编译原理 - 作业(3)：语法分析 LR、语义分析

截止时间: 2023/5/16（周二）课前， 14:19:59
提交方式: https://easyhpc.net/course/164

1. 对于如下文法 G:

(1) *E* -> *X Y*
(2) *X* -> c *X* a | b
(3) *Y* -> d

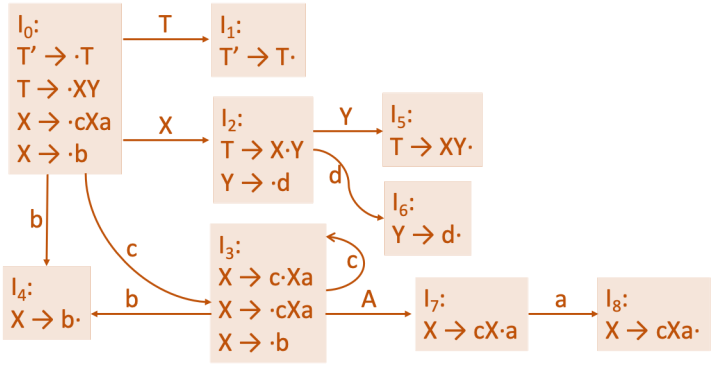
（1）求文法 G 的增广文法 G’[2 分]

【参考答案】

(0) E’-> E
(1) E -> XY
(2) X -> cXa | b
(3) Y -> d

（2）对文法 G’构建 LR(0)解析的有穷自动机(FA)，包括状态和转换。提示：项目集闭包（closure of item sets）[8 分]。

【参考答案】



（3）构建 LR(0)解析表。注：如有需要，请自行添加更多行。[7 分]

State	ACTION					GOTO		
	a	b	c	d	$	E	X	Y

【参考答案】

State	ACTION					GOTO		
	a	b	c	d	$	E	X	Y
0		s4	s3			1	2	
1					acc			
2				s6				5

3		s4	s3				7	
4	r3	r3	r3	r3	r3			
5	r1	r1	r1	r1	r1			
6	r4	r4	r4	r4	r4			
7	s8							
8	r2	r2	r2	r2	r2			

（4）列出解析输入串 cbad 的过程，包括每一步输入串和解析栈变化及采取的具体动作。注：如有需要，请自行添加更多行。[8 分]

Stack	Input	Action

Stack	Input	Action
0 $	cbad$	[0, c]: shift, to state 3
0 3 $ c	bad$	[3, d]: shift, to state 4
0 3 4 $ c b	ad$	[4, a]: reduce, X -> b
0 3 $ c A	ad$	[GOTO: [3, X] => 7
0 3 7 $ c A	ad$	[7, a]: shift, to state 8
0 3 7 8 $ c A a	d$	8, b]: reduce,X-> cXa
0 $ A	d$	GOTO: [0, X] => 2
0 2 $ A	d$	[2, b]: shift, to state 6
0 2 6 $ A d	$	[6, $]: reduce, Y -> d
0 2 $ A B	$	GOTO: [2, Y] => 5
0 2 5 $ A B	$	[5, $]: reduce, S -> AB
0 $ S	$	GOTO: [0, S] => 1
0 1 $ S	$	acc

2. 对于如下文法 G

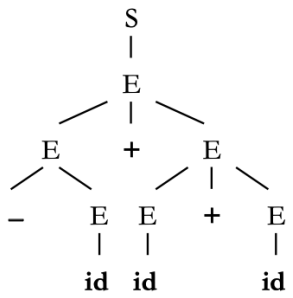
$$S \rightarrow E$$
$$E \rightarrow E + E \mid -E \mid \text{id}$$

