2019 级弘毅班《编译原理》第三次练习答案

一、 设有 C 语言说明语句文法 D 如下所示：

$$
\begin{aligned}
D &\rightarrow T\ L \\
T &\rightarrow \texttt{int} \mid \texttt{char} \\
L &\rightarrow *\,L \mid L\,[\,] \mid L\,(\,) \mid (\,L\,) \mid \texttt{id}
\end{aligned}
$$

(1) 由于该文法是二义文法，因此其 LR 分析表一定有移进/归约或归约/归约冲突，试用 JFLAP 指出那些状态集有怎样的冲突；
解： 在状态 $I_{10} = \{ L \rightarrow *L\bullet, L \rightarrow L\bullet(), L \rightarrow L\bullet[] \}$ 面对 (和 [有移进/归约冲突, 选择移进表示后缀运算优先于前缀运算.

(2) 请正确地对冲突项目选择移进/归约构造其 SLR 分析表，使得按照该分析表分析正表达式时，其运算的优先级别和结合次序与 C 语言说明语句中规定的优先级和结合次序一致 (仅列出冲突项集合对应的 Action 表)；

	action							
状态	int	char	id	(	)	[	]	$
10				s11	r4	s14		r4

其中标号为 4 的产生式为： $L \rightarrow *L$. 完整的分析表截图如下所示：

	(	)	*	[	]	a	c	i	$	D	L	T
0							s3	s4		1		2
1									acc			
2	s5		s6			s8					7	
3	r3		r3			r3						
4	r2		r2			r2						
5	s5		s6			s8					9	
6	s5		s6			s8					10	
7	s11			s12					r1			
8	r8	r8		r8					r8			
9	s11	s13		s12								
10	s11	r4		s12					r4			
11		s14										
12					s15							
13	r7	r7		r7					r7			
14	r6	r6		r6					r6			
15	r5	r5		r5					r5			

其中 a 表示 id, c 表示 char, i 表示 int.

(3) 利用你的分析表分析输入正规表达式 “int * id ( ) [ ]’’ 的分析过程.
解：

剩余串	分析栈	分析动作
`int * id()[]$`	0	shift
`* id()[]$`	0`int`4	reduce $T \to$ `int`
`* id()[]$`	0T2	shift
`id()[]$`	0T2 $*$ 6	shift
`()[]$`	0T2 $*$ 6`id`8	reduce $L \to$ `id`
`()[]$`	0T2 $*$ 6L10	shift
`)[]$`	0T2 $*$ 6L10(11	shift
`[]$`	0T2 $*$ 6L10(11)14	reduce $L \to L()$
`[]$`	0T2 $*$ 6L10	shift
`]$`	0T2 $*$ 6L10[12	shift
`$`	0T2 $*$ 6L10[12]15	reduce $L \to L[]$
`$`	0T2 $*$ 6L10	reduce $L \to *L$
`$`	0T2L7	reduce $D \to TL$
`$`	0D1	分析成功

二、 设有文法 G 定义如下：

$$\begin{aligned} S &\to AB \\ A &\to aAb \mid aA \mid a \\ B &\to bBa \mid bB \mid \varepsilon \end{aligned}$$

(1) 试描述文法 G 所生成的语言；
解：文法生成的语言为: $\{ a^m b^{n+p} a^n \mid m \geqslant 1 \wedge n, p \in \mathbb{N} \}$.

(2) 试不用构造分析表直接说明该文法不是 SLR 文法;
解：A 成分有二义性，即：

$$\begin{aligned} A &\underset{lm}{\Rightarrow} aAb \underset{lm}{\Rightarrow} aaAb \underset{lm}{\Rightarrow} aaab \\ A &\underset{lm}{\Rightarrow} aA \underset{lm}{\Rightarrow} aaAb \underset{lm}{\Rightarrow} aaab \end{aligned}$$

