文法和语言

分为0，1，2，3四种文法

其中1型文法又叫上下文有关文法，2型文法为上下文无关文法，3型文法为规范文法

一般以2型文法为主

3型文法只存在两种形式：

A：=a　　或　　A：=a

A：=aB　　A：=Ba

右线性　　左线性

上下文无关：

出现在左侧的一定是非终结符 仅出现在右侧的为终结符

一个文法只能对应产生一种语言 但一个语言可由不同文法构造

文法推导出的**含有终结符+非终结符**的称为句型 **仅含有非终结符**的称为句子

最左推导：每步直接推导总是替换最左的非终结符

最右推导：规范推导　规范句型：右句型

最右推导的逆过程：最左归约（规范归约）

如果一个文法存在某个句子对应两棵不同的语法树，则该文法是二义的

自顶向下：推导　自底向上：归约

子树的末端结点符号串是相对于子树根的**短语**

简单子树的末端结点组成的符号串是相对于简单子树根的**简单短语**

最左简单子树的末端结点组成的符号串是**句柄**

简单子树：包括根结点只有两层

正规式与有穷自动机

正规式画状态转换图

左线性：（特殊记忆）

· 增设初态S 开始符号对应终态 每个非终结符为状态结点

·对于A->Ba 从B到A画弧 标为a

·对A->a 从初态S到A画弧 标为a

右线性：（正常思维）

·增设终态Z 开始符号对应初态 每个非终结符为状态结点

·对于A->aB 从A到B画弧 标为a

·对于A->a 从A到终态Z画弧 标为a

五元组DFA：（K，∑，M，S，F）

K：所有状态结点的集合

∑：弧上标号的集合

M：转换的集合，例如M（A，b）=B表示从A到B有一条弧标为b

S：初态

F：终态集合

First集和Follow集

First集规则：相应字母在->左边 查找->右边的第一个符号

· A->aB　将a加入First(A)

· A->ε　将ε加入First(A)

· A->Xa　　把First(X)中去掉ε 加入First(A)

Follow集规则：相应字母在->右边 查找相应字母右边的符号

· 若A为开始符号，则Follow(A)有#

· B->Aa　　将a加入Follow(A)　**是终结符直接加入**

· B->AC　First(C)加入Follow(A)　**不是终结符，加入其First集**

· B->aA 或 B->aAC，C->ε Follow(B)加入Follow(A) **后面无符号，或有符号但可取空，将->左边字母的Follow集加入**

LL（1）文法

前置知识-Select集合

自顶向下的文法

第一个L：从左向右看

第二个L：最左推导

1:向前看一个字符

在LL(1)文法中，对于每个非终结符A的两个产生式 A->α , A->β

满足：Select(A->α) ∩ Select(A->β) = ∅

(α 和 β 至多有一个能推导出 ε)

非LL(1)文法到LL(1)文法的等价变换

- 提取左公因子
  - 左公因子：左前缀。例如 A::=aAb|ab ， 左公因子应为 a 而不是 ab

    提取后，有A::=aA', A'::=Ab|b
- 消除左递归

- 1.消除直接左递归（左递归变化成右递归）

- 2.消除间接左递归

  经过若干步推导，将间接左递归变成直接左递归，然后再使用消除直接左递归的方式

构造预测分析表

- 构造预测分析表的步骤
  （1）对每个终结符a∈FIRST（a），将A->a加到M[A,a]中。
  （2）如果ε∈FIRST（a），则对于任何b∈FOLLOW（A），将A->a加到M[A,b]中。

例：

- 例解：

==注意：下文的测试用例中使用的文法如下：==

E→TE’
E’→+TE’|ε
T→FT’
T’→*FT’|ε
F→(E)|i

（1）求文法对应的FIRST和FOLLOW集合

求非终结符号的First集：
First(E)={(,i}
First(E’)={+,ε}
First(T)={(,i}
First(T’)={*,ε}
First(F)={(,i}

非终结符的Follow集：
Follow(E)={),#}
Follow(E’)={),#}
Follow(T)={+,),#}
Follow(T’)={+,),#}
Follow(F)={*,+,),#}

