编译原理 -Assignment 2

1. 考虑文法G(S):

$$S \to AS'$$
$$S' \to +S|\epsilon$$
$$A \to BA'$$
$$A' \to A|\epsilon$$
$$B \to CB'$$
$$B' \to *B'|\epsilon$$
$$C \to (S)|a|b|\wedge$$

（1）判断文法 G(S)是否含有左递归、是否有产生式的候选式具有公共左因子；

不存在左递归，没有产生式候选式存在公共左公因子

（2）计算文法 G(S)中所有非终结符号的 FIRST 集和 FOLLOW 集；

非终结符：S、S'、A、A'、B、B'、C

计算FIRST集：

1：FIRST(S)={FIRST(A)}、FIRST(S')={+,ϵ}、FIRST(A)={FIRST(B)}、FIRST(A')={FIRST(A)、ϵ}、FIRST(B)={FIRST(C)}、FIRST(B')={*,ϵ}、FIRST(C)={ (、a、b、∧}

2：

FIRST(S)={ (、a、b、∧}

FIRST(S')={+,ϵ}

FIRST(A)={ (、a、b、∧}

FIRST(A')={ (、a、b、∧、ϵ}

FIRST(B)={ (、a、b、∧}

FIRST(B')={*,ϵ}

FIRST(C)={ (、a、b、∧}

计算Follow集：

1：FOLLOW(S)={FOLLOW(S')、) 、$}、FOLLOW($S'$)={FOLLOW($S$)、$}、FOLLOW(A)={FIRST(S')、FOLLOW(A')、FOLLOW(S)、$}、FOLLOW($A'$)={FOLLOW(A)、$}、FOLLOW(B)={FIRST(A')、$}、FOLLOW($B'$) = {$}、FOLLOW(C)={ FIRST(B')、$}

因为FIRST(S')、FIRST(A')、FIRST(B')包含ϵ

2：FOLLOW(S)={FOLLOW(S')、) 、$}、FOLLOW($S'$)={FOLLOW($S$)、$}、FOLLOW(A)={FIRST(S')、FOLLOW(A')、$}、FOLLOW($A'$)={FOLLOW(A)、$}、FOLLOW(B)={FIRST(A')、FOLLOW(A)、$}、FOLLOW($B'$) = {FOLLOW($B$)、$}、FOLLOW(C)={ FIRST(B')、FOLLOW(B)、$}

3：FOLLOW(S)={FOLLOW(S')、) 、$}、FOLLOW($S'$)={FOLLOW($S$)、$}、FOLLOW(A)={+,ϵ、FOLLOW(A')、$}、FOLLOW($A'$)={FOLLOW(A)、$}、FOLLOW(B)={ (、a、b、∧、ϵ、FOLLOW(A)、$}、FOLLOW($B'$) = {FOLLOW($B$)、$}、FOLLOW(C)={ *,ϵ、FOLLOW(B)、$}

4:

