

# 第一/二次作业



中国科学技术大学  
University of Science and Technology of China

## 1.2.1

(2.1) 用文字简要描述以下C变量p的类型信息。

```
int (*( *(*p(int x))())[20])(int *y);
```

- 函数指针与指针函数
- p是一个参数为int x的函数，该函数返回一个指针，指向一个无参函数，该函数（无参函数）又返回一个指针，该指针指向一个20个元素的数组，数组的元素类型是一个指针，指向一个参数为int \*y并且返回值为int类型的函数。
- 步骤：
  - 找到需要描述的变量
  - 根据优先级进行描述

## 2.1.1

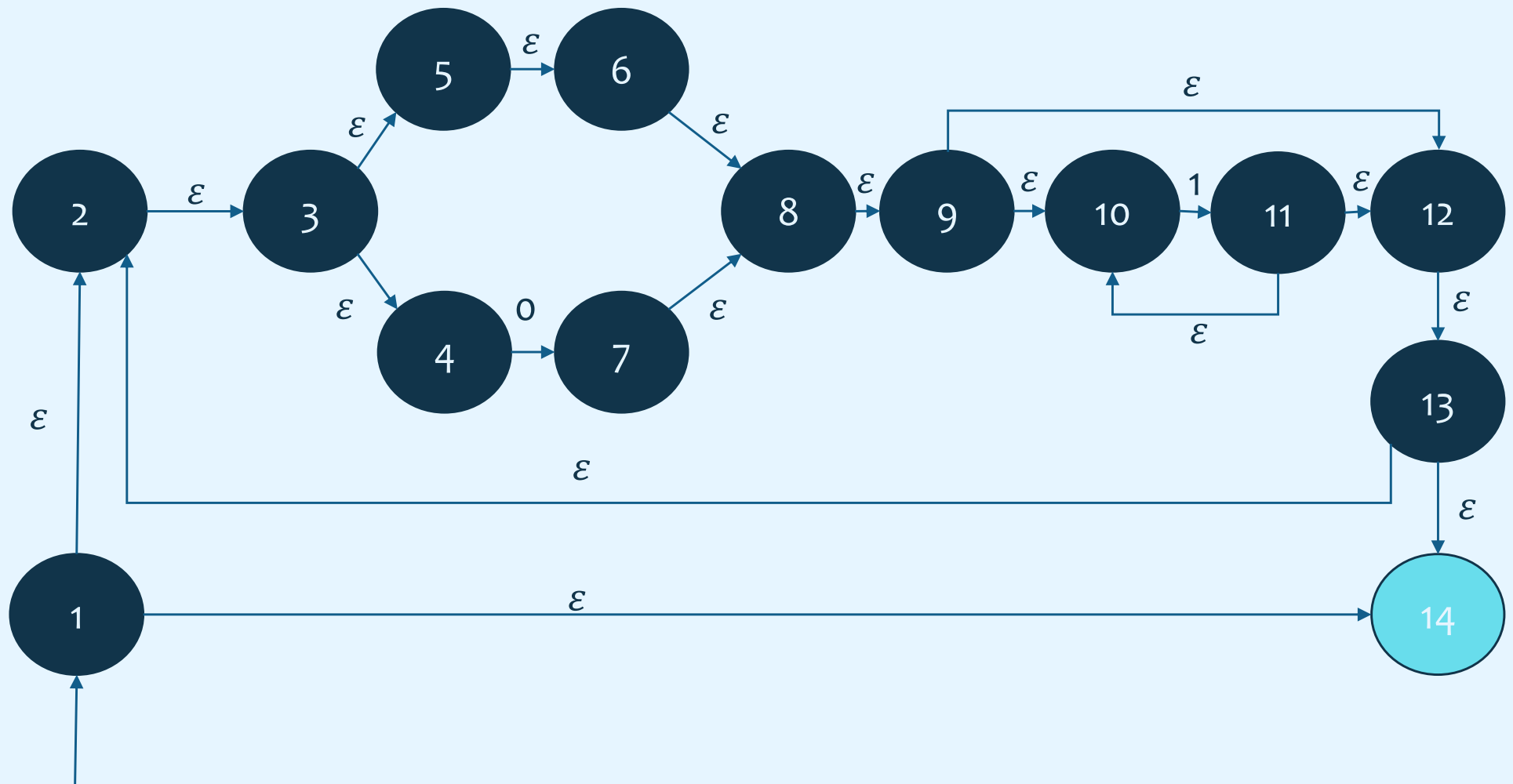
$((\varepsilon|0)1^*)^*$ :任意01串

$0^*10^*10^*10^*$ :有且仅有三个1的01串

$(00|11)^*((01|10)(00|11)^*(01|10)(00|11)^*)^*$ :  
偶数个0、偶数个1组成的01串

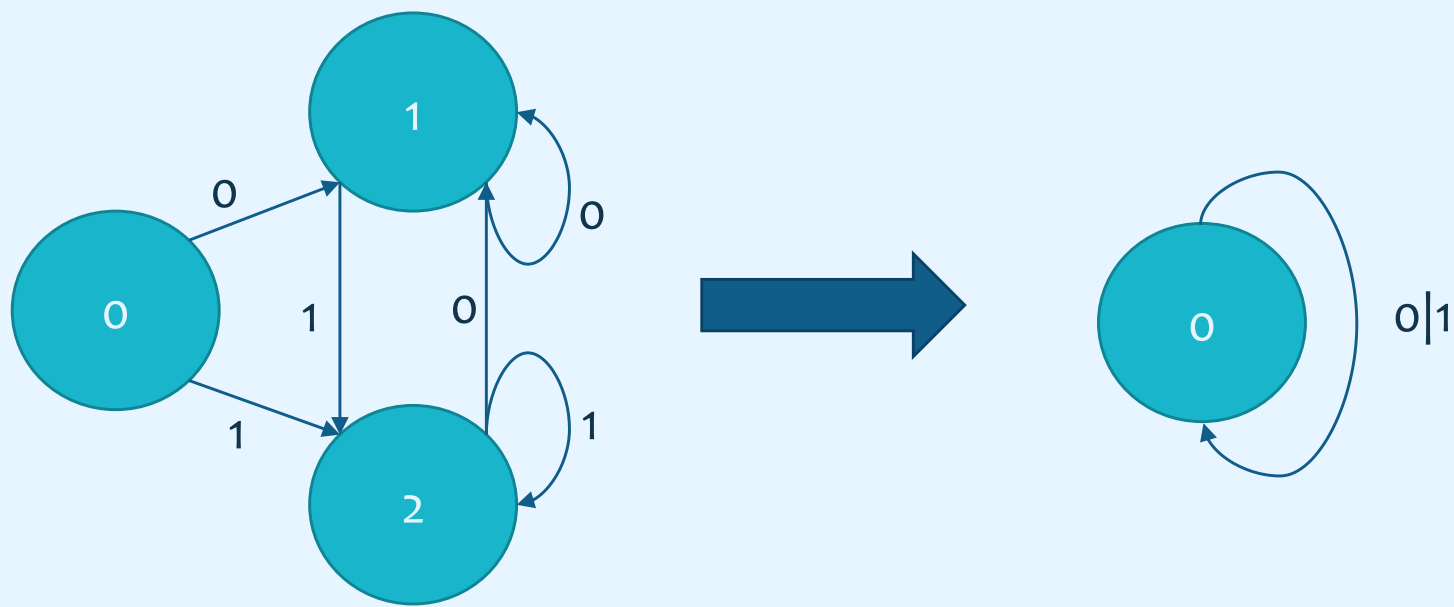


## 2.1.2

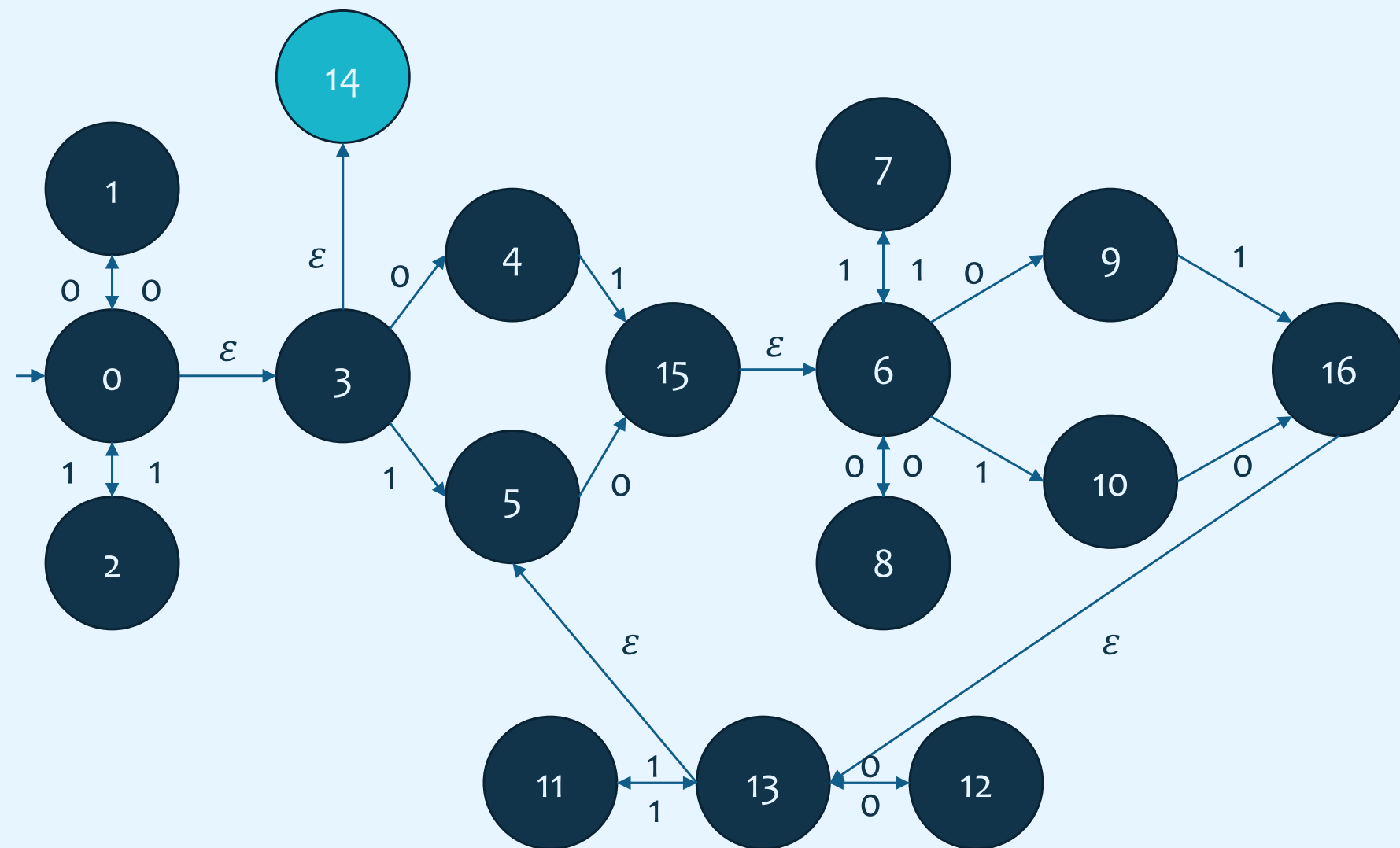


## 2.1.2

	状态	0	1
0	1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 13, 14	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14	1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14
1	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14	1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14
2	1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14	1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14