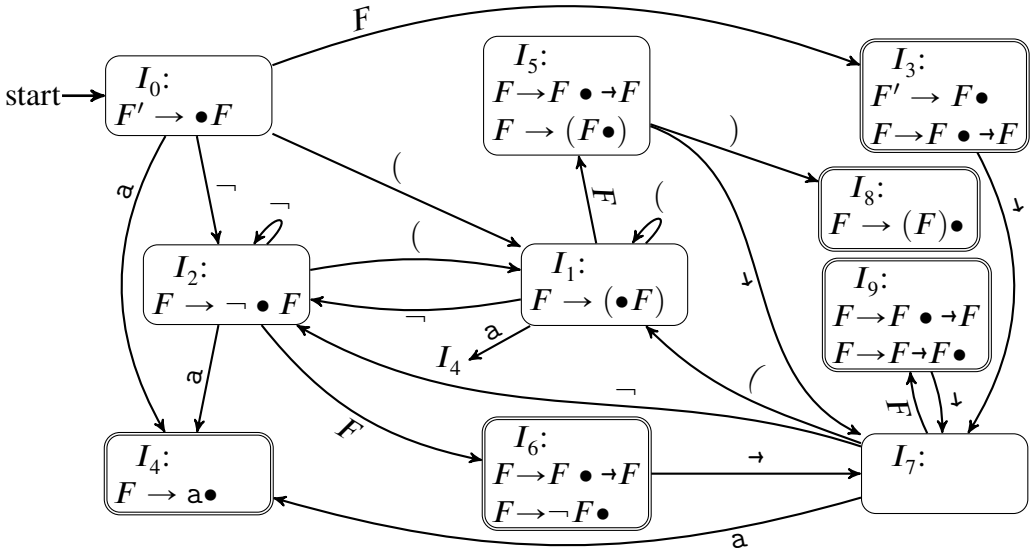
(3) 试修改文法 G 使之成为 SLR 文法.
解：

$$\begin{aligned} S &\to aA \\ A &\to aA \mid bB \mid \varepsilon \\ B &\to bB \mid C \\ C &\to bCa \mid a \end{aligned}$$

三、 设 $G(F)$ 的拓广文法 $G(F')$ 如下所示： (2020 考题)

$$\begin{aligned} F' &\to F && (0) \\ F &\to F \to F && (1) \\ &\mid \neg F && (2) \\ &\mid (F) && (3) \\ &\mid \mathtt{a} && (4) \end{aligned}$$

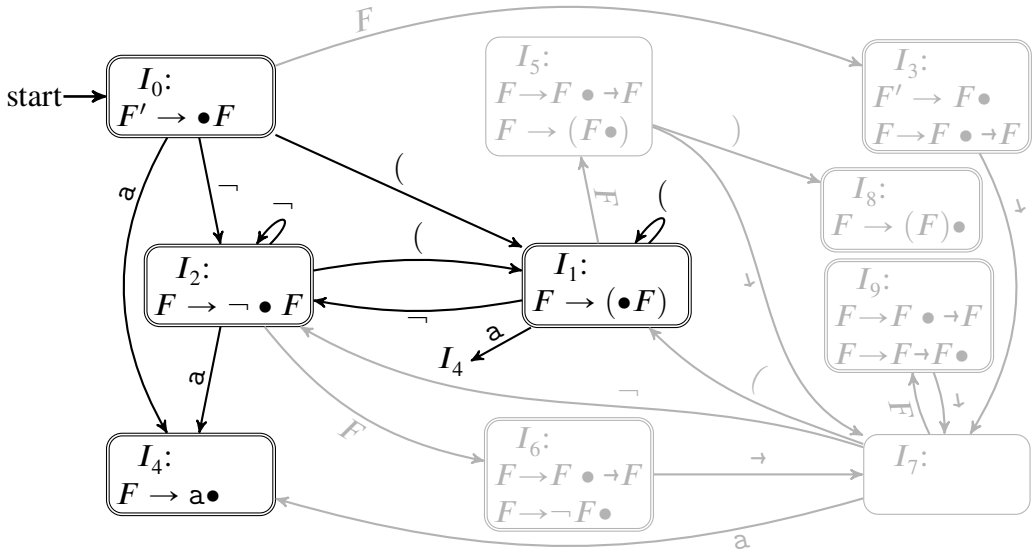
文法 $G(F')$ 的识别活前缀 LR(0) 项目自动机 M 如下图所示 (**注意每个状态仅列出了核心项目，状态 I_7 除外**)：