FOLLOW(S) = FOLLOW(S') = {) 、$}

FOLLOW(A) = FOLLOW(A') = {+、ϵ、$}

FOLLOW(B)={ (、a、b、∧、ϵ、+、$}

FOLLOW(B') = {(、a、b、∧、ϵ、+、$}

FOLLOW(C)={ (、a、b、∧、*,ϵ、+、$}

（3）判断文法 G(S)是否为 LL(1)文法，并说明理由；

对于$S' \to +S|\epsilon$：$+S$不会推出FOLLOW(S')的内容和空串，不冲突

对于$A' \to A|\epsilon$：同理FIRST(A)与FOLLOW(A')不相交，不冲突

对于$B' \to *B'|\epsilon$：$*B'$不会推出FOLLOW(B')的内容和空串，不冲突

对于$C \to (S)|a|b|\wedge$：明显不同产生式的结果不冲突

因此G(S)是LL(1)的文法

（4）构造文法 G(S)的预测分析表，并说明如何通过分析表来证明（3）中的结论；

输入	+	*	(	)	a	b	∧	$
S			$S \to AS'$		$S \to AS'$	$S \to AS'$	$S \to AS'$	
S'	$S' \to +S$			$S' \to \epsilon$				$S' \to \epsilon$
A			$A \to BA'$		$A \to BA'$	$A \to BA'$	$A \to BA'$	
A'	$A' \to \epsilon$		$A' \to A$		$A' \to A$	$A' \to A$	$A' \to A$	$A' \to \epsilon$
B			$B \to CB'$		$B \to CB'$	$B \to CB'$	$B \to CB'$	
B'	$B' \to \epsilon$	$B' \to *B'$	$B' \to \epsilon$		$B' \to \epsilon$	$B' \to \epsilon$	$B' \to \epsilon$	$B' \to \epsilon$
C			$C \to (S)$		$C \to a$	$C \to b$	$C \to \wedge$	

预测分析表中每一个位置最多只存在一个产生式，可证明G(S)是LL(1)文法

（5）根据（4）中所构造的预测分析表，按如下格式写出句子“a+b”的完整分析过程，其中动作一列只需写出Derive（推导）或 Match（匹配），输出一列代表了执行 Derive 动作时输出的产生式，执行 Match 动作时此列为空。

步骤	符号栈	输入串	动作（Derive/Match)	输出
0	a+b $	S $	Derive	$S \to AS'$
1	a+b $	AS' $	Derive	$A \to BA'$
2	a+b $	$BA'S'$ $	Derive	$B \to CB'$
3	a+b $	$CB'A'S'$ $	Derive	$C \to a$
4	a+b $	$aB'A'S'$ $	Match	
5	+b $	$B'A'S'$ $	Derive	$B' \to \epsilon$
6	+b $	$A'S'$ $	Derive	$A' \to \epsilon$
7	+b $	S' $	Derive	$S' \to +S$

步骤	符号栈	输入串	动作（Derive/Match）	输出
8	+b $	$+S$ $	Match	
9	b $	S $	Derive	$S \to AS'$
10	b $	AS' $	Derive	$A \to BA'$
11	b $	$BA'S'$ $	Derive	$B \to CB'$
12	b $	$CB'A'S'$ $	Derive	$C \to b$
13	b $	$bB'A'S'$ $	Match	
14	$	$B'A'S'$ $	Derive	$B' \to \epsilon$
15	$	$A'S'$ $	Derive	$A' \to \epsilon$
16	$	S' $	Derive	$S' \to \epsilon$

2. 考虑文法G(E):

