4.16 考虑文法G[S]，其产生式如下

S→AS | b

A→SA | a

(1)构造文法G[S]的LR(0)项目集规范族及相应的DFA。

(2)如果把每一个LR(0)项目看成一个状态，并从每一个形如B→α•Xβ的状态出发画一条标识为X的箭弧到状态B→αX•β,而且从每一个形如B→α•Aβ的状态出发画标记为ε的箭弧到所有形如A→•γ的状态。这样就得到了一个NFA。说明这个NFA与(a)的DFA是等价的。

(3)构造文法的SLR分析表。

(4)对于输入串bab，给出SLR分析器所作出的动作。

(5)构造文法的LR(1)分析表和LALR分析表。

解：(1)

其拓广文法G'：

(0) S' → S (1) S → AS

(2) S → b (3) A → SA

(4) A → a

构造其LR(0)项目集规范族和goto函数(识别活前缀的DFA)如下:

I0 = {S'→•S, S→•AS, S→•b, A→•SA, A→•a}

I1 = {S'→S•, A→S•A, A→•SA, A→•a, S→•AS, S→•b }

I2 = {S→A•S, S→•AS, S→•b, A→•SA, A→•a }

I3 = {A →a•}

I4 = {S→b•}

I5 = {A→SA•, S→A•S, S→•AS, S→•b, A→•SA, A→•a }

I6 = {A→S•A, A→•SA, A→•a, S→•AS, S→•b}

I7 = {S→AS•, A→S•A, A→•SA, A→•a, S→•AS, S→•b}

(2) 文法G[S]的LR(0)项目如下：

(0) S' → •S (1) S' → S•

(2) S → •AS (3) S → A•S

(4) S → AS• (5) S → •b

(6) S → b• (7) A → •SA

(8) A → S•A (9) A → SA•

(10)A → •a (11)A → a•

对上面的NFA通过求ε-cloasure确定化，得到与(1)相同的识别文法G[S]活前缀的DFA。因此，此NFA与(1)的DFA等价。

(3) 求FOLLOW集：

FOLLOW(S) = { a, b, ＄ }

FOLLOW(A) = { a, b }

G[S]的SLR分析表：

(4)

注：G[S]的SLR分析表中有移进－归约冲突，因此它不是一个SLR文法。其实，G[S]是一个二义性文法，对于句子abab有下面两棵不同的分析树。因此，G[S]不是任何LR文法。

(5) 文法G[S]的LR(1)项目集规范族及转移函数为：

I0 = {[S'→•S, $], [S→•AS, $/a], [S→•b, $/a], [A→•SA, a/b], [A→•a, a/b]}

I1 = {[S'→S•, $], [A→S•A, a/b], [A→•SA,a/b], [A→•a, a/b], [S→•AS, a/b], [S→•b, a/b]}

I2 = {[S→A•S, $/a/b], [S→•AS, $/a/b], [S→•b, $/a/b], [A→•SA, a/b], [A→•a, a/b],}

I3 = {[A→a•, a/b]}

I4 = {[S→b•, a/b/$]}

I5 = {[A→S•A, a/b], [A→•SA, a/b], [A→•a, a/b], [S→•AS, a/b], [S→•b, a/b]}

I6 = {[A→SA•, a/b], [S→A•S, a/b], [S→•AS, a/b], [S→•b, a/b], [A→•SA, a/b], [A→•a, a/b]}

I7 = {[S→AS•, a/b/$], [A→S•A, a/b], [A→•SA, a/b], [A→•a, a/b], [S→•AS, a/b], [S→•b, a/b]}

I8 = {[S→A•S, a/b], [S→•AS, a/b], [S→•b, a/b], [A→•SA, a/b], [a→•a, a/b]}

I9 = {[S→AS•, a/b/$], [A→S•A, a/b], [A→•SA, a/b], [A→•a, a/b], [S→•AS, a/b], [S→•b, a/b], }

I10 = {[S→b•, a/b]}

合并同心项目集(I2和I8，I7和I9，I4和I10)：

I0 = {[S'→•S, $], [S→•AS, $/a], [S→•b, $/a], [A→•SA, a/b], [A→•a, a/b]}

I1 = {[S'→S•, $], [A→S•A, a/b], [A→•SA,a/b], [A→•a, a/b], [S→•AS, a/b], [S→•b, a/b]}

I2,8 = {[S→A•S, $/a/b], [S→•AS, $/a/b], [S→•b, $/a/b], [A→•SA, a/b], [A→•a, a/b],}

I3 = {[A→a•, a/b]}

I4,10 = {[S→b•, a/b/$]}

I5 = {[A→SA•, a/b], [S→A•S, a/b], [S→•AS, a/b], [S→•b, a/b], [A→•SA, a/b], [A→•a, a/b]}

I6 = {[A→S•A, a/b], [A→•SA, a/b], [A→•a, a/b], [S→•AS, a/b], [S→•b, a/b]}

I7,9 = {[S→AS•, a/b/$], [A→S•A, a/b], [A→•SA, a/b], [A→•a, a/b], [S→•AS, a/b], [S→•b, a/b]}