（2）构造预测分析表初步：
行（非终结符号）：E,E’,T,T’,F
列（终结符号）：i，+，*，（，），#
填写：非终结符号U+输入符号a->下一条执行的产生式

	i	+	*	(	)	#
E	E→TE’			E→TE’		
E’		E’→+TE’			E’→ε	E’→ε
T	T→FT’			T→FT’		
T’		T’→ε	T’→*FT’		T’→ε	T’→ε
F	F→i			F→(E)		

个人理解：

- 如果一个非终结符A的first集合中有{a,b}，那么表格[A,a]和[A,b]会添加对应内容
- 如果一个非终结符A的first集合中有ε，那么在以上基础上观察其follow集合，如果follow集合中有{c,d}，那么表格[A,c]和[A,d]会添加对应内容
- 如果一个非终结符A的first集合中没有ε，那么不会关心其follow集合

算符优先分析

- 设有一文法G，如果G中没有形如A->…BC…的产生式，其中B和C为非终结符，则称G为算符文法
- 设有一不含空产生式的算符文法G，如果对任意两个终结符对a,b之间至多只有优先级高于、低于和等于三种关系中的一种成立，则称G是一个算符优先文法

FirstVT和LastVT

- **首先定义两个集合FIRSTVT(B)和LASTVT(B)**
  FIRSTVT(B)={b|B+=>b…或B+=>Cb…}，其中…表示V*中的符号串。
  LASTVT(B)={a|B+=>…a或B+=>…aC}

在以上的基础上：

FirstVT：能够推导出的第一个非终结符集合

- 如果有 b ∈ FirstVT(B) 且 A->B... ，那么有 b ∈ FirstVT(A)

LastVT：能够推导出的最后一个终结符集合

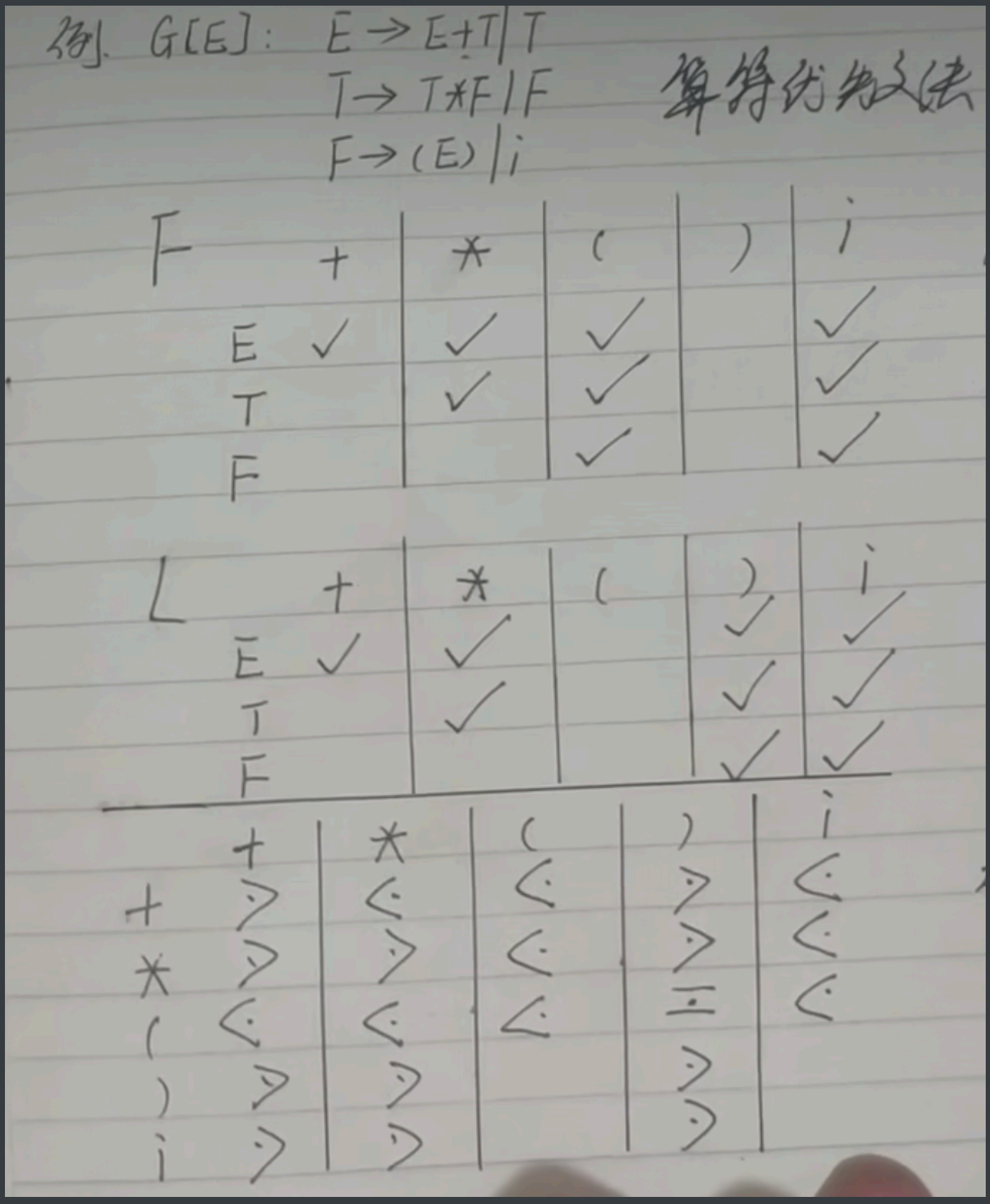
- 如果有 b ∈ LastVT(B) 且 A->...B ，那么有 b ∈ LastVT(A)