$E \to aE|bE|a$

（1）构造文法 G(E)的 LR(0)项目集规范族；（记得对文法进行增广！）

增广文法：$(1)S \to E, (2)E \to aE, (3)E \to bE, (4)E \to a$

$I_0 : S \to \cdot E, E \to \cdot aE, E \to \cdot bE, E \to \cdot a$

$I_1 : S \to E\cdot$

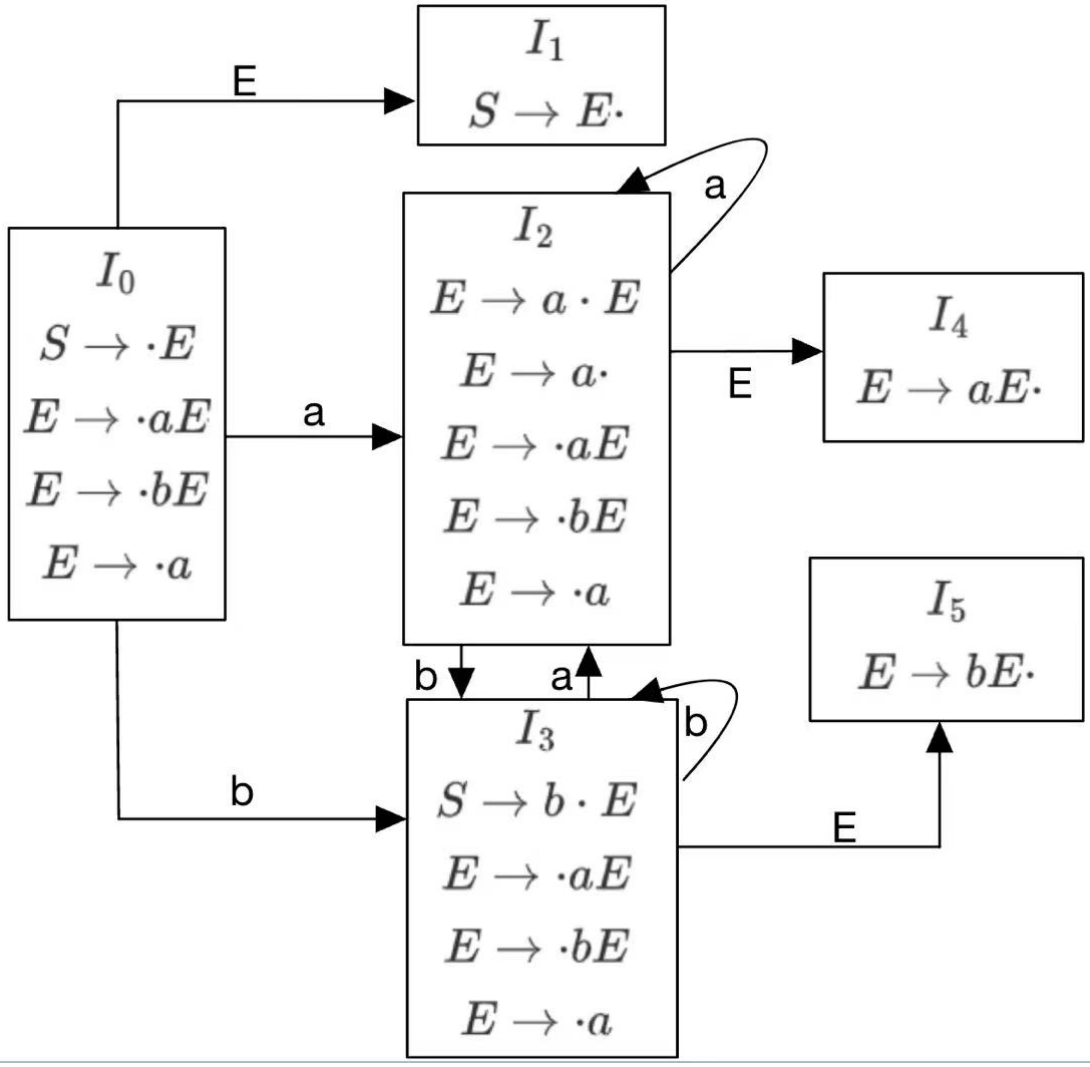
$I_2 : E \to a \cdot E, E \to a\cdot, E \to \cdot aE, E \to \cdot bE, E \to \cdot a$

