《编译原理》第二次书面作业

截止日期：2024 年 11 月 5 日

若发现问题，或有任何想法（改进题目、调整任务量等等），请及时联系助教

Q1. 下面的文法 $G[S]$ 描述由命题变量 p、q，联结词 $\wedge$（合取）、$\vee$（析取）、$\neg$（否定）构成的命题公式集合：

$$S \to S \vee T \mid T$$
$$T \to T \wedge F \mid F$$
$$F \to \neg F \mid p \mid q$$

试分别指出以下句型的所有短语和直接短语。如果这些句型同时也是右句型，那么还要给出其句柄。

1. $\neg F \vee \neg q \wedge p$
2. $\neg F \vee p \wedge \neg F \vee T$

参考答案：

1. 短语：$\neg F \vee \neg q \wedge p, \neg F, \neg q \wedge p, \neg q, q, p$；
   直接短语：$\neg F, p, q$；
   句柄：$\neg F$。
2. 短语：$\neg F \vee p \wedge \neg F \vee T, \neg F \vee p \wedge \neg F, \neg F$（第一个）$, p \wedge \neg F, p, \neg F$（第二个）；
   直接短语：$\neg F$（第一个）$, p, \neg F$（第二个）。（注意不同位置的短语不能合并）

Q2. 小明正在设计一款简单的编程语言。

1. 首先，小明要引入常表达式 `a` 和赋值语句 `a=b`，赋值语句的左侧（L）和右侧（R）都是一个标识符（终结符 i），文法可以写作 $G_1[S]$：

   $$S \to L = R \mid R$$
   $$L \to i$$
   $$R \to i$$

   请问该文法是否是 *LR(0)* 文法？结合 *LR(0)* 自动机说明原因。（思考：如果不构造自动机，你能否直接判断？）

2. $G_1[S]$ 是一个 *SLR(1)* 文法，画出它的 *SLR(1)* 分析表。

3. 小明希望为自己的编程语言引入指针特性，比如允许 `*a=**b` 这样的写法。于是，他将

文法修改为 $G_2[S]$：

$$S \to L = R \mid R$$
$$L \to *R \mid i$$
$$R \to L$$

现在，它不再是 *SLR(1)* 文法了，但仍是 *LR(1)* 文法，请画出它的 *LR(1)* 自动机。

4. 基于 $G_2[S]$ 的 *LR(1)* 自动机进行分析，若处于某个正常状态时所面临的输入符号是 $=$，且当前栈顶已形成句柄，那么这样的状态共有多少个？

5. 基于 $G_2[S]$ 的 *LR(1)* 自动机进行分析，若处于某个出错状态时所面临的输入符号是 $=$，那么这样的状态共有多少个？

6. 小明不想实现上面复杂的 *LR(1)* 自动机，他希望将其简化。如果改用 *LALR(1)* 自动机，那么原 *LR(1)* 自动机中哪些状态可以被合并？

7. 请问 $G_2[S]$ 是否是 *LALR(1)* 文法？要求只判断一种冲突，是哪种？

8. 画出 $G_2[S]$ 的 *LALR(1)* 分析表。

9. 基于你的 *LALR(1)* 分析表，分析下面的输入串：$*i = **i$。

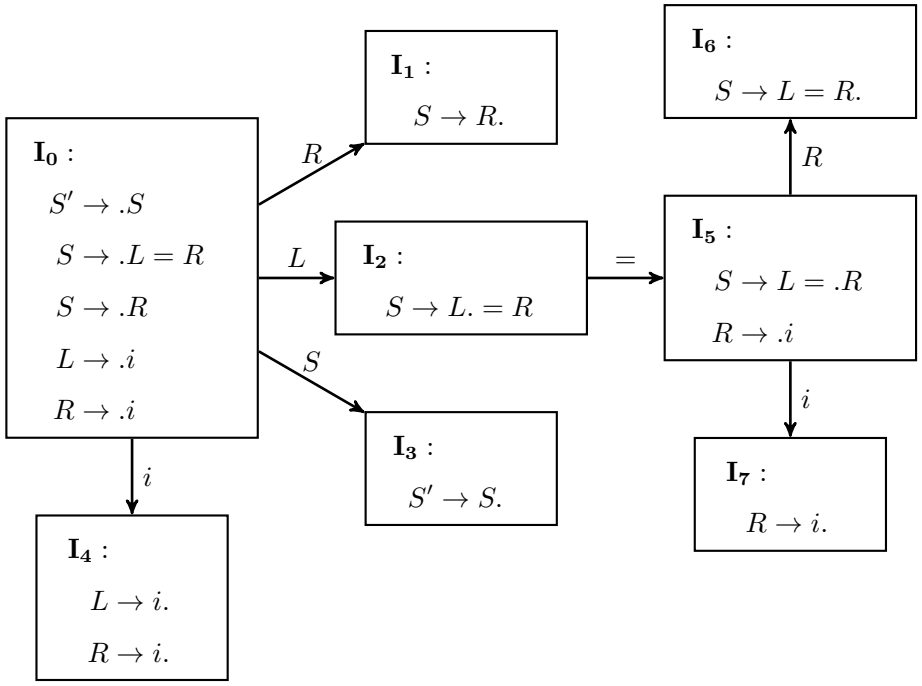
10. 小明希望为自己的编程语言引入连续赋值特性，比如允许 `a=*b=c` 这样的写法，同时小明删去了对常表达式 `a` 的支持。于是，他将文法修改为 $G_3[S]$：

