 = S_1(a + b b)[a b^* a b]^*$
 将⑧代入⑤中, 有
 $S_1 = S_0 b + (S_2 a b^* a + S_1 b) a$
 $\textcircled{9} S_1 = S_0 b + S_2 a b^* a a + S_1 b a$
 将⑨代入⑤中, 有
 $\textcircled{10} S_0 = \varepsilon + \cancel{S_2 b} + S_1(a + b b)[a b^* a b]^* + S_0 a$
 将⑩代入⑤中, 有
 $S_1 = S_0 b + S_1(a + b b)[a b^* a b]^* a b^* a a + S_1 b a$
 $S_1 = S_0 b + S_1 \underbrace{(a + b b)[a b^* a b]^* a b^* a a + b a}$

$\textcircled{11} S_1 = S_0 b Y^*$
 将⑪代入⑤中, 有
 $S_0 = \varepsilon + S_0 b Y^* (a + b b)[a b^* a b]^* + S_0 a$
 得
 $S_0 = [b Y^* (a + b b)[a b^* a b]^* + a]^*$
 其中 $a = 0, b = 1$, “+”即为“|”
 证明中运用如下等式:
 $R = Q + R P$
 $\Rightarrow R = Q P^*$
 转换后的正规式为:
 $\{1[0|11][01^*01]^*01^*00|10]^*(0|11)[01^*01]^*|0\}^*$
 P.S.:
 $\{ \}, []$ 类同 $()$.
 $|$ 为竖线, 非 1 .





(b) $S \Rightarrow (L) \Rightarrow (L, S) \Rightarrow (S, S) \Rightarrow (a, S) \Rightarrow (a, (L)) \Rightarrow (a, (L, S))$
 $\Rightarrow (a, (S, S)) \Rightarrow (a, (a, S)) \Rightarrow (a, (a, a))$

$S \Rightarrow (L) \Rightarrow (L, S) \Rightarrow (S, S) \Rightarrow (a, S) \Rightarrow (a, (L)) \Rightarrow (a, (L, S))$
 $\Rightarrow (a, (S, S)) \Rightarrow (a, ((L), S)) \Rightarrow (a, ((L, S), S)) \Rightarrow (a, ((S, S), S))$
 $\Rightarrow (a, ((a, S), S)) \Rightarrow (a, ((a, a), S)) \Rightarrow (a, ((a, a), (L)))$
 $\Rightarrow (a, ((a, a), (L, S))) \Rightarrow (a, ((a, a), (S, S))) \Rightarrow (a, ((a, a), (a, S)))$
 $\Rightarrow (a, ((a, a), (a, a)))$

(c) $S \Rightarrow (L) \Rightarrow (L, S) \Rightarrow (L, (L)) \Rightarrow (L, (L, S)) \Rightarrow (L, (L, a)) \Rightarrow (L, (S, a))$
 $\Rightarrow (L, (a, a)) \Rightarrow (S, (a, a)) \Rightarrow (a, (a, a))$
 $S \Rightarrow (L) \Rightarrow (L, S) \Rightarrow (L, (L)) \Rightarrow (L, (L, S)) \Rightarrow (L, (L, (L)))$

$\Rightarrow (L, (L, (L, S))) \Rightarrow (L, (L, (L, a))) \Rightarrow (L, (L, (S, a))) \Rightarrow (L, (L, (a, a)))$
 $\Rightarrow (L, (S, (a, a))) \Rightarrow (L, ((L), (a, a))) \Rightarrow (L, ((L, S), (a, a))) \Rightarrow (L, ((L, a), (a, a)))$
 $\Rightarrow (L, ((S, a), (a, a))) \Rightarrow (L, ((a, a), (a, a))) \Rightarrow (S, ((a, a), (a, a)))$
 $\Rightarrow (a, ((a, a), (a, a)))$

(d) 该文法产生的语言, 包括 a 或括号 $()$ 内一系列 a, a, \dots 串, 并且当中若干个 a 可被括号 $()$ 括住。

习题 3.3

文法

$A \rightarrow \text{true} \mid \text{false} \mid (S)$

$B \rightarrow \text{not } A \mid A$

$C \rightarrow C \text{ and } B \mid B$

$S \rightarrow S \text{ or } C \mid C$

Homework3

1.设计如下表格给出其中 C 算符的优先级和结合性

见 hw3-1.pdf

2.阅读 [ANSI C 语法](#)中 declaration 相关产生式，给出如下声明的分析树：

int* (*b[10])[10];

见 hw3-2.pdf

HW4

(1) 习题 3.9 (构造递归下降分析程序) , 3.11 和 3.16 (a 和 b)

(2.1) 删除以下文法 G 中的左递归, 并由此得到文法 G_1 。

	文法 G : A 是开始符号
1	$A \rightarrow B a$
2	$B \rightarrow d a b$
3	$B \rightarrow C b$
4	$C \rightarrow c B$
5	$C \rightarrow A c$

(2.2) G_1 是否为 LL (1) 的文法? 如不是, 适当修改该文法 G_1 , 使之成为 LL(1)的。

只需要去除不满足 LL (1) 文法的条件, 即可。

比如 $C' = acC' \mid \varepsilon$ 即可。

习题 3.9

将文法改造成不含左递归的非二义 LL(1) 文法:

 $S \rightarrow CS'$ $S' \rightarrow \text{or } CS' \mid \epsilon$ $C \rightarrow BC'$ $C' \rightarrow \text{and } BC' \mid \epsilon$ $B \rightarrow \text{not } A \mid A$ $A \rightarrow \text{true} \mid \text{false} \mid (S)$

