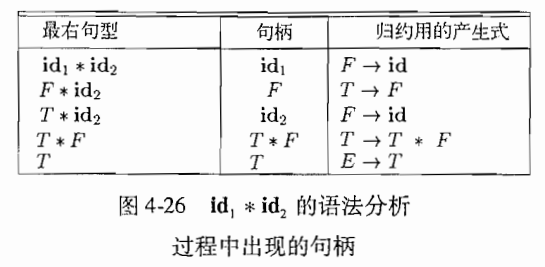
编译原理复习 #4 Bottom-Up Parsing

**自底向上的语法分析**

***自底向上的语法分析过程****对应于为一个输人串构造****语法分析树****的过程*

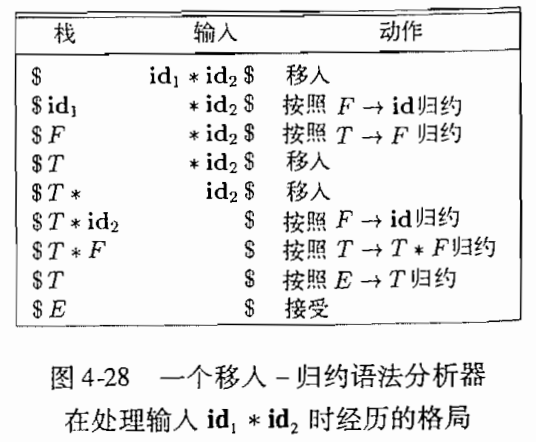
**归约（reduction）：**一个与某产生式体相匹配的特定**子串被替换为该产生式头部**的非终结符号，在自底向上语法分析过程中，关键问题是何时进行归约以及应用哪个产生式进行归约

**句柄（Handle）：**和某个产生式体匹配的**子串**，对它的归约代表了相应的最右推导中的一个反向步骤



**LR(0)**

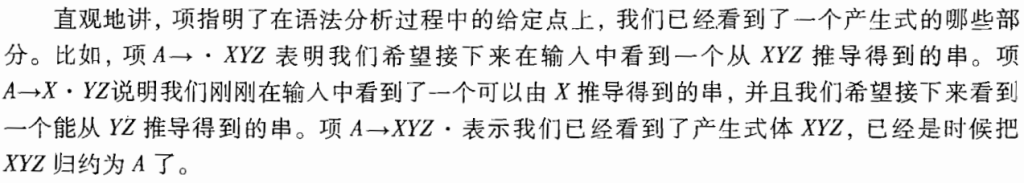
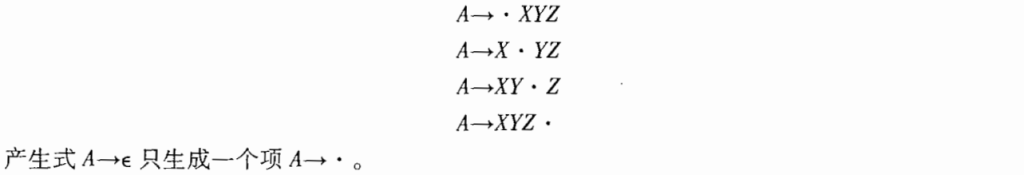
**语法分析器**将零个或多个输入符号**移到栈的顶端**，**直到**它**可**以对栈顶的一个文法符号串进行**归约**为止。分析器可采取如下四种可能的动作：**①移入，②归约，③接受，④报错**