构造预测分析表

假定 G 是一个不含 ε 产生式的算符文法。对于任意一对终结符 a, b，我们说：

- $a = b$，当且仅当文法 G 中含有形如 $P \to \ldots ab \ldots$ 或 $P \to \ldots aQb \ldots$ 的产生式。
- $a < b$，当且仅当文法 G 中含有形如 $P \to \ldots aR \ldots$ 的产生式，而 $R \overset{+}{\Rightarrow} b \ldots$ 或 $R \overset{+}{\Rightarrow} Qb \ldots$ 。
- $a > b$，当且仅当文法 G 中含有形如 $P \to \ldots Rb \ldots$ 的产生式，而 $R \overset{+}{\Rightarrow} \ldots a$ 或 $R \overset{+}{\Rightarrow} \ldots aQ$ 。

- 如果存在 P->...aR...，那么a的**优先级小于**FirstVT(R)中的所有元素
- 如果存在 P->...Rb...，那么LastVT(R)中的所有元素**优先级大于**b



素短语

- 至少包含一个终结符

- 该短语不再包含满足第一个条件的更小的短语

最左边的素短语称为最左素短语

算符优先分析过程

对比栈顶符号和输入串最左边符号的优先级，如果栈顶归约后为终结符，那么比较栈顶的第二个符号。

如果栈顶符号优先级>输入串最左边符号，那么归约

如果栈顶符号优先级<输入串最左边符号，那么移进

如果栈顶符号优先级=输入串最左边符号，那么先移进后归约

LR（0）分析

自底向上语法分析

LR（0）项目集规范族的构造

- 构造识别活前缀的DFA（框图）
- 构成识别一个文法活前缀的DFA项目集（状态）的全体称为这个文法的LR（0）项目集规范族（列表）