LR(1)和LALR分析表从略。

7.2翻译算术表达式一（a＋b）\*（c＋d）＋（a＋b＋c）为

(a)四元式，

(b)三元式

(c)间接三元式

解答：

a)：四元式序列为：

op arg1 arg2 result

(1) + a b t1

(2) + c d t2

(3) \* t1 t2 t3

(4) uminus t3 　 t4

(5) + a b t5

(6) + t5 c t6

(7) + t4 t6 t7

b)：三元式序列为：

op arg1 arg2

(1) + a b

(2) + c d

(3) \* (1) (2)

(4) uminus (3)

(5) + a b

(6) + (5) c

(7) + (4) (6)

c)：间接三元式表示：

statement op arg1 arg2

(1) (11) (11) + a b

(2) (12) (12) + c d

(3) (13) (13) \* (11) (12)

(4) (14) (14) uminus 13

(5) (11) (15) + (11) c

(6) (15) (16) + (14) (15)

(7) (16)

7.3 试把下列C语言程序的可执行语句

main()

{

int i;

int a[10];

while (i<=10)

a[i]=0;

}

翻译为

(a)一棵语法树，

(b)后缀式，

(c)三地址代码。

解答： a)

b) 后缀式为： i 10 <= a i [] 0 = while

从理论上可以说 while ( i <= 10 ) a[i] = 0; 的后缀式如上面表示。但若这样表示，在执行while操作时，赋值语句已经执行，这显然与语义不符，因此改为：

i 10 <= <下一个语句开始地址> BM a i [] 0 = <本语句始址>BRL

其中BM操作为当表达式为假时转向<下一个语句开始地址>,BRL是一个一目运算，无条件转向<本语句始址>。

c) 三地址代码序列为：

100 if i <= 10 goto 102

101 goto 106

102 t1 := 4 \* i

103 t2 := a

104 t2[t1] := 0

105 goto 100

106

8.7 为下列基本块构造dag:

d := b\*c

e := a+b

b := b\*c

a := e-d

解：

设所有变量均用名字表示地址。目标代码如下：

（a) MOV #1 x

（b) MOV y x

（c) ADD #1 x

（d) MOV b R1

MUL c R1

ADD a R1

MOV R1 x

（e) MOV b R0

ADD c R0

MOV a R1

DIV R0 R1

MOV e R2

ADD f R2

MUL d R2

SUB R2 R1

MOV R1 x

9.5 下面是应用筛法求2到N之间素数的程序：

begin

read N;

for i := 2 to N do

A[i] := true; /\*置初值\*/

for i := 2 to N\*\*0.5 do /\*运算符\*\*代表幂乘\*/

if A[i] then /\*i是一个素数\*/

for j := 2 \* i to N by i do

A[j] := false /\*j可被i除尽\*/

end

（1）试写出其四元式中间代码，假设对数组A用静态分配存储单元，且下届为0；

（2）作出流图并求出其中的循环；

（3）进行代码外提；

（4）进行强度削弱和删除归纳变量；

解：采用字节地址，两个字节作为一个机器字。

（1）程序的四元式中间代码如下：

B1: read N /\* 置初值 \*/

i := 2

B2: if i > N goto B4 /\* 第一个for语句 \*/

B3: T1 := i

T2 := addr(A) /\* 数组A的基地址 \*/

T1 := 2 \* T1

T2[T1] := true

i := i + 1

goto B2

B4: i := 2

T3 := N \*\* 0.5

T3 := [T3] + 1 /\* [T3]是对T3的值取整 \*/

B5: if i > T3 goto B12

B6: T4 := i

T5 := addr(A)

T4:= 2 \* T4

if T5[T4] goto B8

B7: goto B11

B8: j := 2 \* i

B9: if j > N goto B11 /\* 第三个for语句 \*/

B10: T6 := j

T7 := addr(A)

T6 := 2 \* T6

T7[T6] = false

j := j + i

goto B9

B11: i := i + 1

goto B5

B12:

（2）根据四元式的中间代码，可划分成基本块B1，B2，B3，B4，B5，B6，B7，B8，B9，B10，B11。其程序流图如下：

考察上面的程序流图：

D(B3) = { B1, B2, B3 } 又有 B3 -> B2，因此 B3 -> B2 是一条回边。根据它找到的循环 L1 = { B2, B3 }。

D(B10) = { B1, B2, B4, B5, B6, B9, B10 }，又有 B10 -> B9，所以 B10 -> B9 是一条回边。根据这条回边找到循环 L2 = { B9, B10 }。

D(B11) = { B1, B2, B4, B5, B6, B9, B11 }，又有 B11 -> B5，因此 B11 -> B5 是一条回边。根据这条回边找到循环 L3 = { B11, B9, B10, B8, B7, B6, B5 }

（3）进行代码外提

把在循环中不随循环变化的操作提到循环外的前置结点中，且在基本块中作复写传播和删除无用赋值。结果程序流图如下：

（4）进行强度削弱和删除归纳变量后，其程序流图如下：