**可行前缀（viable prefix）**：一个可行前缀是一个最右句型的前缀，并且它没有越过该最右句型的最右句柄的右端（注意不是说一定要有到句柄那里那么长），**在移入-规约中栈中的一定是可行前缀**

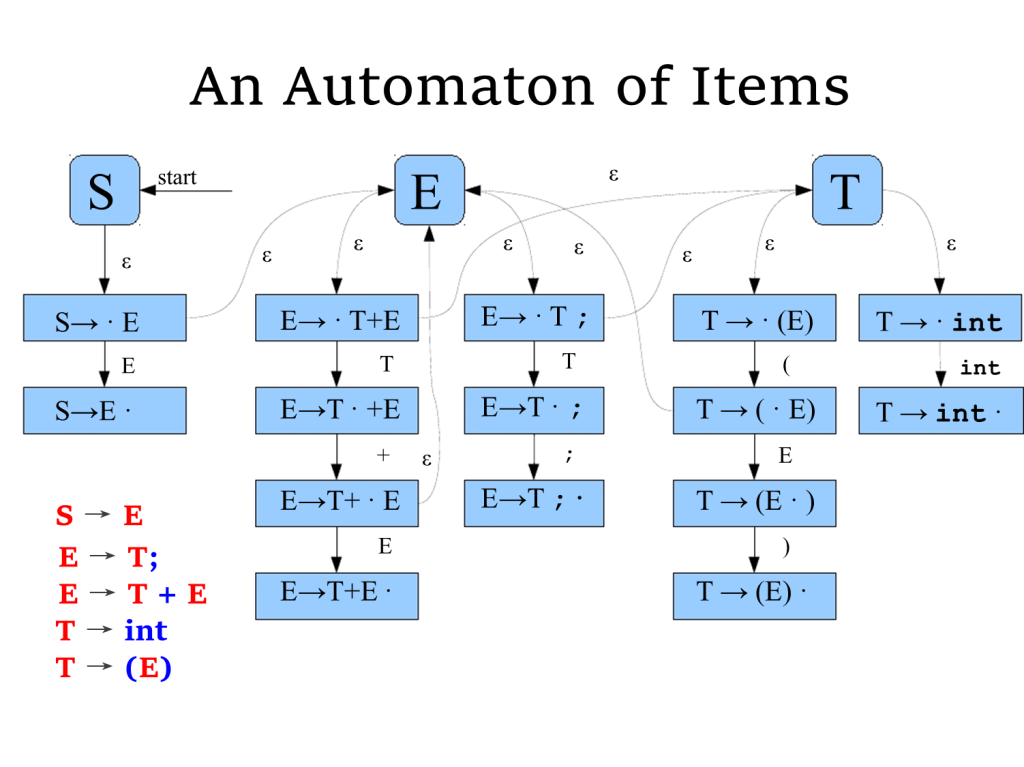
*有趣的是自上向而下的处理时当前产生式的 action 结果立即就体现在了栈中，而自下而上的 action 结果在下一个 pass 才会出现在栈中*

**项（item）：**一个产生式再加上一个位于它的体中某处的点

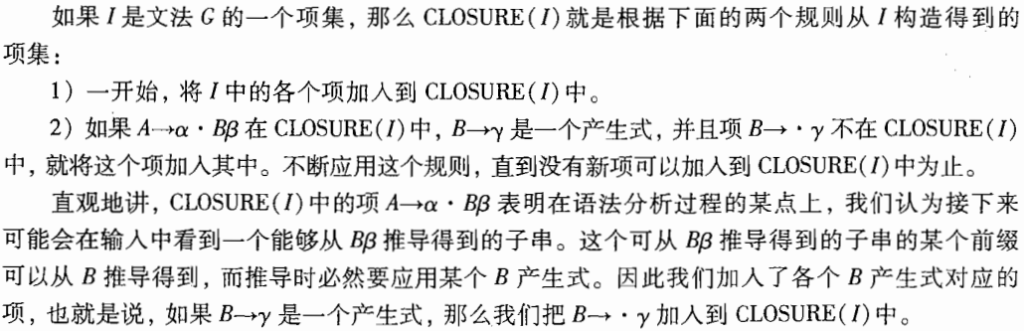


**The LR(0) items can be used as the states of a finite automaton**that maintains information about the parsing stack and the progress of a shift-reduce parse