## 2.1.3

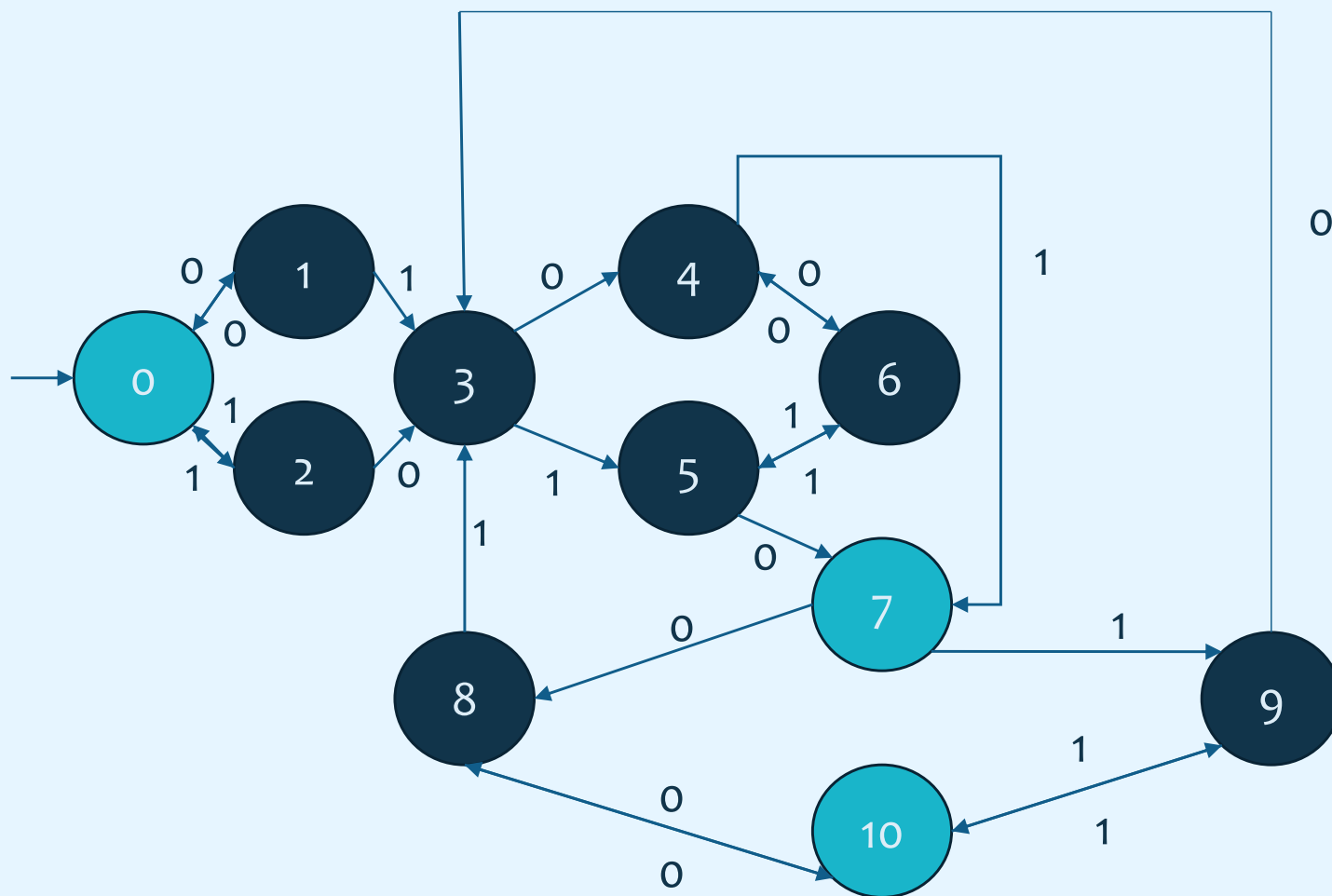


## 2.1.3

	状态	0	1
0	0, 3, 14	1, 4	2, 5
1	1, 4	0, 3, 14	15, 6
2	2, 5	15, 6	0, 3, 14
3	15, 6	8, 9	7, 10
4	8, 9	6	16, 13, 3, 14
5	7, 10	16, 13, 3, 14	6
6	6	8, 9	7, 10
7	16, 13, 3, 14	11, 4	12, 5
8	11, 4	13, 3, 14	15, 6
9	12, 5	15, 6	13, 3, 14
10	13, 3, 14	11, 4	12, 5



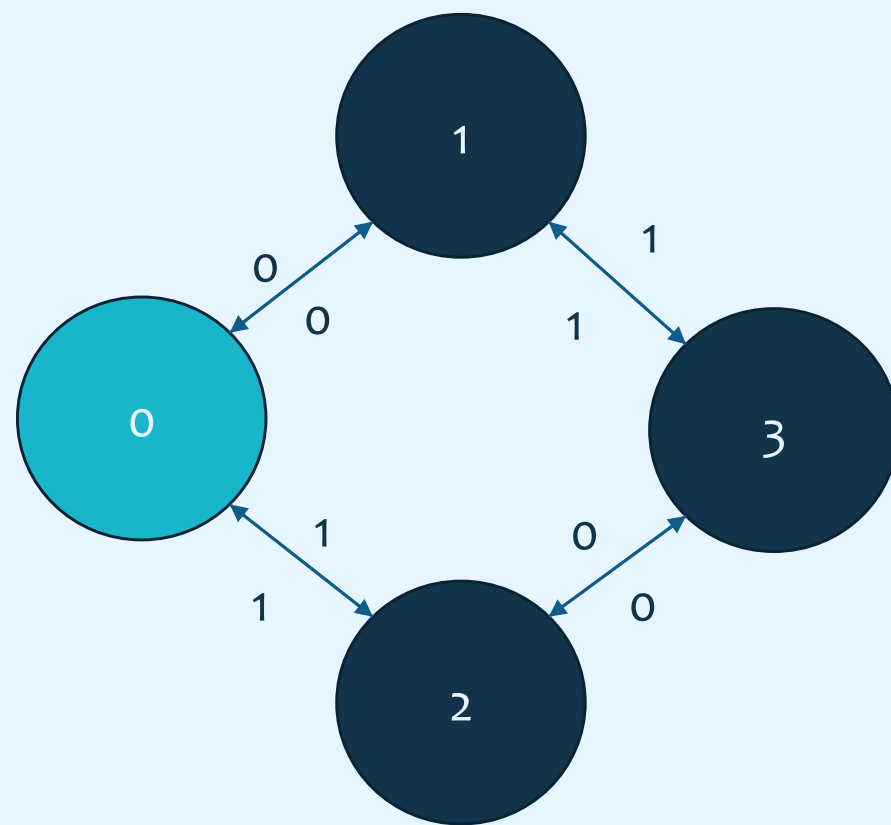
## 2.1.3





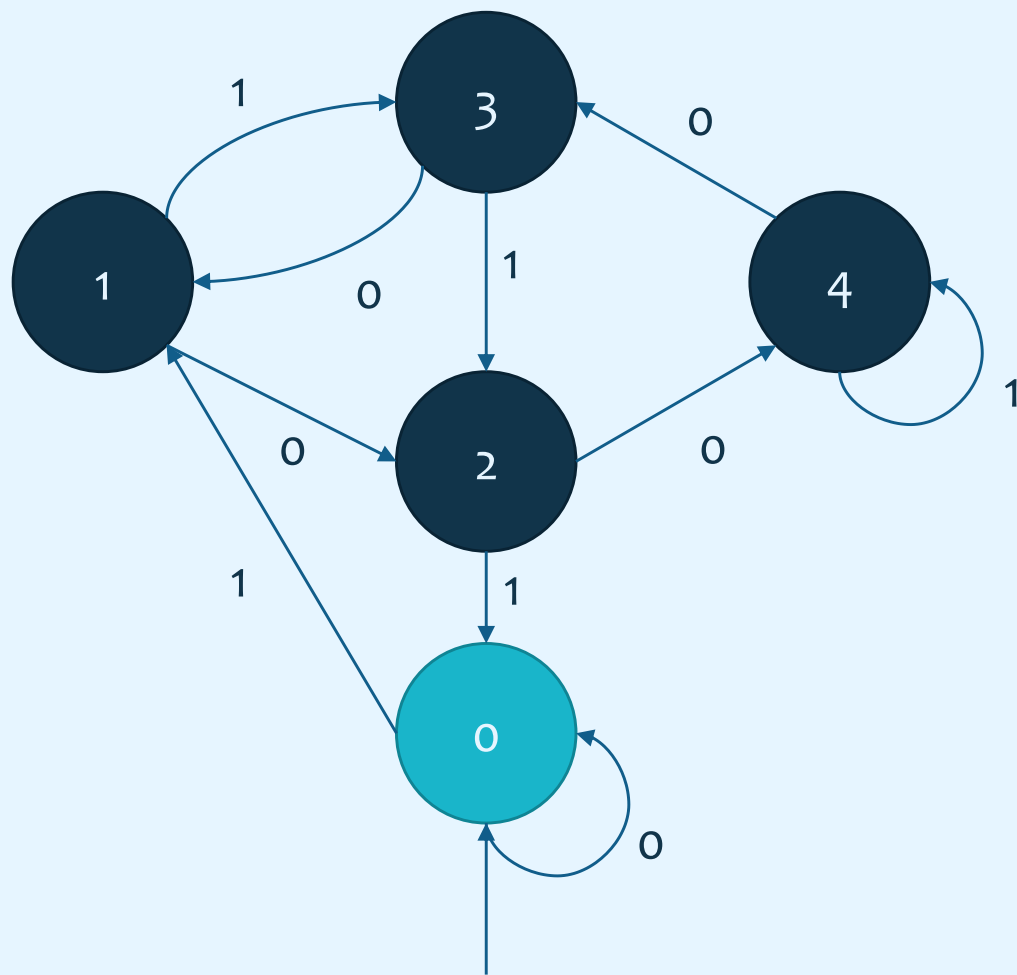
## 2.1.3

- $I_0 = \{0, 7, 10\}$
- $I_1 = \{1, 2, 3, 4, 5, 6, 8, 9\}$
- $I_2 = \{1, 5, 8\}$
- $I_3 = \{2, 3, 4, 6, 9\}$
- $I_4 = \{2, 4, 9\}$
- $I_5 = \{3, 6\}$