$I_3 : E \to b \cdot E, E \to \cdot aE, E \to \cdot bE, E \to \cdot a$

$I_4 : E \to aE\cdot$

$I_5 : E \to bE\cdot$

（2）构造识别文法 G(E)所产生的活前缀的 DFA；



（3）判断文法 G(E)是否是 SLR(1)文法，并为其构造 SLR 分析表；

G(E)是SLR(1)文法，只有I_2中存在移入规约冲突，但a、b和FOLLOW(E)不冲突（FOLLOW(E)={$}），因此是SLR(1)文法

state	a	b	$	E
0	s2	s3		1
1			acc	
2	s2	s3	r4	4
3	s2	s3		5
4	r2	r2	r2	
5	r3	r3	r3	

（4）根据如下格式写出对输入串“ababa”的分析过程。

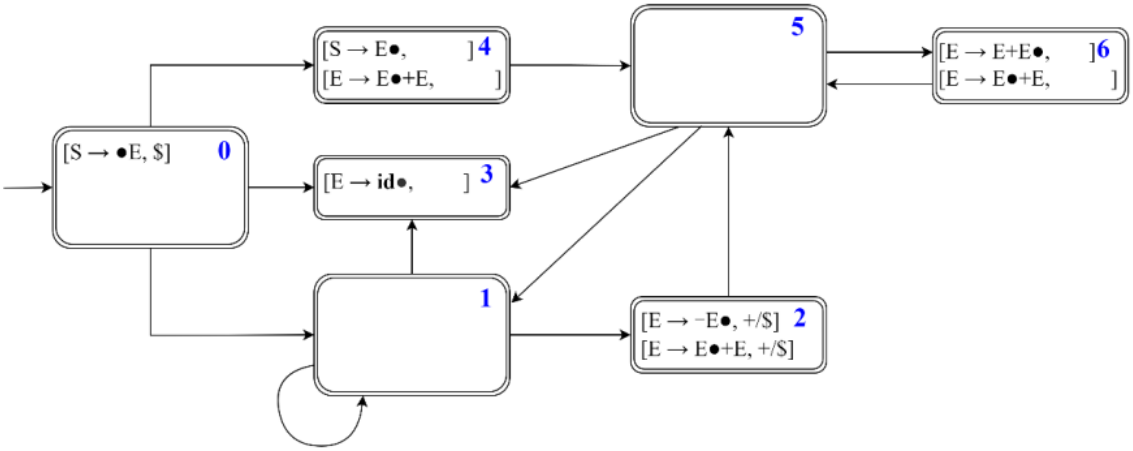
步骤	状态	符号	输入串	action	goto
0	0	$	ababa $	s2	
1	2	a$	baba $	s3	

步骤	状态	符号	输入串	action	goto
2	3	ab$	aba $	s2	
3	2	aba$	ba $	s3	
4	3	abab$	a $	s2	
5	2	ababa$	$	r4	5
6	5	ababE $	$	r3	4
7	4	abaE $	$	r2	5
8	5	abE $	$	r3	4
9	4	aE $	$	r2	1
10	1	E $	$	acc	