LR（0）项目：在文法G中产生式的右部**适当位置**添加一个圆点构成项目

对于空产生式：A->ε仅有一个项目A->·

- 移进项：圆点后面为终结符
- 待约项：圆点后面为非终结符
- 归约项：圆点在最右端

- 接受项（特殊归约项）：如果一个项呈Z->u·形，且Z是识别符号

增广文法

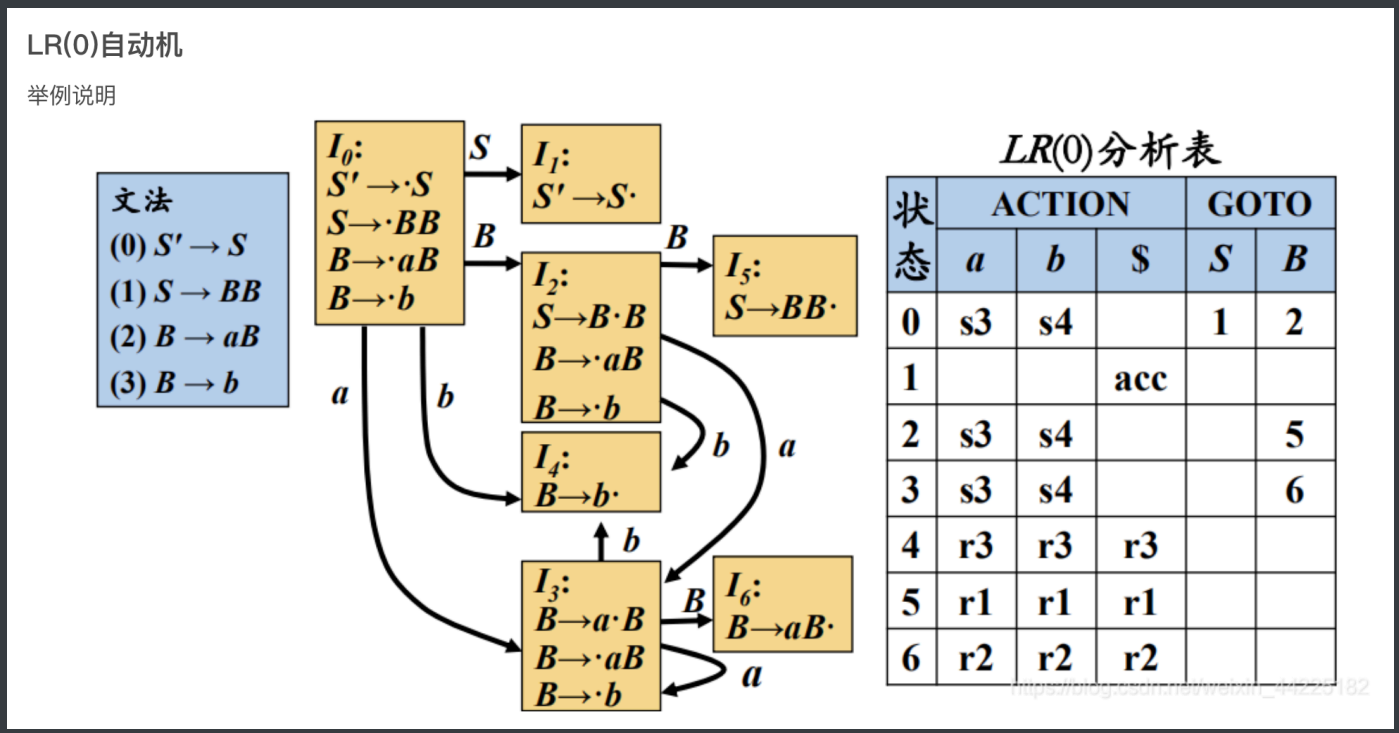
如果G是一个以S为开始符号的文法，则G的增广文法 G' 就是在G中加上新开始符号 S' 和产生式S' → S而得到的文法

后继项目

同属于一个产生式的项目，但圆点的位置只相差一个符号

A->a·Xb 的后继项目是 A->aX·b

LR（0）自动机



- I_0 -> I_1

  输入符号S，观察当前状态中所有圆点后的符号，恰好为S的入栈

- I_0 -> I_3

  输入符号a，其中B->·aB符合条件，入栈后变为B->a·B，由于圆点后面的是非终结符号B，因此要重新对该非终结符进行展开（即将I_0中左部为该符号的产生式入栈）

分析表中，如果下一个字符为终结符，则为ACTION；如果为非终结符，则为GOTO

其中ACTION可能为移进、归约、ACC

- 移进S：圆点后还有字符，接受该字符后进入什么状态，则为 S_i
- 归约R：圆点后无字符，该字符对应文法中的哪一条，即为 R_i

  - 此时I4已经是归约项目了，无论下一个输入什么字符，都采取**归约**操作。通过文法发现，I4中的B–>b·
    对应于文法中的（3）B–>b,所以（4，a/b/$）==r3 (r:归约 3：文法中第三个式子)【这里归约的时候，不用看后面GOTO，因为GOTO中填的只能是数字】
- ACC：如 I_1:S'->S· ，此时下一个字符为$时即为ACC