递归下降预测分析器:

```
void match (terminal t) {
```

```
    if (lookahead == t) lookahead = nextToken();
```

```
    else error();
```

```
}
```

```
void S() { C(); S'(); }
```

```
void S'() {
```

```
    if (lookahead == or) { match(or); C(); S'(); }
```

```
}
```

```
void C() { B(); C'(); }
```

```
void C'() {
```

```
    if (lookahead == and) { match(and); B(); C'(); }
```

```
}
```

```

void B() {
    if (lookahead == not) { match(not); A(); }
    else if ((lookahead == true) || (lookahead == false) || (lookahead
    == '(')) { A(); }
    else error();
}

void A() {
    if (lookahead == true) { match(true); }
    else if (lookahead == false) { match(false); }
    else if (lookahead == '(') { match('('); S(); match(''); }
    else error();
}

```

문제 3.11

$FIRST(S) = \{a, b, \epsilon\}$, $FIRST(A) = \{a, b\}$, $FIRST(B) = \{a, b\}$

$FOLLOW(S) = \{\$ \}$, $FOLLOW(A) = \{a, b, \$ \}$, $FOLLOW(B) = \{a, b, \$ \}$

	a	b	\$
S	$S \rightarrow aBS$	$S \rightarrow bAS$	$S \rightarrow \epsilon$
A	$A \rightarrow a$	$A \rightarrow bAA$	
B	$B \rightarrow aBB$	$B \rightarrow b$	

习题 3.16

(a) 最右推导: $S \Rightarrow (L) \Rightarrow (L, S) \Rightarrow (L, (L)) \Rightarrow (L, (L, S)) \Rightarrow (L, (L, a))$
 $\Rightarrow (L, (S, a)) \Rightarrow (L, (a, a)) \Rightarrow (S, (a, a)) \Rightarrow (a, (a, a))$

其中每个句型的句柄由下划线标出

(b) 因为移进归约分析器跟踪最右推导过程的逆过程, 可列出其步骤:

栈	输入	动作
\$	(a, (a, a))\$	移进
\$(a, (a, a))\$	移进
\$(a	, (a, a))\$	按 $S \rightarrow a$ 归约
\$(S	, (a, a))\$	按 $L \rightarrow S$ 归约
\$(L	, (a, a))\$	移进
\$(L,	(a, a))\$	移进
\$(L,(a, a))\$	移进
\$(L,(a	, a))\$	按 $S \rightarrow a$ 归约
\$(L,(S	, a))\$	按 $L \rightarrow S$ 归约
\$(L,(L	, a))\$	移进
\$(L,(L,	a))\$	移进
\$(L,(L,a))\$	按 $S \rightarrow a$ 归约
\$(L,(L,S))\$	按 $L \rightarrow L, S$ 归约
\$(L,(L))\$	移进
\$(L,(L))\$	按 $S \rightarrow (L)$ 归约

Date

栈	输入	动作
\$(L, S)\$	按 $L \rightarrow L, S$ 归约
\$(L)\$	移进
\$(L)	\$	按 $S \rightarrow (L)$ 归约
\$S	\$	接受

2. 删除左递归:

$A \rightarrow Ba$
 $B \rightarrow dab | Cb$
 $C \rightarrow dabac' | cBC'$
 $C' \rightarrow bacC' | \epsilon$

该文法不是 LL(1) 的, 因为对于产生式 $C' \rightarrow bacC' | \epsilon$
 $FIRST(bacC') \cap FOLLOW(C') = \{b\}$

修改为 LL(1) 文法:

① 将 C 的产生式代入 B

$A \rightarrow Ba$
 $B \rightarrow dab | dabac'b | cBC'b$
 $C' \rightarrow bacC' | \epsilon$

② 提取 B 产生式中左因子 dab

$A \rightarrow Ba$
 $B \rightarrow dabC | cBC'b$

$C \rightarrow acC'b \mid \varepsilon$

$C' \rightarrow bacC' \mid \varepsilon$

③ 调整 C' 的定义, 使 $FIRST(C' \text{ 产生式}) \cap FOLLOW(C') = \emptyset$

$A \rightarrow Ba$

$B \rightarrow dabC \mid cBbC'$

$C \rightarrow acbC' \mid \varepsilon$

$C' \rightarrow acbC' \mid \varepsilon$

④ 观察发现, C' 可以与 C 合并

$A \rightarrow Ba$

$B \rightarrow dabC \mid cBbC$

$C \rightarrow acbC \mid \varepsilon$

⑤ 此时还有 $FIRST(acbC) \cap FOLLOW(C) = \{a\}$, 调整之

$A \rightarrow B$

$B \rightarrow dabaC \mid cBbaC$

$C \rightarrow cbaC \mid \varepsilon$

该文法为 $LL(1)$ 文法

2.2 等价的文法变换, G1 为 LL(1)

$$A \rightarrow Ba$$

$$B \rightarrow dab$$

$$| \epsilon b$$

$$C \rightarrow cB$$

$$C \rightarrow Ac$$

消去 C 得

$$A \rightarrow Ba$$

$$B \rightarrow dab | cBb | Bacb$$

去左递归

$$A \rightarrow Ba$$

$$B \rightarrow dab B' | cBb B'$$

$$B' \rightarrow acb B' | \epsilon$$

写出 B 的串为 $(dab | cBb)(acb)^*$

那么 A 对应串 $(dab | cBb)(acb)^* a$
 $= (dab | cBb) a (cba)^*$

但显然前者在 $(acb)^* a$ 处有重定义
即读到 a 时可能有 acb 或 a.

而后者 $a(cba)^*$ 则不会.

写后者对应的修改文法:

$$A \rightarrow daba A' | cBba A'$$

$$A' \rightarrow cba A' | \epsilon$$

$$B \rightarrow dab B' | cBb B'$$

$$B' \rightarrow acb B' | \epsilon$$