(1) 构建该文法 LR(1)解析的 DFA；

(2) 对每一个含有冲突的状态，列出状态的编号、引起冲突的输入符号、以及冲突的类型；

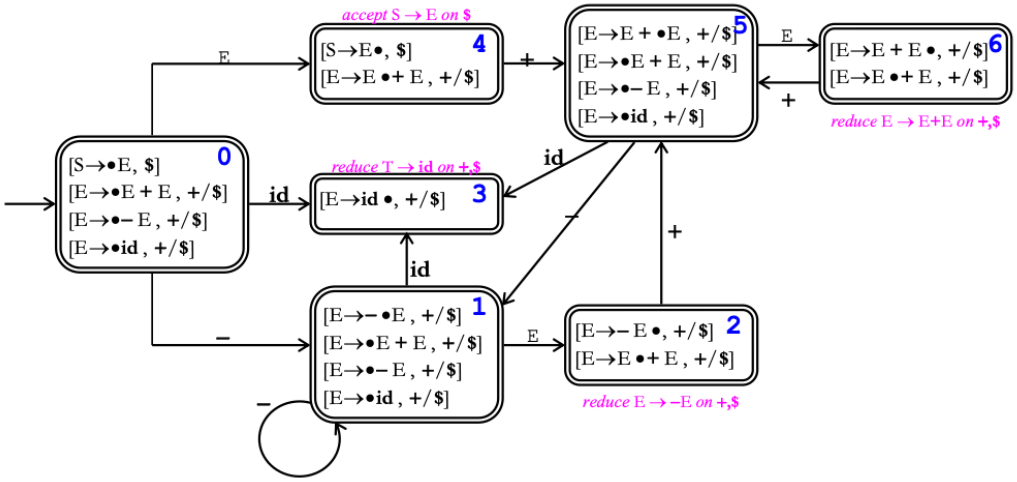
(3) 画出句子 id + – id + id 的所有分析树，并判断文法 G 是否具有二义性；

(4) 假设我们想让句子 – id + id + id 仅有如下一棵分析树是合法的(以下将此称为性质 P)。用自然语言描述:为保证性质 P，相关算符的优先级(Precedence)和结合性质(Associativity) 的规则如何?



(5) 为保证性质 P，根据上述 DFA 构造的 LR(1)分析表中的冲突应如何解析，即在“移进-归约”冲突中选择移进还是归约、在“归约-归约”冲突中选择以哪一个产生式归约。

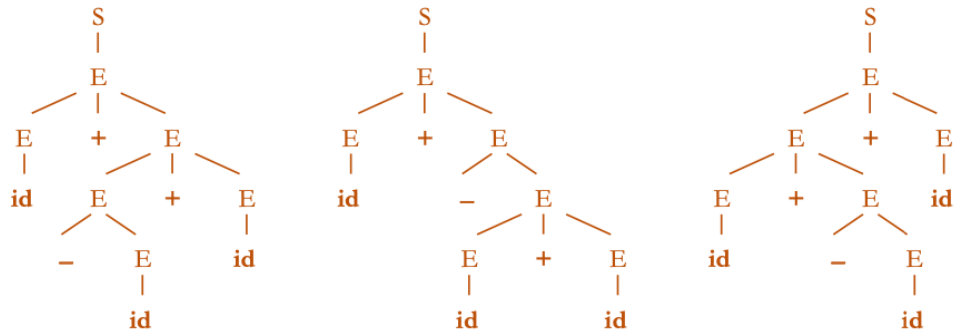
（1） 完整 DFA 如下：



（2） 存在如下冲突

状态	输入符号	冲突类型
2	+	“移入-归约”冲突
6	+	“移入-归约”冲突

（3） 句子 id + – id + id 有 3 棵不同的分析树:



该文法显然是一个二义文法，因为句子 id + – id + id 有多个合法的分析树。

（4） 为保证性质 P，我们需规定一元算符“–”的优先级要高于二元算符“+”，并且算符“+”是 右结合的。

（5） 保证性质 P，按以下方式解析冲突:

状态	输入符号	冲突类型	为解决冲突选择
2	+	“移入-归约”冲突	归约
6	+	“移入-归约”冲突	移入

3. 对于如下文法 G1 和 G2

G1:	G2:
S -> E + T E -> (E) | a T -> b	S -> E + T E -> (E) | a T -> b | b E

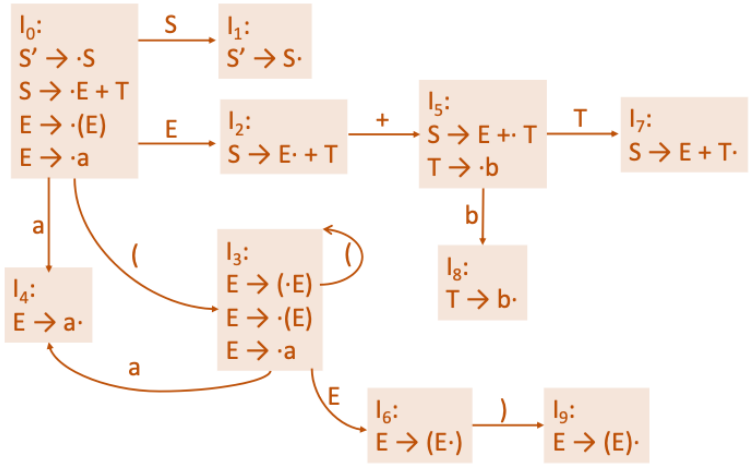
(1) 求文法 G1 的拓广文法（Augmented Grammar）G3，并为 G3 中的产生式编号（从 0 开始）；