GOTO仅可能出现数字，表示当前状态接受下一个字符为当前非终结符时，将进入什么状态

分析表中可能含有移进-归约冲突或归约-归约冲突。如果LR（0）分析表中没有语法分析动作冲突，那么给定的文法就称为LR（0）文法

LR（0）分析过程

若对文法 G' 的产生式编号如下：

(0) $S' \to E$	(4) $A \to d$
(1) $E \to aA$	(5) $B \to cB$
(2) $E \to bB$	(6) $B \to d$
(3) $A \to cA$	

表 6.3　LR(0)分析表

状 态	ACTION					GOTO		
	a	b	c	d	#	E	A	B
0	S_2	S_3				1		
1					acc			
2			S_4	S_{10}			6	
3			S_5	S_{11}				7
4			S_4	S_{10}			8	
5			S_5	S_{11}				9
6	r_1	r_1	r_1	r_1	r_1			
7	r_2	r_2	r_2	r_2	r_2			
8	r_3	r_3	r_3	r_3	r_3			
9	r_5	r_5	r_5	r_5	r_5			
10	r_4	r_4	r_4	r_4	r_4			
11	r_6	r_6	r_6	r_6	r_6			

表 6.4　对输入串 $bccd$ # 的 LR(0)分析过程

步 骤	状态栈	符号栈	输入串	ACTION	GOTO
(1)	0	#	$bccd$#	S_3	
(2)	03	#b	ccd#	S_5	
(3)	035	#bc	cd#	S_5	
(4)	0355	#bcc	d#	S_{11}	
(5)	0355(11)	#$bccd$	#	r_6	9
(6)	03559	#$bccB$	#	r_5	9
(7)	0359	#bcB	#	r_5	7
(8)	037	#bB	#	r_2	1
(9)	01	#E	#	acc	

- 步骤1时，状态栈为0，符号栈为#
- 每次观察状态栈栈顶和输入串左侧符号，在分析表中找到对应的动作
- 进入下一步骤时，如果是移进，则ACTION动作中的下标进入状态栈，终结符进入符号栈；如果是归约，则GOTO动作中的数字进入状态栈，非终结符进入符号栈
- 归约时，需要从状态栈和符号栈中弹出对应字符

SLR（1）分析

基于容许LR（0）规范族中有冲突的项目集，用**向前查看一个符号**的方法来解决冲突。由于仅对有冲突的状态才向前查看一个符号，因此是**简单的**LR（1）分析法，即SLR（1）

假定一个 LR(0)规范族中含有如下的项目集(状态)I：

$$I=\{X\to\alpha\cdot b\beta, A\to\gamma\cdot, B\to\delta\cdot\}$$

也就是在该项目集中含有移进-归约冲突和归约-归约冲突。其中 α、β、γ、δ 为文法符号串，b 为终结符。那么只要在所有含有 A 或 B 的句型中，直接跟在 A 或 B 后的可能终结符的集合即 FOLLOW(A)和 FOLLOW(B)互不相交，且都不包含 b，也就是只要满足