(1) 试求状态 I_7 所对应的 LR(0) 项目集；
状态 I_7 的 LR(0) 项目集为

$$\overline{\{\, F \to F \to \bullet\, F \,\}}$$
$$= \{\, F \to F \to \bullet\, F,\ F \to \bullet F \to F,\ F \to \bullet \neg F,\ F \to \bullet (F),\ F \to \bullet \mathtt{a} \,\}.$$

(2) 试求仅由**终结符号**组成的活前缀对应的正则表达式；
仅由**终结符号**组成的活前缀对应的前缀 DFA 为：



其对应的正则表达式为：(¬ | ()*a?.

(3) 试构造该文法的 SLR 分析表，并对分析表中的移进/归约和归约/归约冲突选择正确的移进或归约动作，使得文法 $G(F)$ 的所有语句能被正确地分析

且运算的优先级与结合次序与题三所规定的一致；
Follow(F) = {→,), \$}. 状态 I_6 和状态 I_9 面对 '→' 有移进/归约冲突. 分析表如下所示：

	action						goto
状态	a	¬	→	(	)	\$	F
0	s4	s2		s1			3
1	s4	s2		s1			5
2	s4	s2		s1			6
3			s7			acc	
4			r4		r4	r4	
5			s7		s8		
6			**r2**		r2	r2	
7	s4	s2		s1			9
8			r3		r3	r3	
9			**s7**		r1	r1	

(4) 试利用你的分析表写出语句“¬a→a”的分析过程.
语句“¬a→a”的分析过程如下所示：

剩余串	分析栈	分析动作
¬a→a\$	0	shift
a→a\$	0¬2	shift
→a\$	0¬2a4	reduce $F \to$ a
→a\$	0¬2F6	reduce $F \to \neg F$
→a\$	0F3	shift
a\$	0F3→7	shift
\$	0F3→7a4	reduce $F \to$ a
\$	0F3→7F9	reduce $F \to F \to F$
\$	0F3	reduce 分析成功

四、 现需对题四文法 $G(F')$ 所生成的命题公式转换为析取 (∨)、合取 (∧) 和仅有对原子取否的逻辑等价公式，如:

序号	原命题公式	转换后的命题公式
1	A→B	¬A∨B
2	¬¬A	A
3	¬(A→B)	(A)∧(¬B)
4	A→(B→C)	¬A∨¬B∨C
5	(A→B)→C	(A)∧(¬B)∨C

为此设计继承属性 F.is_neg, 其取值为布尔量 `True` 和 `False`; 综合属性 F.nnf，其取值为 F 所表示的语法成分对应的转换后的命题公式 (字符串)； a.lexeme

取值为 a 所对应的字符串. F.is_neg 的语义规则如下所示: (2020 年考题)

产生式	语义规则
$F' \to F$	F.is_neg = False
$F \to F_1 \to F_2$	F_1.is_neg = $\neg F$.is_neg; F_2.is_neg = F.is_neg
$F \to \neg F_1$	F_1.is_neg = $\neg F$.is_neg
$F \to (F_1)$	F_1.is_neg = F.is_neg

(1) 试写出属性 F.nnf 的语法制导定义;

产生式	语义规则

$F \to F_1 \to F_2$	**if** (F.is_neg == True) **then**
	F.nnf = "(" + F_1.nnf + ") ∧ (" + F_2.nnf + ")"
	else
	F.nnf = F_1.nnf + "∨" + F_2.nnf

$F \to \neg F_1$	F.nnf = F_1.nnf
$F \to (F_1)$	F.nnf = F_1.nnf

$F \to$ a	**if** (F.is_neg == True) **then**
	F.nnf = "¬" + a.lexeme
	else
	F.nnf = a.lexeme