【参考答案】拓广文法 G3 如下:

```
(0)    S' → S
(1)    S  → E + T
(2)    E  → ( E )
(3)    E  → a
(4)    T  → b
```

(2) 基于 LR(0)有效项目，构建一个识别文法 G3 所有活前缀的确定有限自动机（DFA）；

【参考答案】识别文法所有活前缀的 DFA 如下：



(3) 按下表格式构建文法 G3 的 LR（0）分析表；如有需求，请自行添加更多行；

State	ACTION						GOTO		
	a	b	(	)	+	$	S	E	T
0									
1									
… …									

【参考答案】文法 G3 的 LR(0)分析表如下所示：

State	ACTION						GOTO		
	a	b	(	)	+	$	S	E	T
0	s4		s3				1	2	
1						acc			
2					s5				
3	s4		s3					6	
4	r3	r3	r3	r3	r3	r3			
5		s8							7
6				s9					
7	r1	r1	r1	r1	r1	r1			
8	r4	r4	r4	r4	r4	r4			
9	r2	r2	r2	r2	r2	r2			

(4) 请参照下表格式给出输入串（a）+ b 的完整分析过程，包括每一步输入串和分析栈的变化及采取的动作（如果是 Reduce 动作，请指明归约所用的产生式），直到最后执行 Accept 或 Error 动作；如有需要，请自行添加更多行；

Step #	Stack		Input	Action
	Symbols	States		
0	$	0	(a) + b $	… …
1	… …	… …	… …	… …

【参考答案】该句子的 LR(0)分析过程如下：

Step #	Stack		Input	Action
	Symbols	States		
0	$	0	(a) + b $	A[0, (] = s3 Shift
1	$ (	0 3	a) + b $	A[3, a] = s4 Shift
2	$ (a	0 3 4	) + b $	A[4,)] = r3; G[3, E] = 6 Reduce with E → a
3	$ (E	0 3 6	) + b $	A[6,)] = s9 Shift
4	$ (E)	0 3 6 9	+ b $	A[9, +] = r2; G[0, E] = 2 Reduce with E → (E)
5	$ E	0 2	+ b $	A[2, +] = s5 Shift
6	$ E +	0 2 5	b $	A[5, b] = s8 Shift
7	$ E + b	0 2 5 8	$	A[8, $] = r4; G[5, T] = 7 Reduce with T → b
8	$ E + T	0 2 5 7	$	A[7, $] = r1; G[0, S] = 1 Reduce with S → E+T
9	$ S	0 1	$	A[1, $] = acc Accept

(5) 对于文法 G2，其基于 LR(0)有效项目的识别所有活前缀的 DFA 中有一个状态如下：

I_8:
T -> b •
T -> b • E
E -> • (E)
E -> • a

在 LR(0)分析过程中，该状态是否存在冲突？采用 SLR(1)是否可解析该冲突？请简要解释。

【参考答案】该状态存在“移进-归约”冲突，故文法 G2 不是 LR(0)的；由于 FOLLOW(T) = {$}，故“(”和“a”均不属于 FOLLOW(T)，因而 SLR(1)可以解析其中的冲突，文法 G2 是一个 SLR(1)文法。

(6) 简述 LR(1)是如何进一步改进 SLR(1) 分析技术的，并基于 LR(1)有效项目构造识别文法 G2 所有活前缀的 DFA 初始状态（提示：你不需要构造完整的 DFA，只需给出该 DFA 的初始状态）。

【参考答案】两者利用展望符号（Lookahead）的方式不同：SLR(1)简单地看 Lookahead 能否跟随被归约出来的非终结符（即 Lookahead 必须属于该非终结符的 FOLLOW 集），而 LR(1)要看 Lookahead 能否跟随整个分析栈内容（即活前缀），显然比 SLR(1)更加精确（可跟随的符号是 SLR(1)的子集）。因而，一些 SLR(1)无法解析的冲突有可能被 LR(1)解析。

该 DFA 的初始状态如下：

I_0:
S’ -> •S, $
S -> • E + T, $
E -> • (E), +
E -> • a, +