$$\mathrm{FOLLOW}(A)\cap\mathrm{FOLLOW}(B)=\varnothing$$

$$\mathrm{FOLLOW}(A)\cap\{b\}=\varnothing$$

$$\mathrm{FOLLOW}(B)\cap\{b\}=\varnothing$$

那么，当在状态 I 时，如果面临某输入符号为 a，则动作可按以下规定决策：

(1) 若 $a=b$，则移进。

(2) 若 $a\in\mathrm{FOLLOW}(A)$，则用产生式 $A\to\gamma$ 进行归约。

(3) 若 $a\in\mathrm{FOLLOW}(B)$，则用产生式 $B\to\delta$ 进行归约。

(4) 此外，报错。

处理一个有冲突的项目集时，其中的每一个句型都会产生一个集合：

- 对于归约句，等式左部非终结符的Follow集合
- 对于移进句，圆点后面的非终结符

需要保证这些集合两两不相交

如果对于一个文法的LR（0）项目集规范族的某些项目集或LR（0）分析表中所含有的动作冲突都能用上述方法解决，则这个文法是SLR（1）文法

SLR（1）分析表

在LR（0）分析表里，ACTION中如果某一状态对应的一个非终结符为归约，那么该行一定均为归约操作，原因如下：

- LR（0）认为，当前语句为归约型时，不论下一个符号是什么，都采取归约操作

- 而SLR（1）中，对于A→α，仅当下一个符号属于 Follow(A) 时，才会在SLR（1）分析表中写入归约操作

LR（1）分析

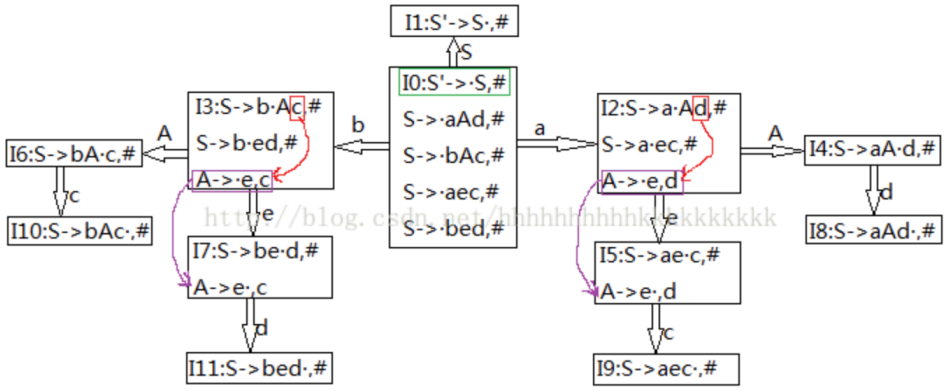
在SLR（1）中，只凭Follow集合判断是否采用归约不合适，SLR（1）依然可能存在语法冲突。

LR（1）项目由两部分组成，一部分为LR（0）项目，一部分为向前搜索符集合

初始S_0状态的项目集为：[S'->S·,#]

向前搜索符：

课本的答案是对的，但是写法很是让我们一头雾水，下面让我们来看看答案是怎么出来的：



1.一开始，绿框处S'是整个句子，所以后面理所当然跟句子结束符#。然后S后面是‘ε’（就是什么都没有）所以β=ε，接着逗号后面是‘#’即a=#，这样FIRST(βa)=FIRST(ε#)={#}。这就是I0中S后面#号的来历。

2.在I2中，如红线所示，在I2:S->a·Ad,#中A的后面是d,所以FIRST(d#)={d}（就是“d#”的第一个终结符d），所以接下来A的后面跟的是d。

在I_2中，有项目1:S->a·Ad,项目2:A->·e，其中项目2中的A是从项目1中得来的，因此需要看在项目1中A后面的字符串：d，以及项目1的向前搜索符#，两个拼接后得到{d#}的First集即位项目2的向前搜索符