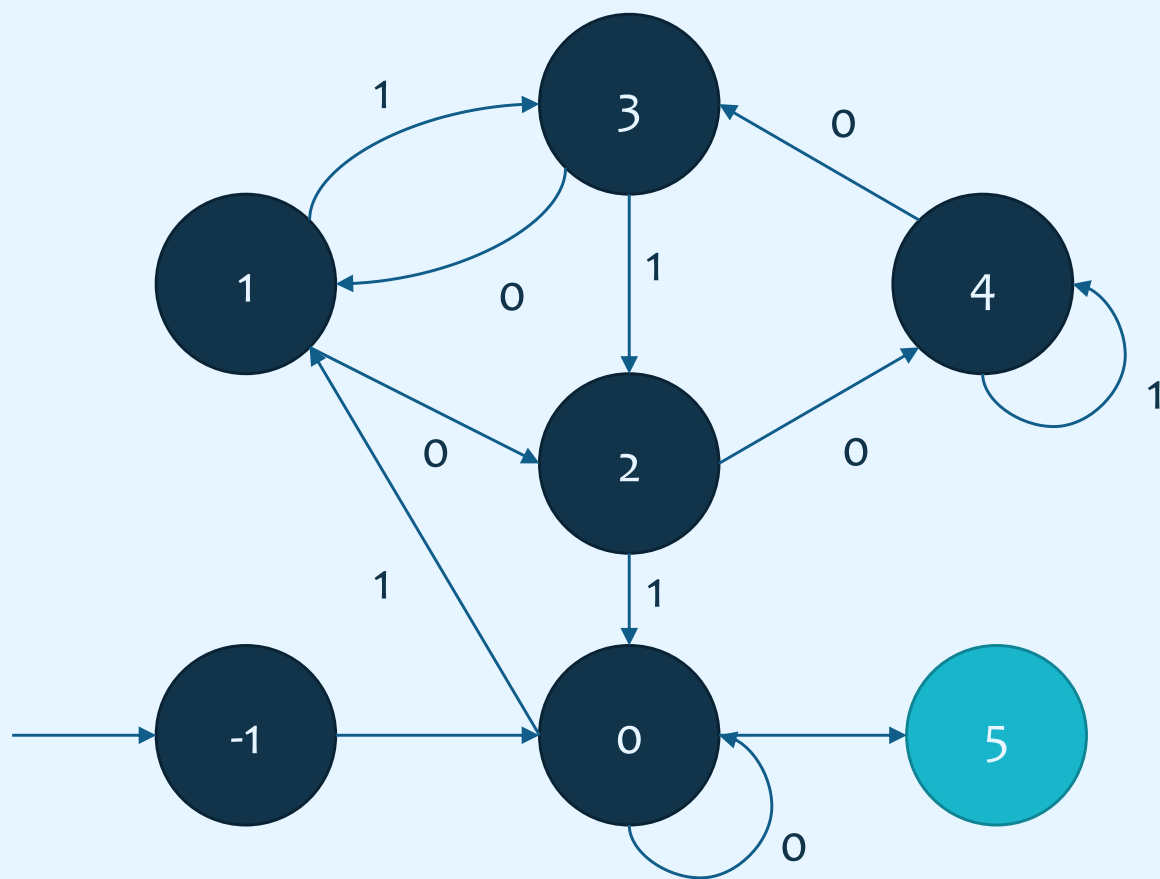
## 2.2.1

构造DFA:5个状态, 代表被5除余s



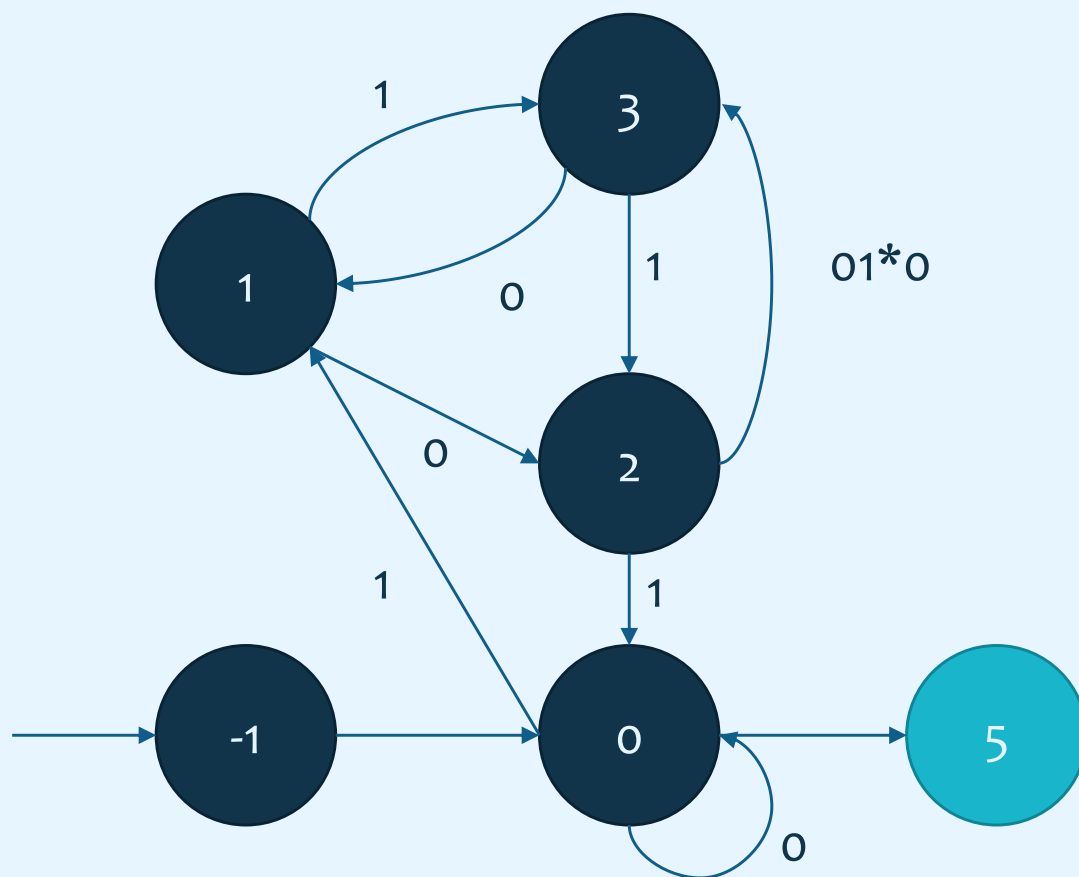
## 2.2.1

添开始，结束状态



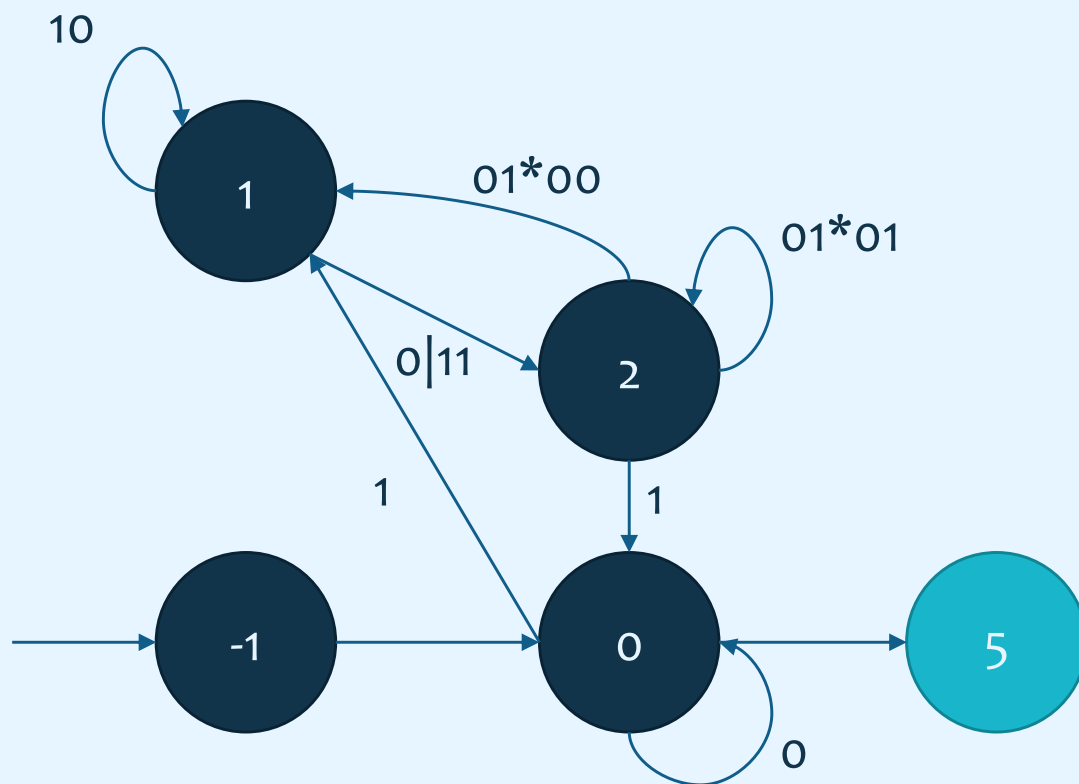
## 2.2.1

删去4



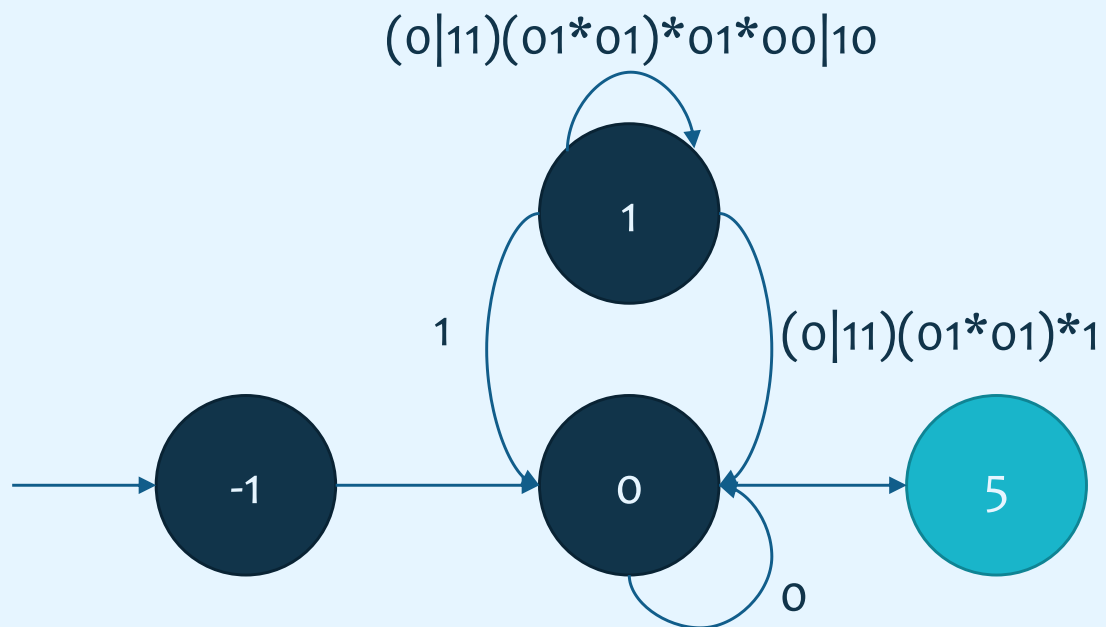
## 2.2.1

删去3



## 2.2.1

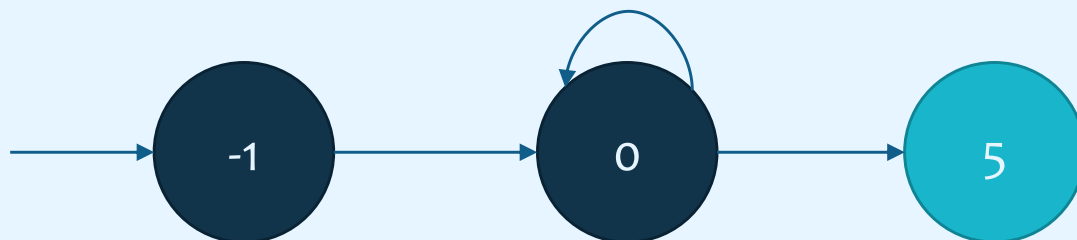
删去2



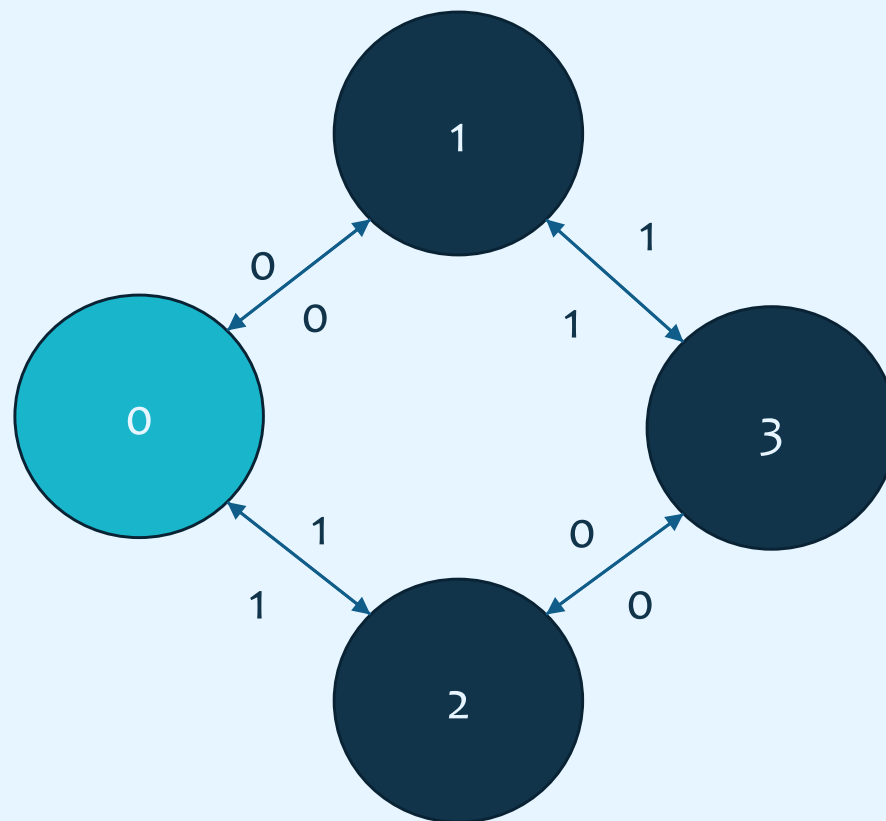
## 2.2.1

删去1

$(0|1((0|11)(01^*01)^*01^*00|10)^*(0|11)(01^*01)^*1)$



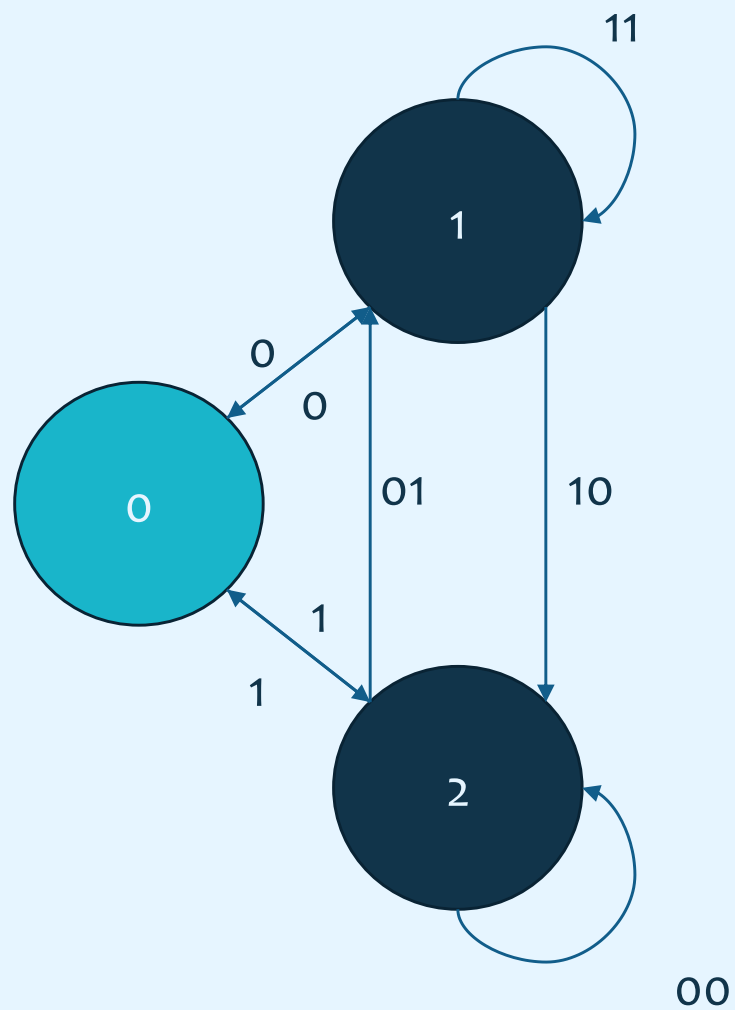
## 2.2.2





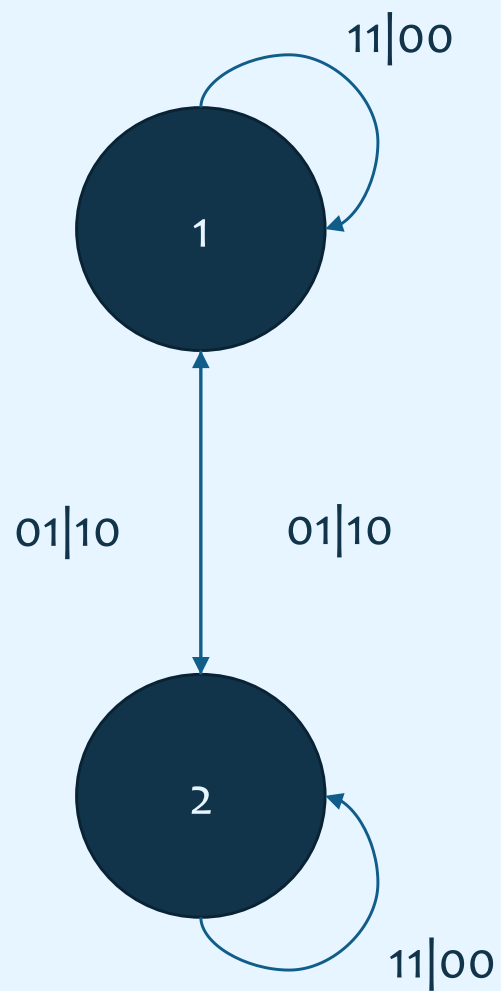
## 2.2.2

删去3



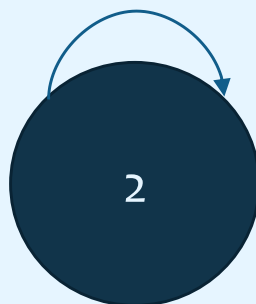
## 2.2.2

删去0



## 2.2.2

删去<sub>1</sub>

$$(00|11)^*((01|10)(00|11)^*(01|10)(00|11)^*)^*$$


# 谢谢!

结束页



中国科学技术大学  
University of Science and Technology of China

# 第三次作业

施海涵



中国科学技术大学

University of Science and Technology of China

## 3.1

(1) 以下是C语言数组变量声明初始化的示例：

```
int f[][8][5] = { {{1},{2}},{3},{4},{5},{6},{7},8, };
```

填写合适的数字，补全数组变量f 声明[]处的空白！

并给出初始化描述中数值1~8在数组f中的下标。

1 0 0 0 0	2 0 0 0 0	}	8x5
.....	.....		
3 0 0 0 0	0 0 0 0 0	}	8x5
.....	.....		
4 5 6 7 8	0 0 0 0 0	}	8x5
.....	.....		

[0][0][0]  
[0][1][0]  
[1][0][0]  
[2][0][0]  
[2][0][1]  
[2][0][2]  
[2][0][3]  
[2][0][4]

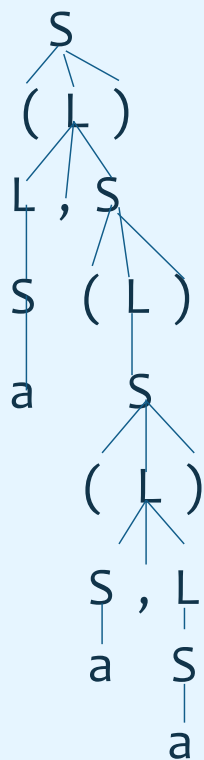
不以左括号开始的初始化项，此后直接进行填充。

## 3.2

最左推导:

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

分析树:



最右推导:

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

分析树: 略

## 3.3

(1)

- $S \Rightarrow aSbS \Rightarrow abSaSbS \Rightarrow abaSbS \Rightarrow ababS \Rightarrow abab$
- $S \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSbS \Rightarrow ababS \Rightarrow abab$

(2) (不唯一)

- $S \Rightarrow aSbS \Rightarrow aSb \Rightarrow abSaSb \Rightarrow abSab \Rightarrow abab$
- $S \Rightarrow aSbS \Rightarrow aSbaSbS \Rightarrow aSbaSb \Rightarrow aSbab \Rightarrow abab$

(3)略

(4) ab成对出现的ab串



## 3.4

- 给出语句1所对应的汇编代码;

`int *p = &i; //语句1`