(2) 试求“¬(((A→¬B)→C)→¬(D→E))”转换后的命题公式.
$((A) \wedge (B) \vee C) \wedge (\neg D \vee E)$

六、 设有正规表达式文法 G 定义如下:

$$
\begin{aligned}
T &\to AS \\
S &\to S + S \mid SS \mid S* \mid (S) \mid \epsilon \mid \mathbf{c} \\
A &\to \varepsilon \quad \text{(空产生式)}
\end{aligned}
$$

其中: '+' 表示并运算, 终结符 '**c**' 表示任意的字符, 'ϵ' 为正规表达式中的空串, 不是空产生式用到的空串, 其运算优先级与结合次序与正规表达式的规定一致。
现给出一个正规表达式直接到 DFA 的算法, 该算法需要对每个正规表达式计算如下属性:

(a) 终结符 **c** 的属性 *lexval* 为 **c** 所对应的字符, 如: $(a+b)*abb$ 的第一个 a 在词法分析抽象为单词 **c** 后, 其对应单词 **c** 的属性 $lexval = a$;

(b) 终结符 **c** 的属性 *pos* 为 **c** 所对应的字母在正规表达式中出现位置序号, 如: $(a+b)*abb$ 的序号为:
$(\underbrace{a}_{1} \mid \underbrace{b}_{2})* \underbrace{a}_{3} \underbrace{b}_{4} \underbrace{b}_{5}$;

(c) 非终结符 S 的属性 $firstpos$ 为 S 所表示的正规表达式中出现的字符对应 *pos* 集合, 且该 *pos* 上的字符能作为 S 所表达的字符串的首字符, 如:

$(a+b)*abb$ 的 $firstpos = \{1,2,3\}$，$pos = 4$ 上的字符 b 虽然能作为首字符出现在 $(a+b)*abb$ 所生成的字符串的首字符，但是该 b 是由 pos 为 2 上的 b 提供，不是 pos 为 4 上的 b 提供, 所以该正规表达是的 $firstpos$ 不包含 4;

(d) 非终结符 S 的属性 $S.lastpos$ 为 S 所表达的正规表达式中出现的字符对应 pos 集合，且该字符能作为 S 所表达的字符串的尾字符，如：$(a+b)*abb$ 的 $last = \{5\}$, $pos = 4$ 上的字符 b 虽然能作为尾字符出现在 $(a+b)*abb$ 所生成的字符串的首字符，但是该 b 是由 pos 为 5 上的 b 提供，不是 pos 为 4 上的 b 提供, 所以该正规表达是的 $lastpos$ 不包含 4;

(e) 非终结符终 S 和终结符 **c** 的属性 $nullable$ 为 $true$，当且仅当 S 所表达的字符串集合含有空串 ϵ, 否则为 $false$, 如：$(a+b)*abb$ 的 $nullable = false$;

(f) 非终结符终 S 和终结符 **c** 的属性 $followpos$ 为 pos 或输入结束标志 '$' 组成的集合，某一个 pos 属于 $S.followpos$ 或 **c**.$followpos$ 当且仅当该 pos 对应的字符能作为 S 或 **c** 对应的子正规表达式的后随字符。如 S 对应的是整个正规表达式，则 $followpos$ 为 $\{\$\}$; $pos = 1$ 的 a 所对应单词 **c** 的 $followpos = \{1,2,3\}$, 因为 $pos = 1$ 的 a 出现在 $(a+b)*abb$ 所生成的字符串之中时，其后随符号可以是 $pos = 1$ 提供的 a，或 $pos = 2$ 提供的 b，或 $pos = 3$ 上提供的 a, 但不可能在该 a 之后紧随 $pos = 4$ 提供的 b。

计算上述属性的语法制导定义如下, 其中终结符 **c** 的属性 pos 置为 $count$; $lexval$, $firstpos$ 和 $lastpos$ 均置为 **c** 所对应的字符:

编号	产生式	语义规则
0	$T \to AS$	$S.followpos = \{\$\}$
1	$A \to \varepsilon$	$count = 1$ (初始化全局变量 $count$)
2	$S \to S_1 + S_2$	$S.firstpos = S_1.firstpos \cup S_2.firstpos$ $S.lastpos = S_1.lastpos \cup S_2.lastpos$ $S.nullable = S_1.nullable \vee S_2.nullable$ $S_1.followpos = S.followpos$ $S_2.followpos = S.followpos$
3	$S \to S_1 S_2$	$S.firstpos =$ **if** $(S_1.nullable)$ **then** $\quad S_1.firstpos \cup S_2.firstpos$ **else** $S_1.firstpos$ $S.lastpos =$ **if** $(S_2.nullable)$ **then** $\quad S_1.lastpos \cup S_2.lastpos$ **else** $S_2.lastpos$ $S.nullable = S_1.nullable \wedge S_2.nullable$ $S_2.followpos = S.followpos$ $S_1.followpos =$ **if** $(S_2.nullable)$ **then** $\quad S_2.firstpos \cup S.followpos$ **else** $S_2.firstpos$
4	$S \to S_1*$	$S.firstpos = S_1.firstpos$ $S.lastpos = S_1.lastpos$ $S.nullable = true$ $S_1.followpos = S.followpos \cup S_1.firstpos$
5	$S \to (S_1)$	$S.firstpos = S_1.firstpos$ $S.lastpos = S_1.lastpos$ $S.nullable = S_1.nullable$ $S_1.followpos = S.followpos$
6	$S \to \mathbf{c}$	$S.firstpos = \mathbf{c}.firstpos$ $S.lastpos = \mathbf{c}.firstpos$ $S.nullable = false$ $\mathbf{c}.followpos = S.followpos$ $count = count + 1$
7	$S \to \epsilon$	$S.firstpos = \varnothing$ $S.lastpos = \varnothing$ $S.nullable = true$

(1) 分别指出属性 pos，$lexval$，$firstpos$, $lastpos$, $count$ 和 $followpos$ 是否为综合属性，L 属性和继承属性；

(2) 试画出 $(a+b)*abb$ 对应的附注语法树;

(3) 利用每个单词 **c** 的 $followpos$ 可以从正规表达式直接构造 DFA，首先 DFA 的初始状态 S_0 为语法树树根 T 的儿子 S 的 $firstpos$ 值，即整个表达式对应的 $firstpos$, 如果整个表达式的 $nullable$ 为 $true$, 表示自动机接受空串，

即 S_0 此时需要加上元素 '$'; 设 S_i 为 DFA 的一个状态，则状态转换函数:

$$Dtrans(S_i, a) = \bigcup_{\mathbf{c}.pos \in S_i \wedge \mathbf{c}.lexval = a} \mathbf{c}.followpos.$$

S_i 是接受状态, 当且仅当 $\$ \in S_i$; 如 $(a+b)*abb$ 的初始状态为 $\{1,2,3\}$, $Dtrans(\{1,2,3\}, a) = \mathbf{c}_1.followpos \cup \mathbf{c}_3.followpos$, 其中 $\mathbf{c}_1.pos = 1 \wedge \mathbf{c}_1.lexval = a$; $\mathbf{c}_3.pos = 3 \wedge \mathbf{c}_3.lexval = a$。试用上述方法求 $(a+b)*abb$ 对应的 DFA。

(4) 对正规表达式 $b*(ab*ab*)*$ 做第 (2) 和第 (3) 问。

解：

(1) 每个属性的类型如下表所示：

属性	综合	L 属性	继承
pos	y	y	n
$lexval$	y	y	n
$firstpos$	y	y	n
$count$	n	y	y
$lastpos$	y	y	n
$followpos$	n	n	y