$$S \to L = R$$
$$L \to *R \mid i$$
$$R \to L \mid L = R$$

小明发现，这是一个二义文法。请给出一个合理的限定规则，并说明该规则会引起 *LR* 分析表中何种变化（描述到位即可，无需构造出分析表）。

参考答案：

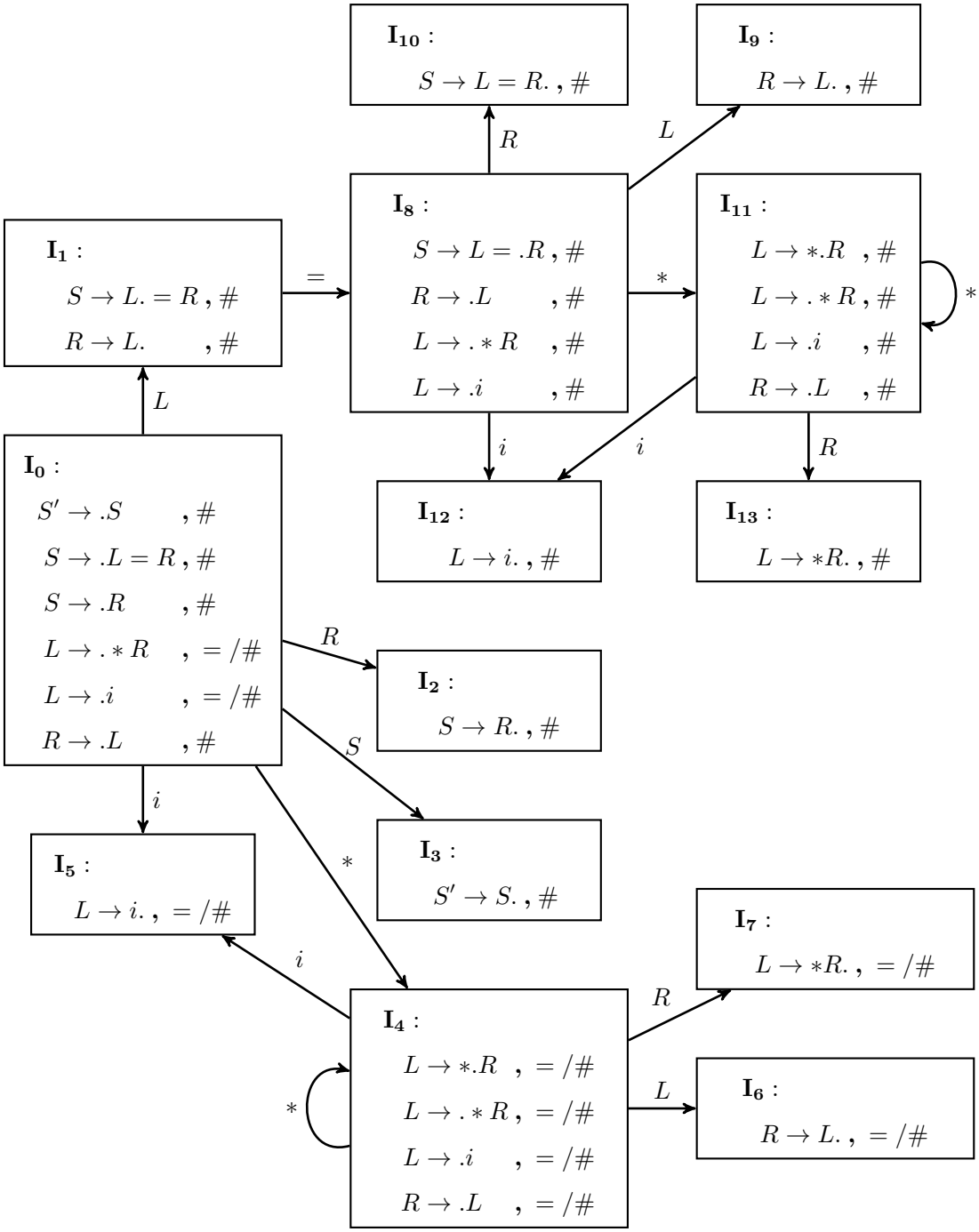
1. 增广文法 $G_1'[S']$ 的 LR(0) 自动机如下图所示。状态 I_4 存在归约-归约冲突，因此 $G_1[S]$ 不是 LR(0) 文法。（思考：对于一个合法字符串，第一步一定是移进 i，此后会陷入归约/归约冲突（$L \to i$ 和 $R \to i$），故不是 LR(0) 文法。）