```
lea    rax, [rbp-12]
mov     QWORD PTR [rbp-8], rax
```

`((++(*((p)++)))`

`(p)`                    `mov rax, QWORD PTR [rbp-8]`

`((p)++)`

```
lea    rdx, [rax+4]
mov     QWORD PTR [rbp-8], rdx
```

`*((p)++)`

```
mov     edx, DWORD PTR [rax]
```

`...`

```
add     edx, 1
mov     DWORD PTR [rax], edx
```



# 谢谢!

结束页



中国科学技术大学  
University of Science and Technology of China

# HW4参考

### (1) 3.1

**G:**

**$S \rightarrow (L) \mid a$**

**$L \rightarrow L, S \mid S$**

构造两个版本的递归下降语法分析程序。

消除直接左递归之后如下

$S \rightarrow (L) \mid a$

$L \rightarrow S L'$

$L' \rightarrow , S L' \mid \epsilon$

预测分析表

	(	)	a	,	\$
S	$S \rightarrow (L)$		$S \rightarrow a$		
L	$L \rightarrow S L'$		$L \rightarrow S L'$		
L'		$L' \rightarrow \epsilon$		$L' \rightarrow , S L'$	

$\text{FIRST}(L') = \{ , , \epsilon \}$

$\text{FIRST}(L) = \{ ( , a \}$

$\text{FIRST}(S) = \{ ( , a \}$

$\text{FOLLOW}(S) = \{ \$ , , ) \}$

$\text{FOLLOW}(L) = \{ ) \}$

$\text{FOLLOW}(L') = \text{FOLLOW}(L) = \{ ) \}$

## (1) 3.1

```
void match(token t) {
    if (lookhead == t) {
        lookhead = next_token();
    } else {
        error();
    }
}

void S() {
    if (lookhead == '(') {
        match('(');
        L();
        match(')');
    } else if (lookhead == 'a') {
        match('a');
    } else {
        error();
    }
}

void L() {
    if (lookhead == '(' || lookhead == 'a') {
        S();
        L();
    } else {
        error();
    }
}
```

```
/*
 * 版本一,  $\epsilon$ 与任意符号匹配
 * 直接返回
 */
void L`() {
    if (lookhead == ',') {
        match(',');
        S();
        L`();
    } else {
        return;
    }
}

/*
 * 版本二, 查看lookhead是否属于FOLLOW(L`)
 * 是, 直接返回
 * 否则进入错误处理
 */
void L`() {
    if (lookhead == ',') {
        match(',');
        S();
        L`();
    } else if (lookhead == ')') {
        return;
    } else {
        error();
    }
}
```

## (2) 3.11

G:

$S \rightarrow aBS \mid bAS \mid \varepsilon$

$A \rightarrow bAA \mid a$

$B \rightarrow aBB \mid b$

$\text{FIRST}(B) = \{ a, b \}$

$\text{FIRST}(A) = \{ b, a \}$

$\text{FIRST}(S) = \{ a, b, \varepsilon \}$

$\text{FOLLOW}(S) = \{ \$ \}$

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

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

LL(1)分析表

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

产生语言:  $L(G)$ 为a和b个数相等的 (任意) ab串。

(1)、a和b个数相等    (2)、任意ab串

### (3) 3.17

G:

$S \rightarrow (L) \mid a$

$L \rightarrow L, S \mid S$

$\text{FIRST}(L) = \{ (, a \}$

$\text{FIRST}(S) = \{ (, a \}$

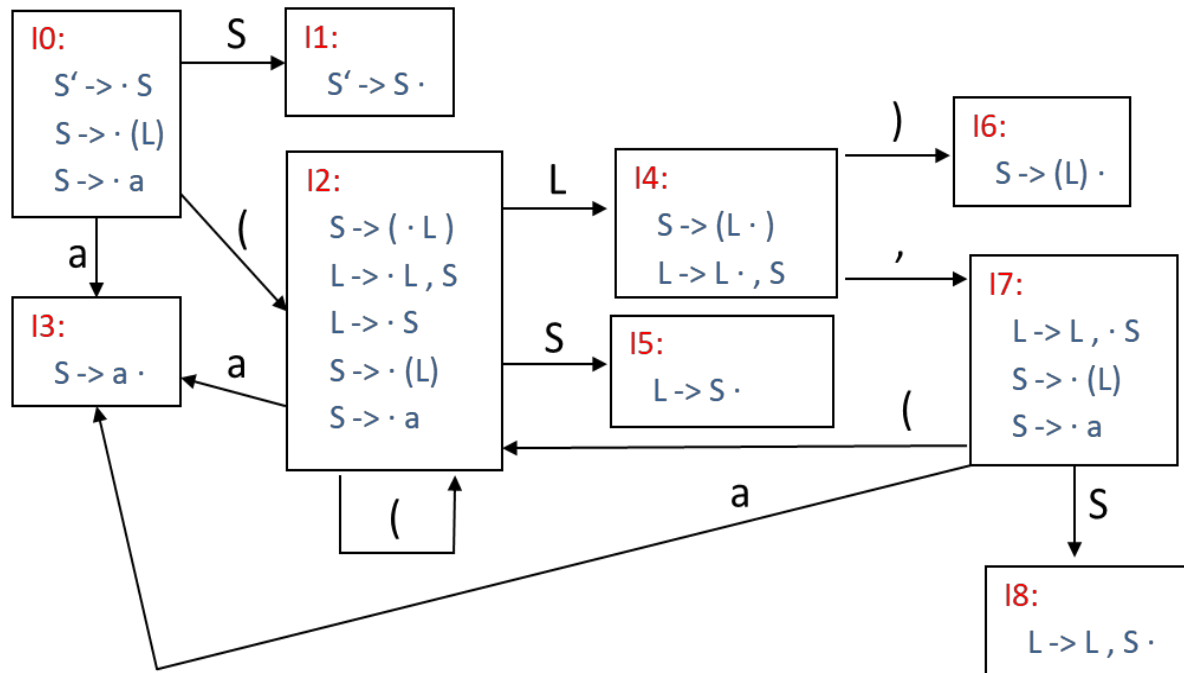
$\text{FOLLOW}(S) = \{ \$, ), , , \}$

$\text{FOLLOW}(L) = \{ ), , , \}$

拓展文法G:

- (0)  $S' \rightarrow S$
- (1)  $S \rightarrow (L)$
- (2)  $S \rightarrow a$
- (3)  $L \rightarrow L, S$
- (4)  $L \rightarrow S$

识别文法G活前缀的DFA



# SLR(1)分析表

状态	action					goto	
	(	)	a	,	\$	S	L
0	s2		s3			1	
1					acc		
2	s2		s3			5	4
3		r2		r2	r2		
4		s6		s7			
5		r4		r4			
6		r1		r1	r1		
7	s2		s3			8	
8		r3		r3			

FOLLOW(s)





### (3) 3.21

**G:**

**$S \rightarrow AaAb \mid BbBa$**

**$A \rightarrow \varepsilon$**

**$B \rightarrow \varepsilon$**

(a) 证明文法G是LL(1)文法，但不是SLR(1)文法。

(b) 证明所有LL(1)文法都是LR(1)文法。

(a) 证:  $\text{FIRST}(AaAb) = \{ a \}$ ,  $\text{FIRST}(BbBa) = \{ b \}$ , 即  $\text{FIRST}(AaAb) \cap \text{FIRST}(BbBa)$  为空, 且  $\varepsilon$  不属于  $\text{FIRST}(AaAb)$ 、 $\text{FIRST}(BbBa)$ , 所以文法G是LL(1)文法。

### (3) 3.21

拓展文法G:

(0)  $S' \rightarrow S$

(1)  $S \rightarrow AaAb$

(2)  $S \rightarrow BbBa$

(3)  $A \rightarrow \epsilon$

(4)  $B \rightarrow \epsilon$

$FIRST(B) = \{ \epsilon \}$

$FIRST(A) = \{ \epsilon \}$

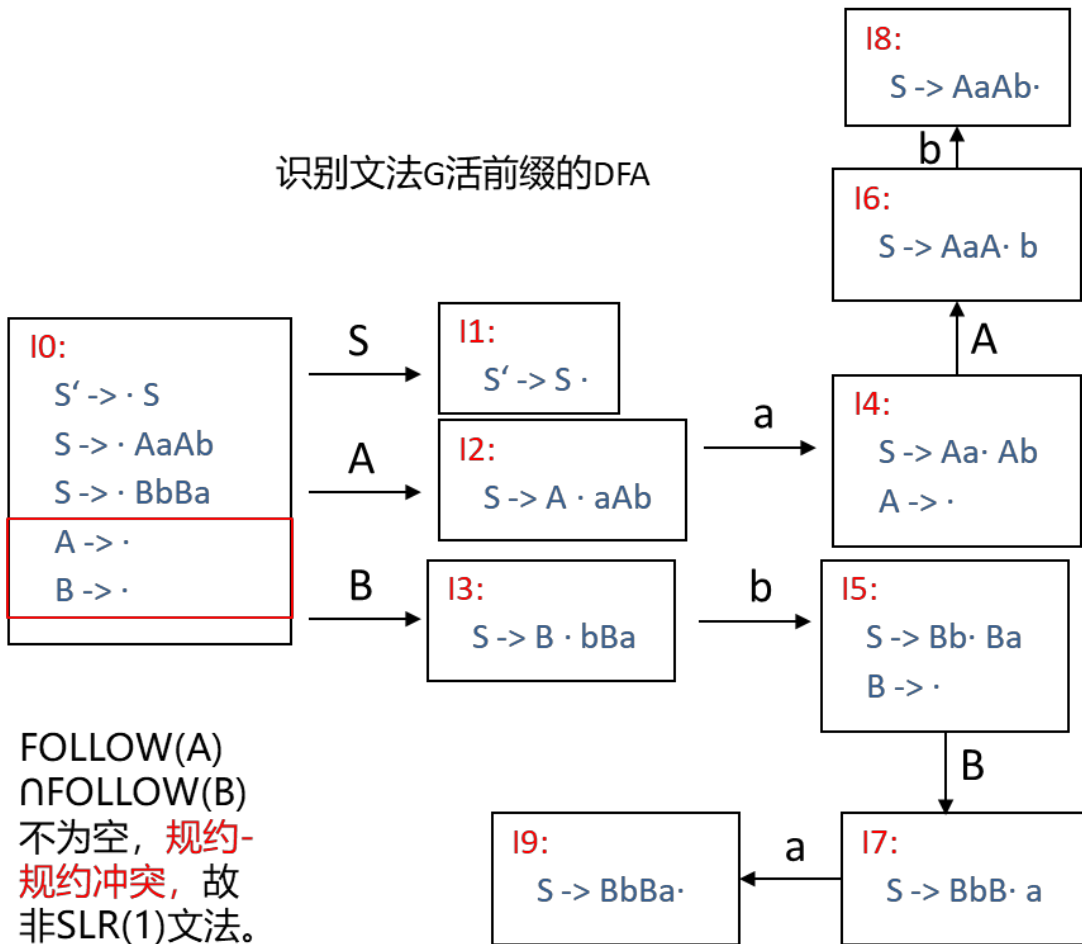
$FIRST(S) = \{ a, b \}$

$FOLLOW(S) = \{ \$ \}$

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


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

识别文法G活前缀的DFA



### (3) 3.21

(b) 证明所有LL(1)文法都是LR(1)文法。

 考虑证明：若一个文法不是LR(1)文法，则它一定不是LL(1)文法。

若一个文法不是LR(1)文法，则它存在分析动作的冲突（移进-规约和归约-归约冲突），也就是说，该文法存在二义性的最左推导，于是不满足LL(1)文法的要求，所以该文法不是LL(1)文法。

即说明所有LL(1)文法都是LR(1)文法。

另外，LL(1)文法和LR(1)文法对比，两者都是从左往右看，并且只看下一个符号，再进行相应的动作。满足LL(1)文法的语言，对它进行LL(1)分析的每一步实际上也对应着LR(1)分析的每一步。但是，LL(1)文法看到一个符号后，就提前选择对应的产生式进行推导，而LR(1)文法有了句柄的概念后，拥有了更多的信息，可以在看到一个符号时，根据搜索符进行移进或规约动作（它可以看完了句柄再选择产生式），也就是说，LR(1)文法是大于LL(1)文法的，即所有LL(1)文法都是LR(1)文法。

(3) 3.22

**G:**

**$S \rightarrow Aa \mid bAc \mid dc \mid bda$**

**$A \rightarrow d$**

证明上述文法是LALR(1)文法，但不是SLR(1)文法。

### (3) 3.22

拓展文法G:

(0)  $S' \rightarrow S$

(1)  $S \rightarrow Aa$

(2)  $S \rightarrow bAc$

(3)  $S \rightarrow dc$

(4)  $S \rightarrow bda$

(5)  $A \rightarrow d$

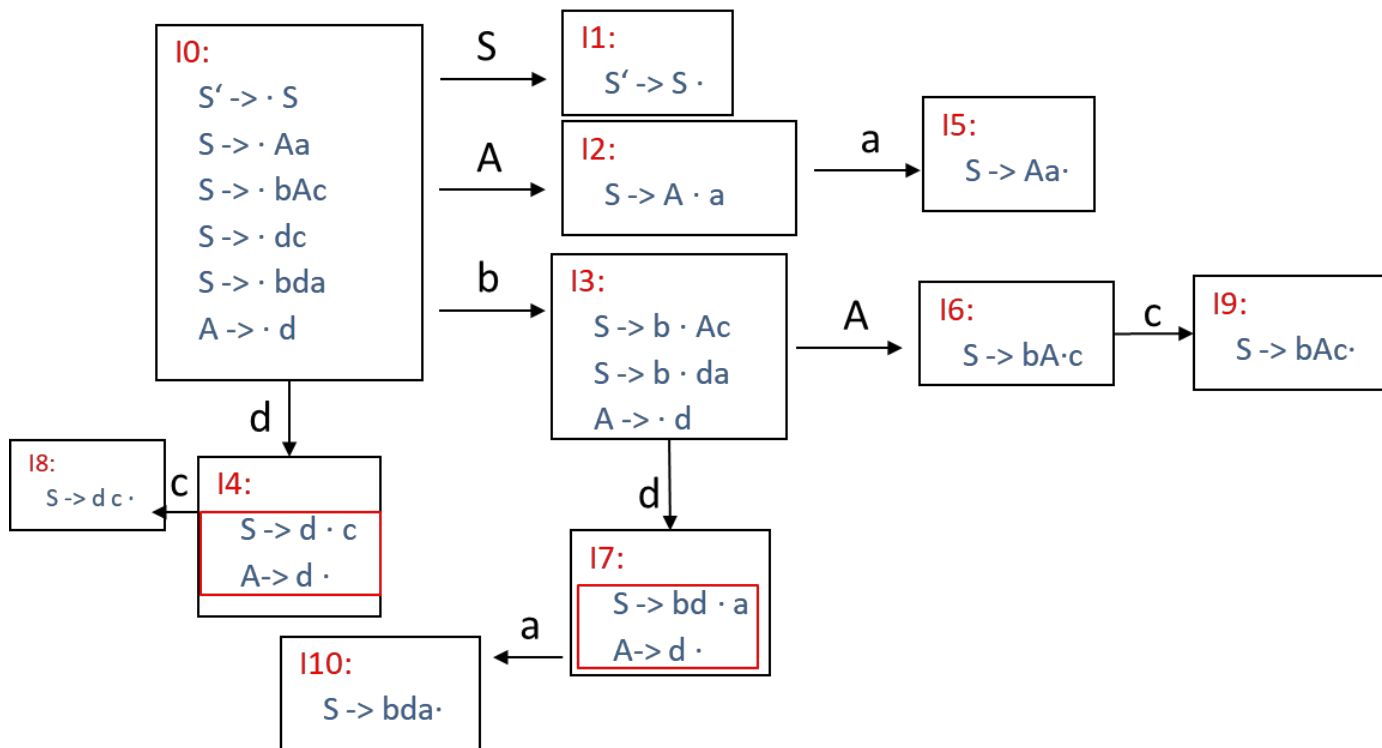
$\text{FIRST}(A) = \{ d \}$

$\text{FIRST}(S) = \{ d, b \}$

$\text{FOLLOW}(S) = \{ \$ \}$

$\text{FOLLOW}(A) = \{ a, c \}$

识别文法G活前缀的DFA(SLR(1))



I4和I7移进-规约冲突,  
故非SLR(1)文法。

### (3) 3.22

拓展文法G:

(0)  $S' \rightarrow S$

(1)  $S \rightarrow Aa$

(2)  $S \rightarrow bAc$

(3)  $S \rightarrow dc$

(4)  $S \rightarrow bda$

(5)  $A \rightarrow d$

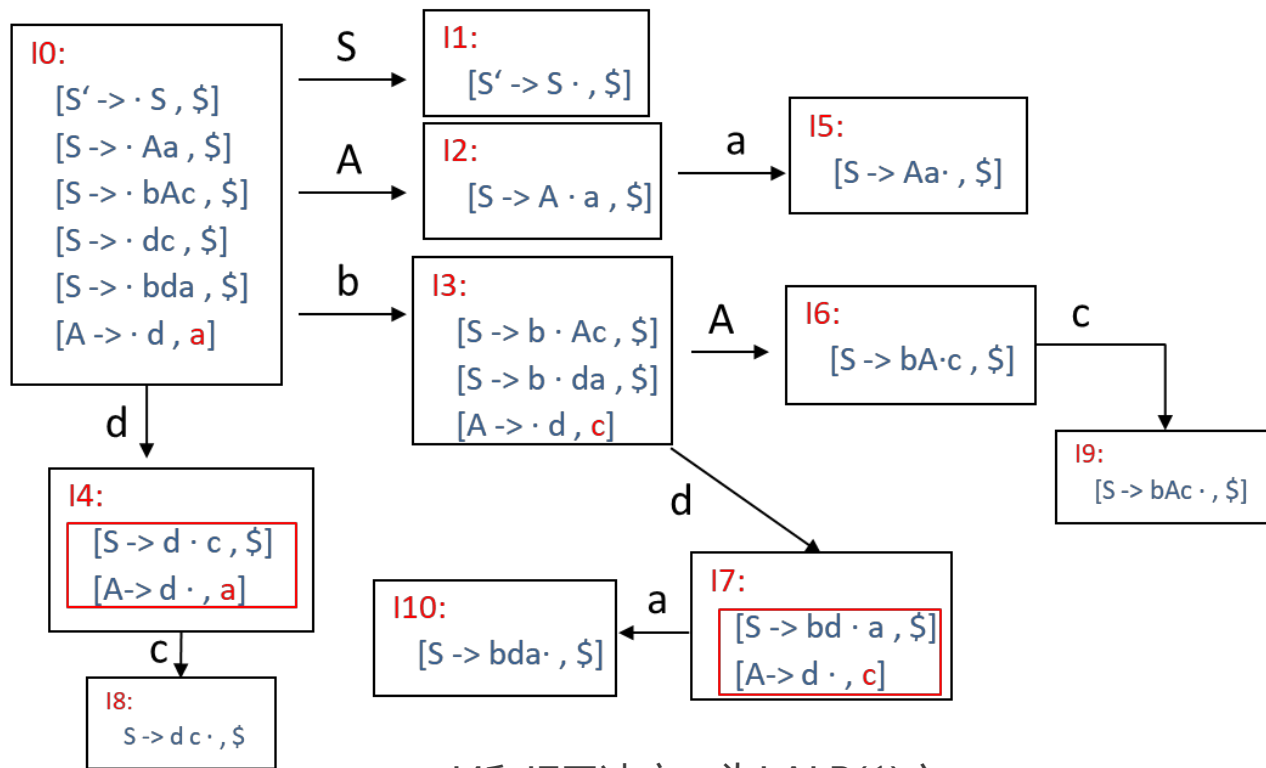
$\text{FIRST}(A) = \{ d \}$

$\text{FIRST}(S) = \{ d, b \}$

$\text{FOLLOW}(S) = \{ \$ \}$

$\text{FOLLOW}(A) = \{ a, c \}$

识别文法G活前缀的DFA(LALR(1))



I4和I7无冲突, 为LALR(1)文法。

(3) 3.24

**G:**

**$S \rightarrow Aa \mid bAc \mid Bc \mid bBa$**

**$A \rightarrow d$**

**$B \rightarrow d$**

证明文法是LR(1)文法，但不是LALR(1)文法。

### (3) 3.24

拓展文法G:

(0)  $S' \rightarrow S$

(1)  $S \rightarrow Aa$

(2)  $S \rightarrow bAc$

(3)  $S \rightarrow Bc$

(4)  $S \rightarrow bBa$

(5)  $A \rightarrow d$

(6)  $B \rightarrow d$

$\text{FIRST}(A) = \{ d \}$

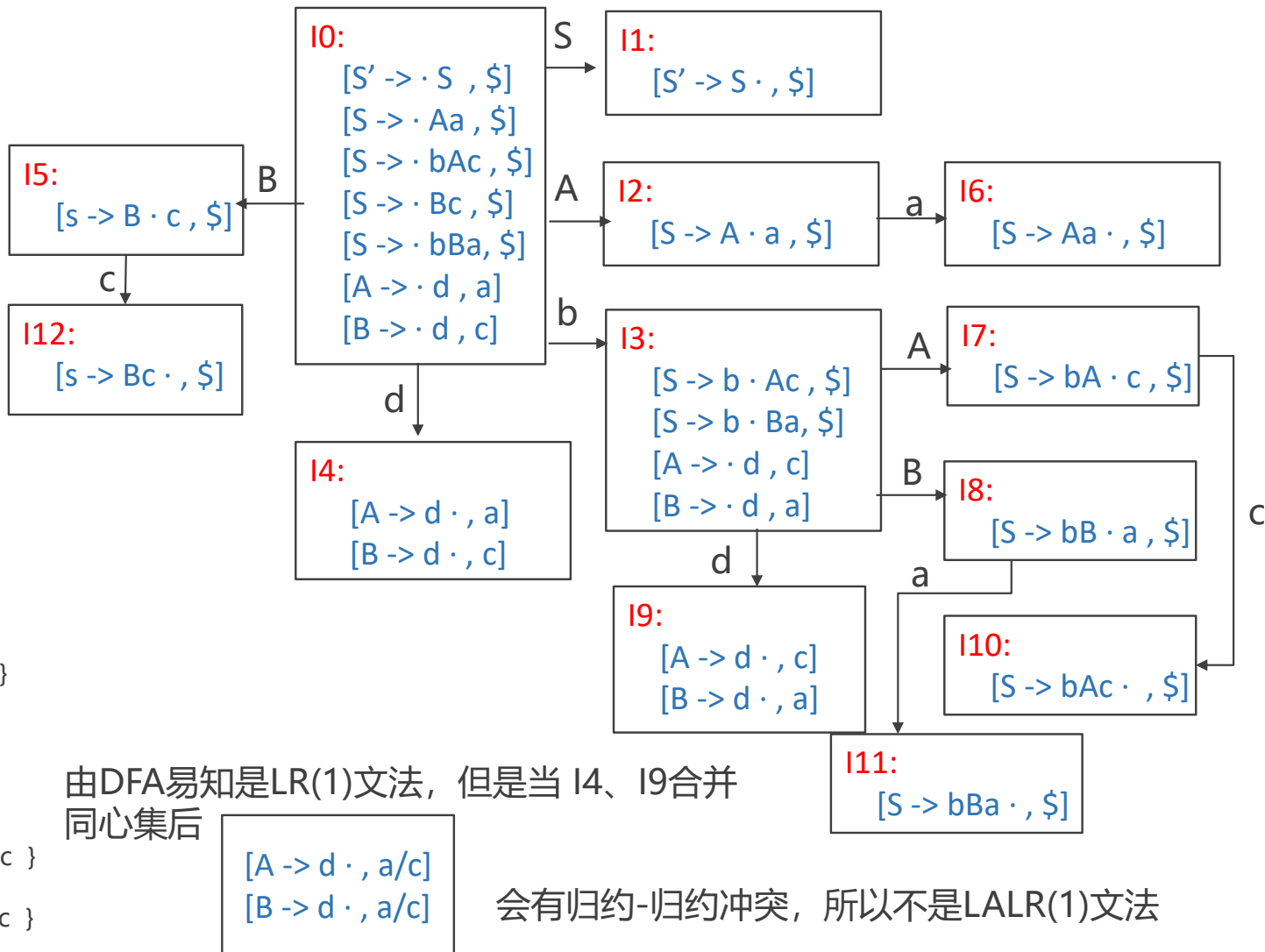
$\text{FIRST}(S) = \{ d, b \}$

$\text{FIRST}(B) = \{ d \}$

$\text{FOLLOW}(S) = \{ \$ \}$

$\text{FOLLOW}(A) = \{ a, c \}$

$\text{FOLLOW}(B) = \{ a, c \}$





(3) 3.25

**G:**

**$L \rightarrow MLb \mid a$**

**$M \rightarrow \epsilon$**

给出所有移进-规约冲突的规范LR(1)项目集，以说明该文法不是LR(1)的。

### (3) 3.25

拓展文法G:

(0)  $L' \rightarrow L$

(1)  $L \rightarrow MLb$

(2)  $L \rightarrow a$

(3)  $M \rightarrow \varepsilon$

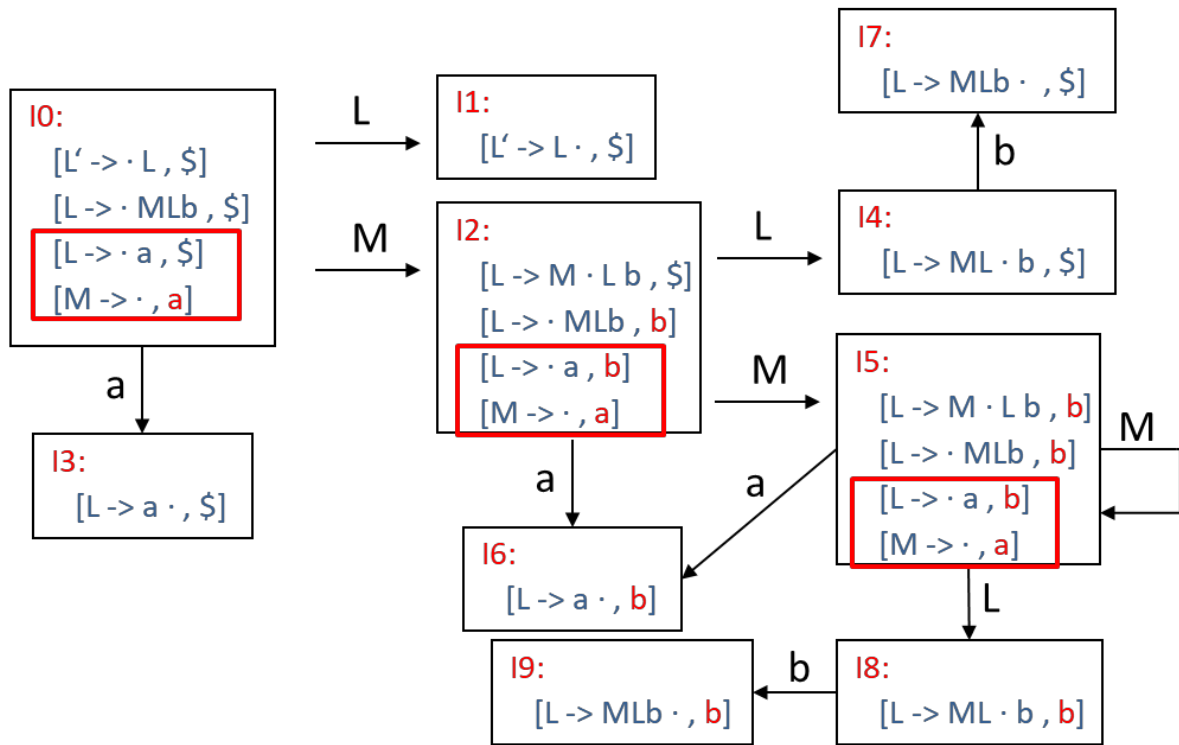
$FIRST(M) = \{ \varepsilon \}$

$FIRST(L) = \{ a \}$

$FOLLOW(L) = \{ \$, b \}$

$FOLLOW(M) = \{ a \}$

识别文法G活前缀的DFA(LR(1))



I0、I2和I5有移进-规约冲突。

(4) 

**G:**

**$S \rightarrow aS$**

**$S \rightarrow A$**

**$A \rightarrow aAb$**

**$A \rightarrow \epsilon$**

构造LR(1)分析表。

(4)

拓展文法G:

(0)  $S' \rightarrow S$

(1)  $S \rightarrow a S$

(2)  $S \rightarrow A$

(3)  $A \rightarrow a A b$

(4)  $A \rightarrow \varepsilon$

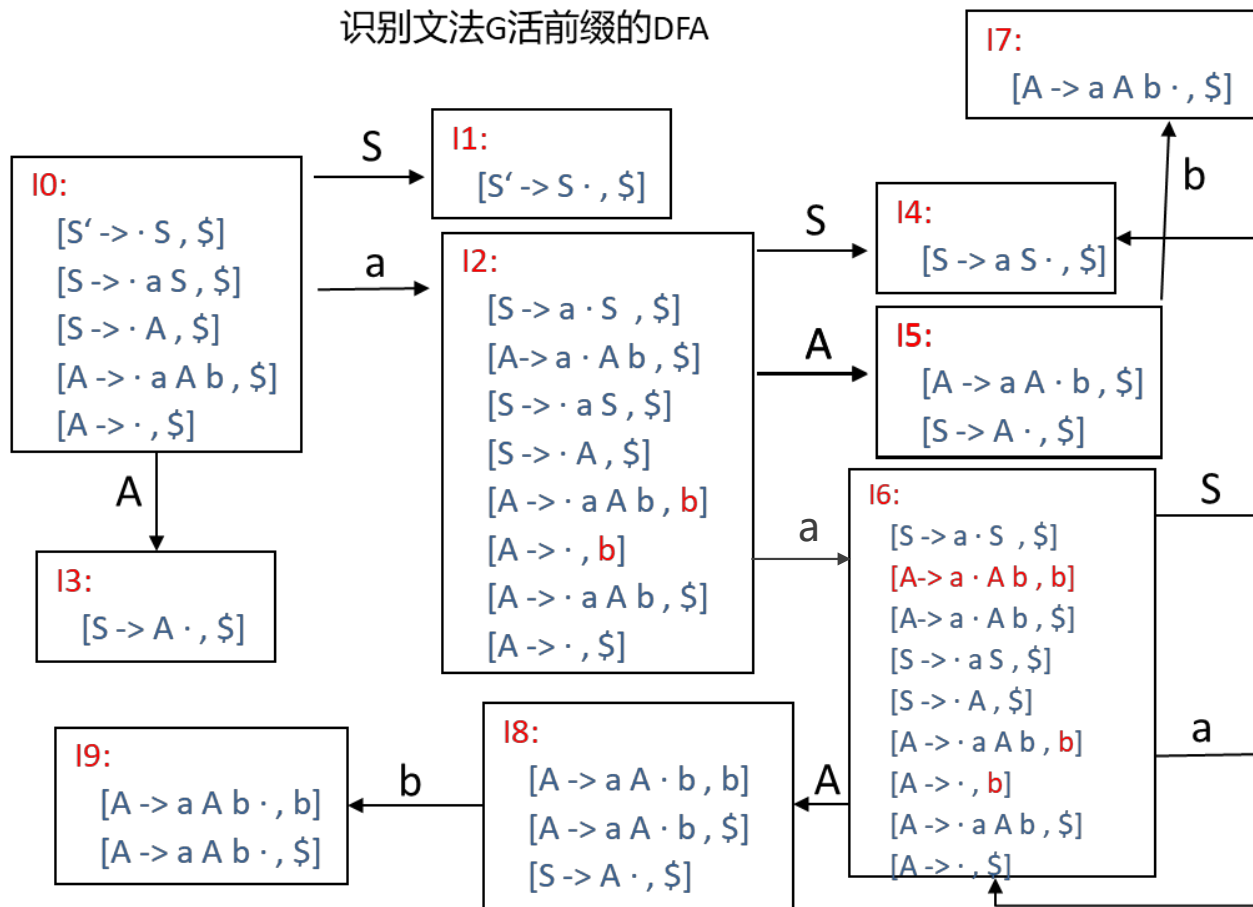
$\text{FIRST}(A) = \{ \varepsilon, a \}$

$\text{FIRST}(S) = \{ \varepsilon, a \}$

$\text{FOLLOW}(S) = \{ \$ \}$

$\text{FOLLOW}(A) = \{ \$, b \}$

识别文法G活前缀的DFA



## LR(1)分析表

状态	action			goto	
	a	b	\$	S	A
0	S2		r4	1	3
1			acc		
2	s6	r4	r4	4	5
3			r2		
4			r1		
5		s7	r2		
6	s6	r4	r4	4	8
7			r3		
8		s9	r2		
9		r3	r3		

(5)

G:

$S \rightarrow BB$

$B \rightarrow bB$

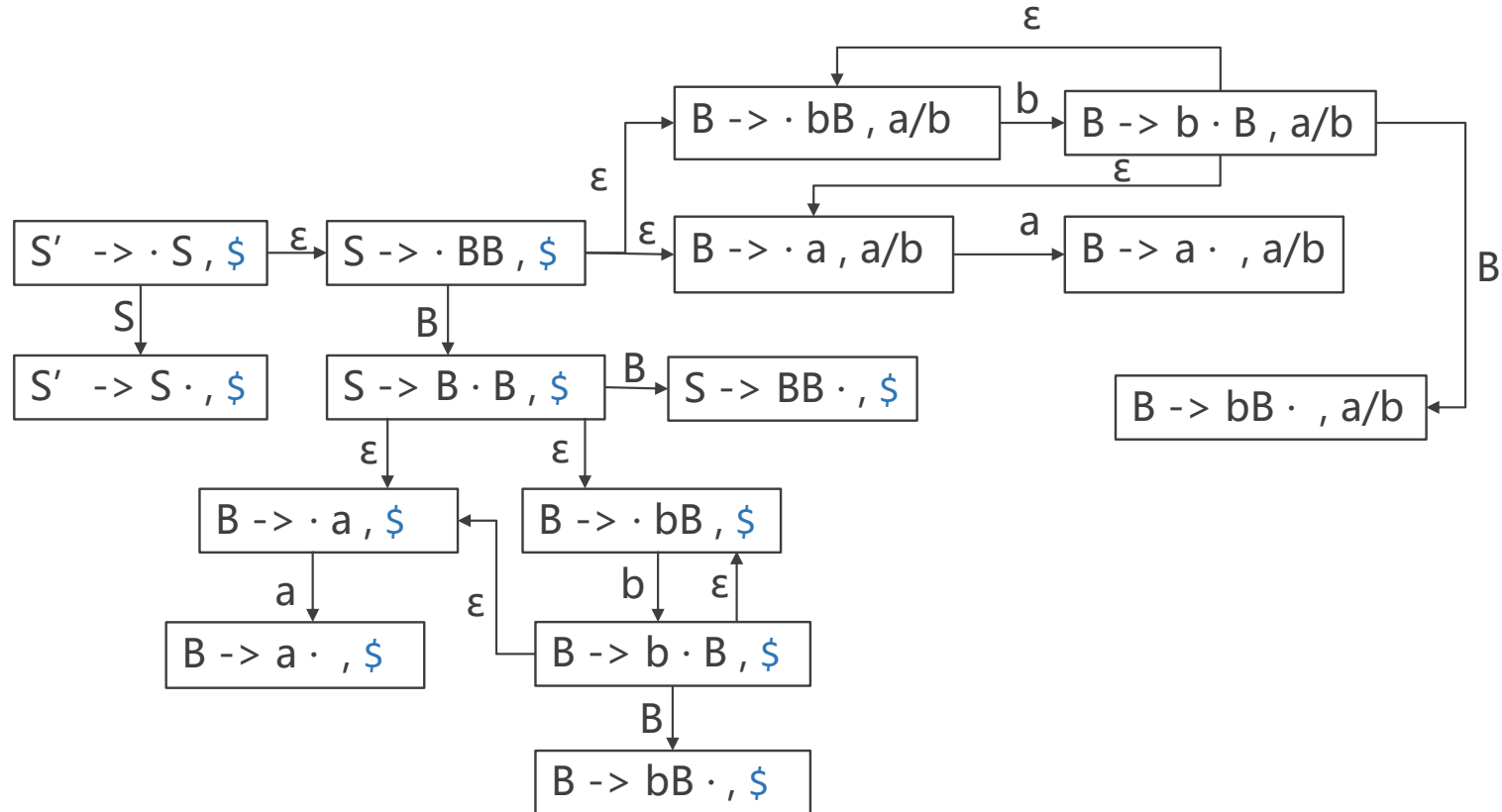
$B \rightarrow a$

扩充文法：  
 $S' \rightarrow S$   
 $S \rightarrow BB$   
 $B \rightarrow bB$   
 $B \rightarrow a$

给出接受文法的活前缀且以LR(1)项目为状态的一个NFA。

(5)

NFA



# 谢谢！



中国科学技术大学  
University of Science and Technology of China



# HW05

(1) 习题4.9, 并分别给出 (a) 和 (b) 两个语法制导定义的属性栈代码实现 (非yacc代码)。

文法:

easy.