由于 $followpos$ 不是 L 属性，因此对上述语法制导定义不能写翻译规程，计算 $followpos$ 可以采用的方法是：

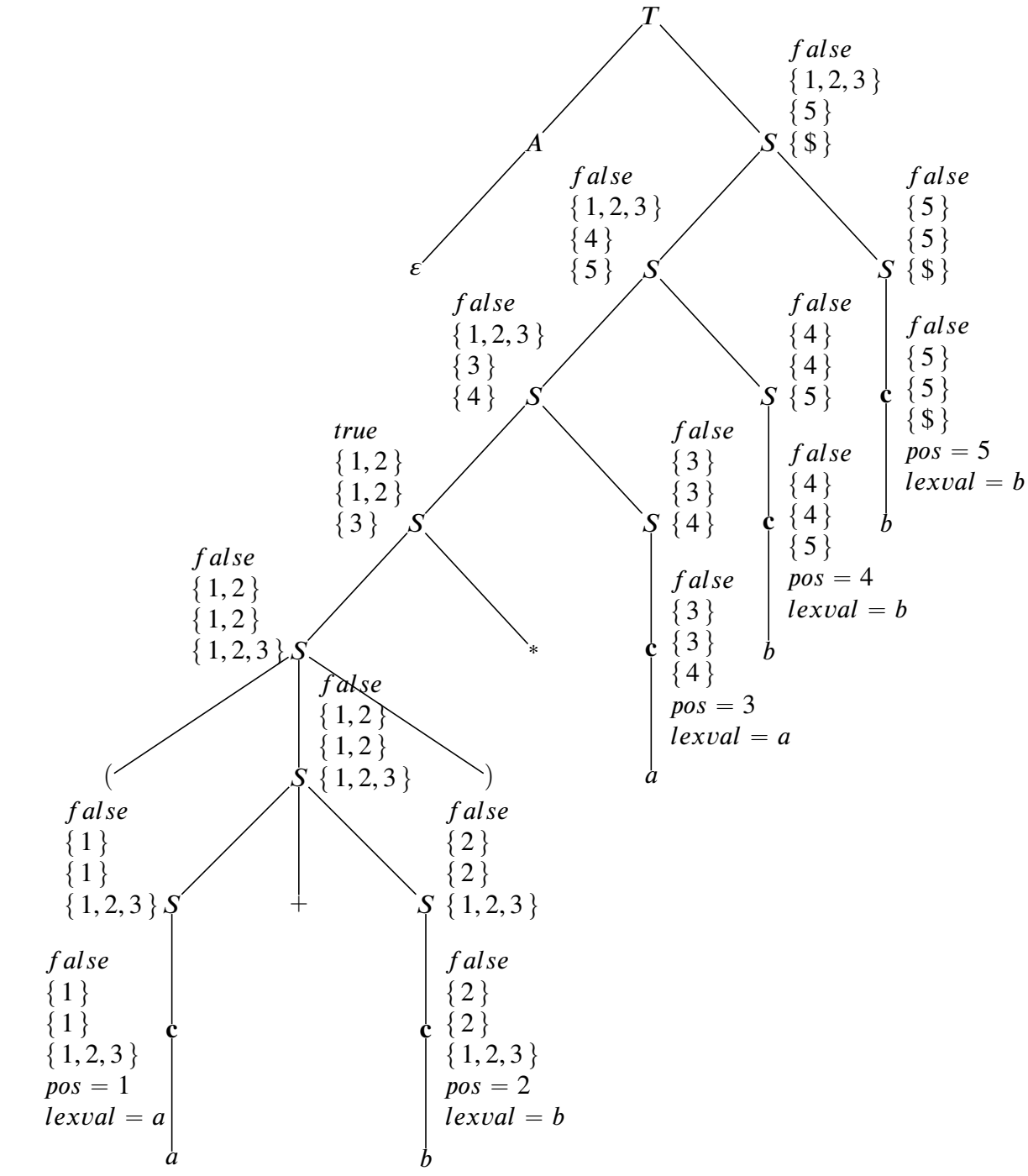
i. 先计算正规表达式的抽象语法树属性 ast；

ii. 对 ast 深度优先递归遍历求上述 L 属性，并将属性保存在 ast 上形成附注语法树。注意由于由于抽象文法与上具体文法不同，因此语义规则不再是对产生式制定，而是对不同树结点类型指定。此时结点类型有链接运算，并运算，闭包运算，字符，空串等类型，其对应的语义规则与具体文法的语义规则一致；