2. 增广文法 $G'_1[S']$ 的 SLR(1) 分析表如下表所示。

	ACTION			GOTO		
栈顶状态	i	$=$	\$	S	L	R
0	s4			3	2	1
1			r2			
2		s5				
3			acc			
4		r3	r4			
5	s7					6
6			r1			
7			r4			

3. 增广文法 $G'_2[S']$ 的 LR(1) 自动机如下图所示。



4. 共 3 个状态：I_5, I_6, I_7。栈顶已经形成句柄的含义是该状态将进行归约。

5. 共 5 个状态：$I_0, I_4, I_8, I_{11}, I_{12}$。注意这样的状态不能通过归约得到，因为即将归约时会检查下一个符号，如不匹配会直接报错。

6. 共 4 组：状态 I_4, I_{11}、状态 I_5, I_{12} 、状态 I_6, I_9 以及状态 I_7, I_{13}。

7. 合并过程没有引入归约-归约冲突，因此 $G_2[S]$ 是 LALR(1) 文法。

8. 增广文法 $G_2'[S']$ 的 LALR(1) 分析表如下表所示。

栈顶状态	ACTION				GOTO		
	i	$*$	$=$	$\#$	S	L	R
0	s5-12	s4-11			3	1	2
1			s8	r5			
2				r2			
3				acc			
4-11	s5-12	s4-11				6-9	7-13
5-12			r4	r4			
6-9			r5	r5			
7-13			r3	r3			
8	s5-12	s4-11				6-9	10
10				r1			

9. 分析过程如下表所示。

状态栈	余留符号串	下一步动作
0	$*i=**i\#$	ACTION[0, $*$] = s4-11
0 4-11	$i=**i\#$	ACTION[4-11, i] = s5-12
0 4-11 5-12	$=**i\#$	ACTION[5-12, $=$] = r4, GOTO[4-11, L] = 6-9
0 4-11 6-9	$=**i\#$	ACTION[6-9, $=$] = r5, GOTO[4-11, R] = 7-13
0 4-11 7-13	$=**i\#$	ACTION[7-13, $=$] = r3, GOTO[0, L] = 1
0 1	$=**i\#$	ACTION[1, $=$] = s8
0 1 8	$**i\#$	ACTION[8, $*$] = s4-11
0 1 8 4-11	$*i\#$	ACTION[4-11, $*$] = s4-11
0 1 8 4-11 4-11	$i\#$	ACTION[4-11, i] = s5-12
0 1 8 4-11 4-11 5-12	$\#$	ACTION[5-12, $\#$] = r4, GOTO[4-11, L] = 6-9
0 1 8 4-11 4-11 6-9	$\#$	ACTION[6-9, $\#$] = r5, GOTO[4-11, R] = 7-13
0 1 8 4-11 4-11 7-13	$\#$	ACTION[7-13, $\#$] = r3, GOTO[4-11, L] = 6-9
0 1 8 4-11 6-9	$\#$	ACTION[6-9, $\#$] = r5, GOTO[4-11, R] = 7-13
0 1 8 4-11 7-13	$\#$	ACTION[7-13, $\#$] = r3, GOTO[8, L] = 6-9
0 1 8 6-9	$\#$	ACTION[6-9, $\#$] = r5, GOTO[8, R] = 10
0 1 8 10	$\#$	ACTION[10, $\#$] = r1, GOTO[0, S] = 3
0 3	$\#$	ACTION[3, $\#$] = acc

10. 可以规定 $*$ 运算符优先级高于 $=$ 运算符。原 LR 分析表中会存在移进-归约冲突（冲突表项对应的输入符号为 $=$），前述规则下我们强制移进（不允许归约），该表项不再冲突。

Q3. 在课堂上，我们介绍了 *LR(0)* 分析器，却不曾提到 *LL(0)* 分析器。尝试说明为什么我们不研究 *LL(0)* 分析器？进一步地，二者都不向前查看符号，那么 *LR(0)* 分析器比 *LL(0)* 分析器强在哪里？

参考答案： LL(0) 分析器只能根据当前最左侧非终结符选择产生式，这就要求每个非终结符的产生式唯一，即一个文法要么产生唯一字符串，要么陷入死循环，这不具备应用价值。相比之下，虽然 LR(0) 分析器也不能向前查看符号，但它维护了一个状态栈，可以根据栈顶状态选择移进或归约，因此分析能力强于 LL(0)。