LALR分析

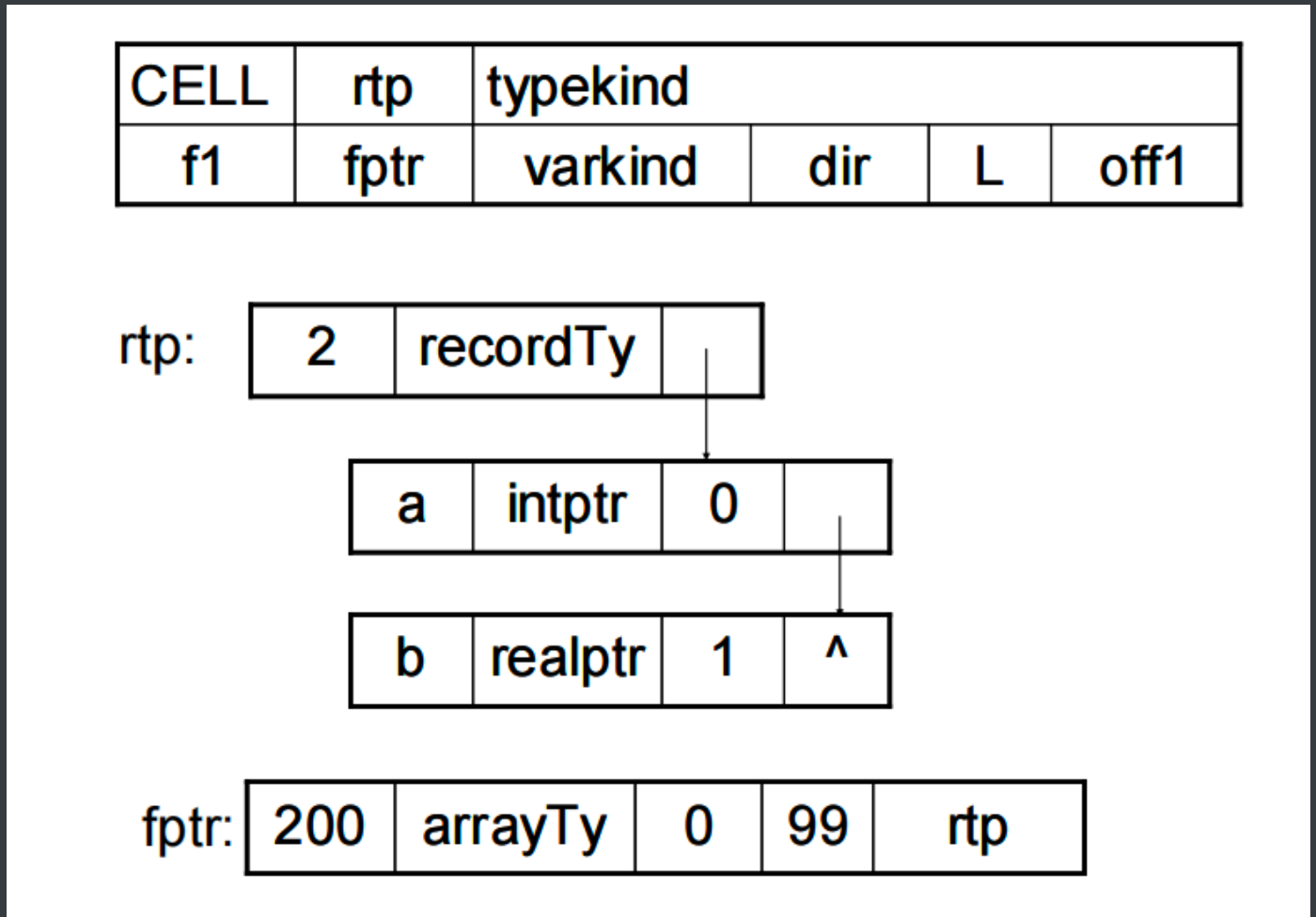
将LR（1）分析中的同心（除展望符外，两个LR（1）项目集是相同的）项目进行简化

没有归约-归约冲突

符号表与类型表

三、（10 分）设当前层数为 L，可用偏移量值为 off1，且有下面 C 程序段，试完成相关的符号表和类型表。

```
typedef struct {
        int a;
        float b;
  } CELL;
CELL f1[100];
```



源程序的中间表示

四元式中间代码

- 算式法

四元式：（操作符，左操作数，右操作数，结果）

三元式：（操作符，运算对象1，运算对象2）

四、（10 分）试写出下列表达式的四元式（或三地址代码）中间代码，假设运算符优先级由高到低依次为-(负号).*./.+.-，且均为左结合。

$$x= -(a*b)+(c+d)-(a*b-c)$$

(1) (*, a, b, t1)

(2) (-, t1, _, t2)

(3) (+, c, d, t3)

(4) (+, t2, t3, t4)

(5) (*, a, b, t5)

(6) (-, t5, c, t6)

(7) (-, t4, t6, t7)

(8) (=, t7, _, x)

- 程序法

3、设有 C 语言部分程序：

```
{
    int   i;
    int   max;
    i=1;          max=1;
    while (i<=10)
      {
        max=max*i;
        i=i+1;
      }
}
```

将其中的语句翻译成四元式序列。

```
(MOV,1,i)                (MOV,1,max)
(LABEL, inL)             (<=,i,10,T1)
(JUMP0,outL)             (*,max,i,T2)
(MOV,T2,max)             (+,i,1,T3)
(MOV,T3,i)               (JUMP,inL)
(LABEL,outL)
```