$S \rightarrow L.L \mid L$

$L \rightarrow LB \mid B$

$B \rightarrow 0 \mid 1$

$S \rightarrow L_1.L_2$

$S.val = L_1.val + L_2.val / 2^{L_2.length}$

$S \rightarrow L$

$S.val = L.val$

$L \rightarrow L_1B$

$L.val = L_1.val \times 2 + B.val; \quad L.length = L_1.length + 1$

$L \rightarrow B$

$L.val = B.val; \quad L.length = 1$

$B \rightarrow 0$

$B.val = 0$

$B \rightarrow 1$

$B.val = 1$

## 属性栈代码

- $S \rightarrow L1.L2$        $stack[ntop].val = stack[top-2].val + stack[top].val / 2^{stack[top].length}$
- $L \rightarrow L1B$        $stack[ntop].val = stack[top-1].val * 2 + stack[top].val$   
                          $stack[ntop].length = stack[top-1].length + 1$
- $L \rightarrow B$        $stack[ntop].length = 1$
- $B \rightarrow 0$        $stack[ntop].val = 0$
- $B \rightarrow 1$        $stack[ntop].val = 1$

## (b)——翻译方案 *hard.*

- $S \rightarrow L.R$        $L.w = 1; R.w = 0.5; S.val = L.val + R.val;$
- $S \rightarrow L$        $L.w = 1; S.val = L.val$
- $L \rightarrow L1B$        $L1.w = L.w * 2; B.w = L.w; L.val = B.c + L1.val$
- $L \rightarrow B$        $B.w = L.w; L.val = B.c;$
- $R \rightarrow BR1$        $B.w = R.w; R1.w = R.w / 2; R.val = B.c + R1.val$
- $R \rightarrow B$        $B.w = R.w; R.val = B.c;$
- $B \rightarrow 0$        $B.c = 0;$
- $B \rightarrow 1$        $B.c = B.w;$

**(b)**

- $S \rightarrow M\{L.w = M.w\}L.N\{R.w = N.w\}R$        $S.val = L.val + R.val$
- $M \rightarrow \varepsilon$        $M.w = 1$
- $N \rightarrow \varepsilon$        $N.w = 0.5$
- $S \rightarrow P\{L.w = P.w\}L$        $S.val = L.val$
- $P \rightarrow \varepsilon$        $P.w = 1$
- $L \rightarrow \{Q.win = L.w\}Q\{L1.w = Q.wout\}L1\{T.win = L.w\}T\{B.w = T.wout\}B\{L.val = B.c + L1.val\}$
- $Q \rightarrow \varepsilon$        $Q.wout = Q.win * 2$
- $T \rightarrow \varepsilon$        $T.wout = T.win$
- $L \rightarrow \{B.w = R.w\}B$        $R.val = B.c$

**(b)**

- $R \rightarrow \{U.win = R.w\} U \{B.w = U.wout\} B \{V.win = R.w\} V \{R1.w = V.wout\} R1 \{R.val = B.c + R1.val\}$
- $U \rightarrow \varepsilon$                        $U.wout = U.win$
- $V \rightarrow \varepsilon$                        $V.wout = V.win/2$
- $R \rightarrow \{B.w = R.w\} B$          $R.val = B.c$
- $B \rightarrow 0$                        $B.c = 0$
- $B \rightarrow 1$                        $B.c = B.w$

## (b) ——属性栈代码

- $S \rightarrow ML.NR$      `stack[ntop].val=stack[top].val+stack[top-3].val`
- $M \rightarrow \varepsilon$             `stack[ntop].w=1`
- $N \rightarrow \varepsilon$             `stack[ntop].w=0.5`
- $S \rightarrow PL$             `stack[ntop].val=stack[top].val`
- $P \rightarrow \varepsilon$             `stack[ntop].w=1`
- $L \rightarrow QL1TB$         `stack[ntop].val=stack[top].c+stack[top-2].val`
- $Q \rightarrow \varepsilon$             `stack[ntop].w=stack[top].w*2`
- $T \rightarrow \varepsilon$             `stack[ntop].w=stack[top-2].w`
- $L \rightarrow B$             `stack[ntop].val=stack[ntop].c`

## (b) ——属性栈代码

- $R \rightarrow UBVR1$      $stack[ntop].val = stack[top].val + stack[top-2].c$
- $U \rightarrow \varepsilon$          $stack[ntop].w = stack[top].w$
- $V \rightarrow \varepsilon$          $stack[ntop].w = stack[top-2].w/2$
- $R \rightarrow B \text{ `}$          $stack[ntop].val = stack[ntop].c$
- $B \rightarrow 0$           $stack[ntop].c = 0$
- $B \rightarrow 1$           $stack[ntop].c = stack[top-1].w$



## 4.12 (a)

(a) 用继承属性 *depth* 表示嵌套深度, 所求的翻译方案如下:

$$S' \rightarrow \{ S. depth = 0; \mid S$$

$$S \rightarrow \{ L. depth = S. depth + 1; \mid ( L )$$

$$S \rightarrow a \mid \text{print} ( S. depth ); \mid$$

$$L \rightarrow \{ L_1. depth = L. depth; \mid L_1, \mid S. depth = L. depth; \mid S$$

$$L \rightarrow \{ S. depth = L. depth; \mid S$$

## 4.12(a)

- $S' \rightarrow M \{S.\text{depth} = M.s;\} S$
- $M \rightarrow \varepsilon \{M.s = 0;\}$  `val[ntop]= 0;`
- $S \rightarrow (\{N.i = S.\text{depth};\} N \{L.\text{depth} = N.s;\} L)$
- $N \rightarrow \varepsilon \{N.s = N.i+1;\}$  `val[ntop] = val[top - 1]+ 1;`
- $S \rightarrow a \{\text{print}(S.\text{depth});\}$  `print (val[top - 1]);`
- $L \rightarrow \{L_1.\text{depth} = L.\text{depth};\} L_1, \{P.i = L.\text{depth};\} P \{S.\text{depth} = P.s;\} S$
- $P \rightarrow \varepsilon \{P.s = P.i;\}$  `val[ntop] = val[top - 2]`
- $L \rightarrow \{S.\text{depth} = L.\text{depth};\} S$

## 4.12(b)

(b) 给文法符号  $S$  和  $L$  一个继承属性  $in$  和一个综合属性  $out$ , 分别表示在句子中, 该文法符号推出的字符序列的前面已经有多少个字符, 以及该文法符号推出的字符序列的最后一个字符在句子中是第几个字符。那么所求的翻译方案如下:

$$S' \rightarrow \{ S.in = 0; \mid S$$

$$S \rightarrow \{ L.in = S.in + 1; \mid ( L ) \mid S.out = L.out + 1; \mid$$

$$S \rightarrow a \mid S.out = S.in + 1; \text{ print } ( S.out ); \mid$$

$$L \rightarrow \{ L_1.in = L.in; \mid L_1, \mid S.in = L_1.out + 1; \mid S \mid L.out = S.out; \mid$$

$$L \rightarrow \{ S.in = L.in; \mid S \mid L.out = S.out; \}$$

## 4.12(b)

- $S' \rightarrow M \{S.in = M.s;\} S$
- $M \rightarrow \epsilon \{M.s = 0;\}$
- $S \rightarrow (\{N.i = S.in;\} N \{L.in = N.s;\} L) \{S.out = L.out + 1;\}$
- $N \rightarrow \epsilon \{N.s = N.i + 1;\}$
- $S \rightarrow a \{S.out = S.in + 1; \text{print}(S.out);\}$
- $L \rightarrow \{L_1.in = L.in;\} L_1, \{P.i = L_1.out + 1;\} P \{S.in = P.s;\} S \{L.out = S.out\}$
- $P \rightarrow \epsilon \{P.s = P.i;\}$
- $L \rightarrow \{S.in = L.in;\} S \{L.out = S.out\}$

## 4.12(b)

- $S' \rightarrow MS$

- $M \rightarrow \varepsilon$

- $S \rightarrow (NL)$

- $N \rightarrow \varepsilon$

- $S \rightarrow a$

- $L \rightarrow L1, PS$

- $P \rightarrow \varepsilon$

- $L \rightarrow S$

```
stack[ntop].in = 0;
```

```
stack[ntop].out = stack[top - 1].out + 1;
```

```
stack[ntop].in = stack[top - 1].in + 1;
```

```
stack[ntop].out = stack[top - 1].in + 1; print(stack[ntop].out);
```

```
stack[ntop].out = stack[top].out
```

```
stack[ntop].in = stack[top - 1].out + 1
```

```
stack[ntop].out = stack[top].out
```

# 3

移进(	(	
移进id	(id	
F->id归约	(F	print(6)
T->F归约	(T	print(4)
E->T归约	(E	print(2)
移进+	(E+	
移进id	(E+id	
F->id归约	(E+F	print(6)
T->F归约	(E+T	print(4)
E->E+T归约	(E	print(1)
移进)	(E)	
F->(E)归约	F	print(5)
T->F归约	T	print(4)
移进*	T*	
移进id	T*id	
F->id归约	T*F	print(6)
T->T*F归约	T	print(3)
E->T归约	E	print(2)

## 4-4.3-递归下降语法分析函数

```
• void S(){  
    if(lookahead()=='('){match('(');L();match(')');}  
    else if (lookahead()=='a'){match('a');}  
    else error()  
}  
  
void L(){  
    S();  
    while(lookahead()==' '){match(' ');S();}  
}
```

## 4-4.3-预测翻译器

- 消除左递归:
- $S' \rightarrow S$             `print(S.val)`
- $S \rightarrow (L)$             `S.val=L.val+1`
- $S \rightarrow a$                 `S.val=0`
- $L \rightarrow ST$               `L.val=S.val+L.val`
- $T \rightarrow ,ST_1$            `T.val=S.val+T1.val`
- $T \rightarrow \varepsilon$                `T.val=0`



## 4-4.3-预测翻译器

- `void S'(){print(S());}`
- `int S(){  
 int val;  
 if (lookahead()=='('){match('('); val=L()+1; match(')');}  
 else {match('a'); val=0;}  
 return val;  
}`  
`int L(){  
 int val; val=S()+T();return val;  
}`

## 4-4.3-预测翻译器

- `int T()`

```
{  
    int val=0;  
    if(lookahead()=='(',')' {match(',');val=S()+T();}  
    return val  
}
```

(也可以将一个`nodeptr`作为参数和返回值，将属性设置为`nodeptr`的属性即可)

## 4-4.12-预测翻译器

- $S' \rightarrow S$              $S.depth = 0$
- $S \rightarrow (L)$              $L.depth = S.depth + 1$
- $S \rightarrow a$              $print(S.depth)$
- $L \rightarrow ST$              $S.depth = L.depth; T.depth = L.depth$
- $T \rightarrow ,ST_1$            $S.depth = T.depth; T_1.depth = T.depth$
- $T \rightarrow \epsilon$

## 4-4.12-预测翻译器

- `void S'(){S(0);}`
- `void S(int depth){  
    int mydep;  
    if (lookahead()=='(')  
        {match('('); mydep=depth+1; L(mydep); match(')');}  
    else  
        {match('a');print(depth);}  
}`
- `void L(int depth){int mydep=depth;S(mydep);T(mydep);}`

## 4-4.12-预测翻译器

- void T(int depth)
- {  
    int mydep=depth;  
    if(lookahead()==','){match(',');S(mydep);T(mydep);} }  
}

**谢谢！**