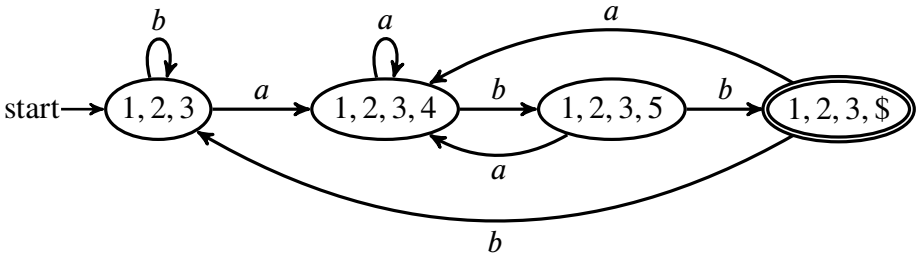
iii. 对 ast 反前序遍历计算 $followpos$ 属性。

该方法也是对抽象语法树定义和求解属性的典型应用。

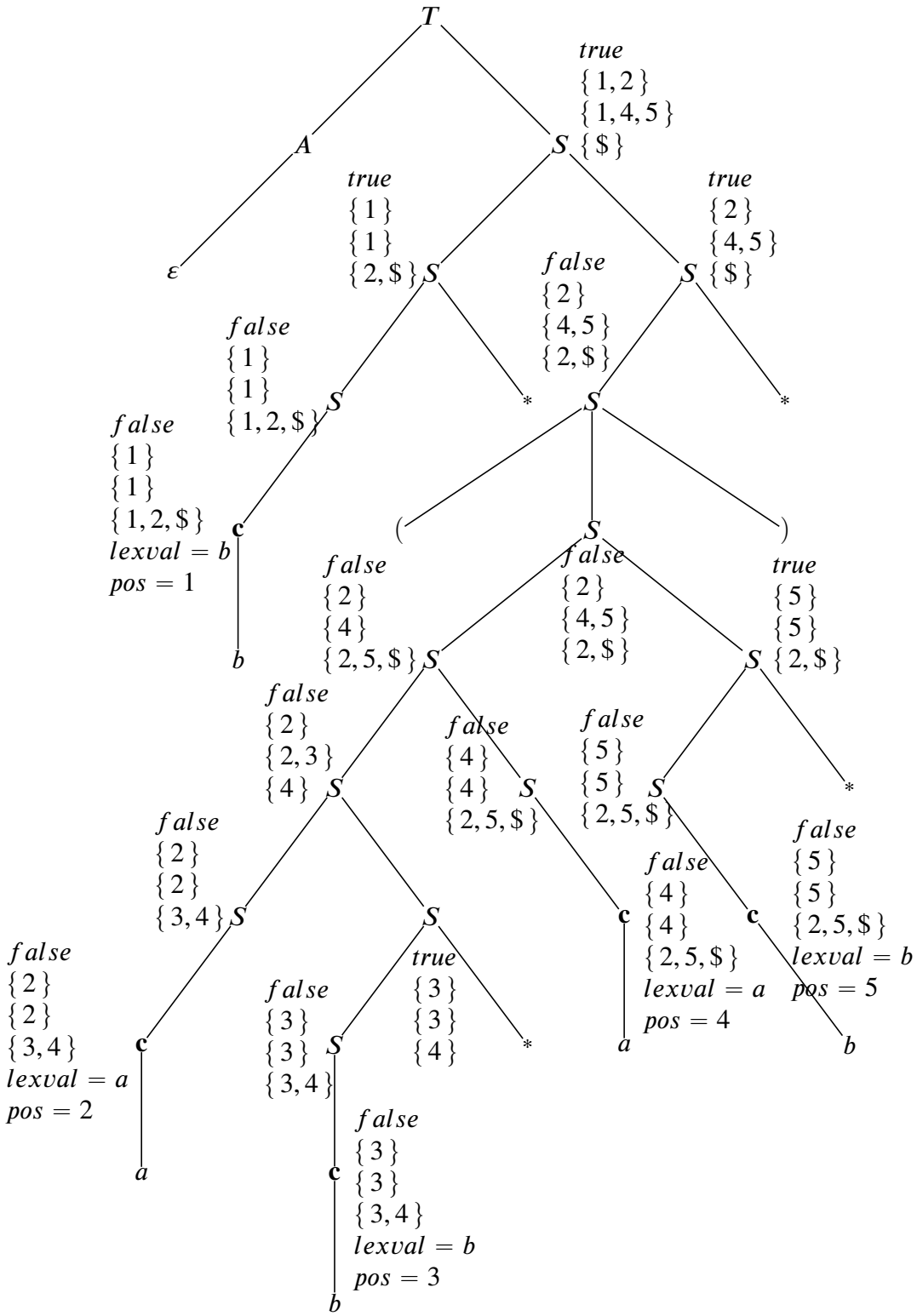
(2) $(a+b)*abb$ 对应的附注语法树如下图所示, 其中每个结点的属性从上至下分别是 $nullable$, $firstpos$, $lastpos$ 和 $followpos$ 等：