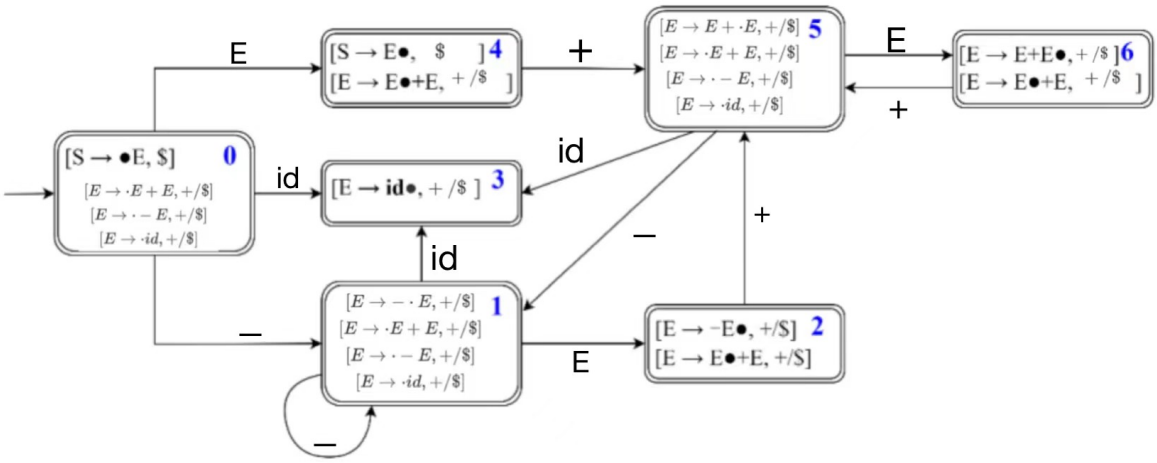
3. 考虑以下文法：

$$S \to E$$
$$E \to E + E | -E | id$$

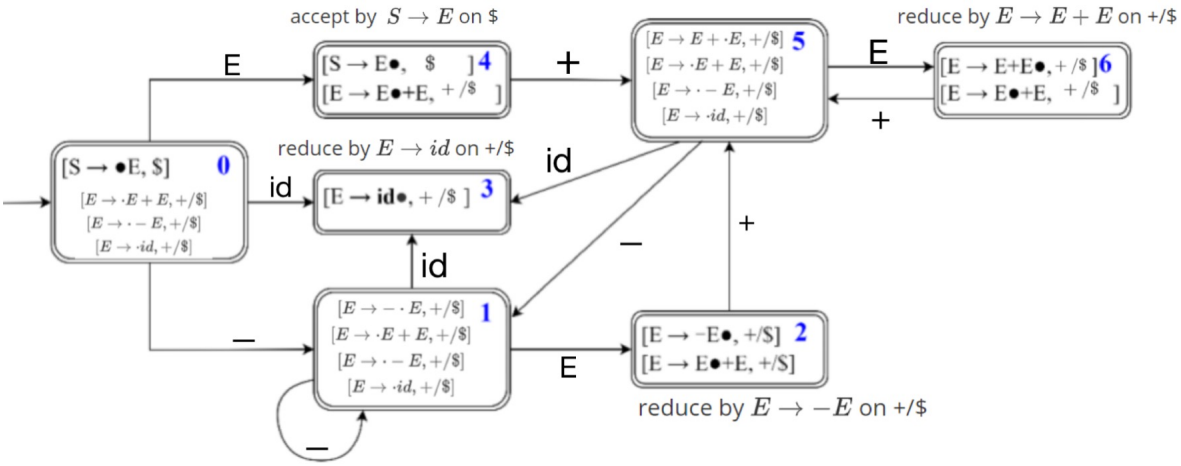
以下是识别该文法所有活前缀的DFA的一个局部图：



（1）补充完成上述DFA，具体包括：计算状态0中已有有效项目的闭包并完成状态0的填写；填写状态1和状态5中的元素；填写状态3、状态4和状态6中的向前看符号集；填写所有变迁上遗漏的符号。



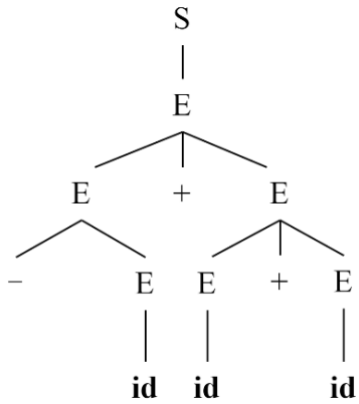
（2）在该DFA含有归约项目的状态旁边标识*“reduce by P on x, y, ...”*，表示在该状态见到*x*, y, ...*等向前看符号时用产生式*P* 归约；对于接受状态则将*reduce*...改为*accept*。



（3）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（“移进 - 归约”冲突、“归约 - 归约”冲突）

状态	输入符号	冲突类型
2	+	移进 - 归约
6	+	移进 - 归约

（4）显然，该文法是一个二义文法。假设我们想让句子−id+id+id仅有如下图所示的这一棵分析树是合法的（以下将此称为性质*P*），请用自然语言描述：为保证性质P，相关算符的优先级和结合性质的规则如何?



优先级：[+]的优先级高于[-]

结合性：[+]右结合，[-]左结合

（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”冲突中选择哪一个产生式归约?

状态	输入符号	冲突类型	为解析冲突选择
2	+	移进 - 归约	归约
6	+	移进 - 归约	移进

（6）写出一个与原文法等价、但能够保证性质P的无二义文法。

$$S \to E$$
$$E \to T + E | T$$
$$T \to -T | id$$