T
A
S
false
{1,2,3}
{5}
{$}
ε
false
{1,2,3}
{4}
{5}
S
S
false
{5}
{5}
{$}
false
{1,2,3}
{3}
{4}
S
S
false
{4}
{4}
{5}
c
false
{5}
{5}
{$}
pos = 5
lexval = b
b
true
{1,2}
{1,2}
{3}
S
S
false
{3}
{3}
{4}
c
false
{4}
{4}
{5}
pos = 4
lexval = b
b
false
{1,2}
{1,2}
{1,2,3}
S
*
c
false
{3}
{3}
{4}
pos = 3
lexval = a
a
false
{1,2}
{1,2}
S {1,2,3}
(
)
false
{1}
{1}
{1,2,3}
S
+
S
{2}
{2}
{1,2,3}
false
{1}
{1}
{1,2,3}
pos = 1
lexval = a
c
a
false
{2}
{2}
{1,2,3}
pos = 2
lexval = b
c
b

(3) 对应的自动机为：



(4) $b*(ab*ab*)*$ 对应的附注语法树如下图所示, 其中每个结点的属性从上至下分别是 $nullable$, $firstpos$, $lastpos$ 和 $followpos$ 等：


T
A
ε
true
{ 1, 2 }
{ 1, 4, 5 }
S { $ }
true
{ 1 }
{ 1 }
{ 2, $ } S
true
{ 2 }
{ 4, 5 }
S { $ }
false
{ 1 }
{ 1 }
{ 1, 2, $ } S
*
false
{ 2 }
{ 4, 5 }
{ 2, $ } S
*
false
{ 1 }
{ 1 }
{ 1, 2, $ }
c
lexval = b
pos = 1
b
(
S
)
false
{ 2 }
{ 4 }
{ 2, 5, $ } S
false
{ 2 }
{ 4, 5 }
{ 2, $ }
true
{ 5 }
{ 5 }
{ 2, $ } S
false
{ 2 }
{ 2, 3 }
{ 4 } S
false
{ 4 }
{ 4 } S
{ 2, 5, $ }
false
{ 5 }
{ 5 } S
{ 2, 5, $ }
*
false
{ 2 }
{ 2 }
{ 3, 4 } S
S
c
false
{ 4 }
{ 4 }
{ 2, 5, $ }
lexval = a
pos = 4
a
c
false
{ 5 }
{ 5 }
{ 2, 5, $ }
lexval = b
pos = 5
b
false
{ 2 }
{ 2 }
{ 3, 4 }
lexval = a
pos = 2
c
a
false
{ 3 }
{ 3 }
{ 3, 4 } S
true
{ 3 }
{ 3 }
{ 4 }
*
c
false
{ 3 }
{ 3 }
{ 3, 4 }
lexval = b
pos = 3
b

对应的自动机为：