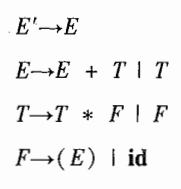
**由项构造 DNF：**



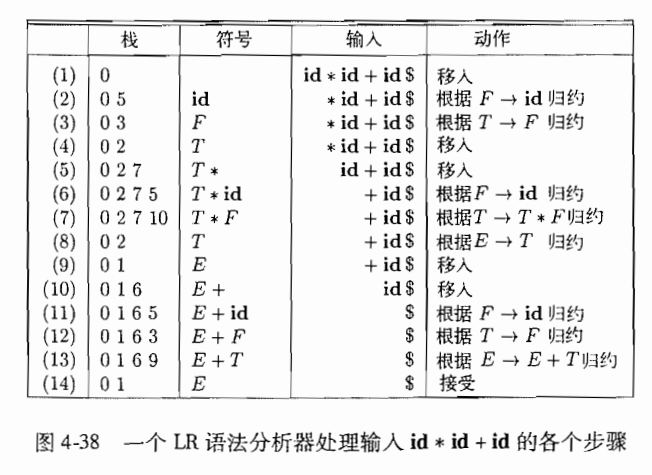
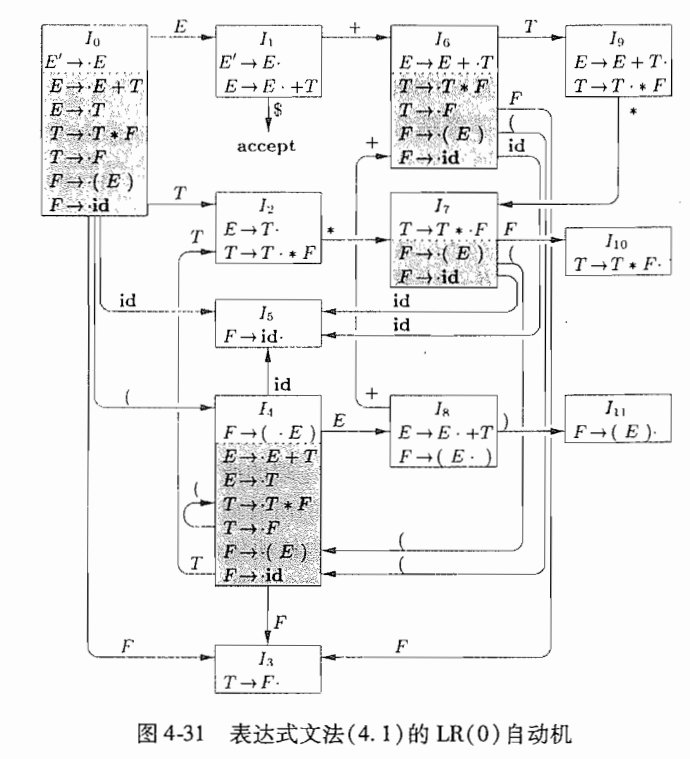
**使用 LR (0) 自动机的移入-归约语法分析器分析 id \* id**



**增广文法：**就是在原本的 E 上面加一撇放在最前面

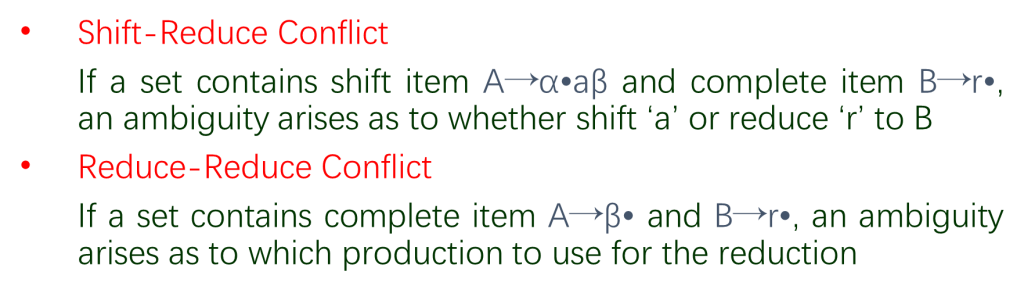


**增广文法后构造 DNF：**



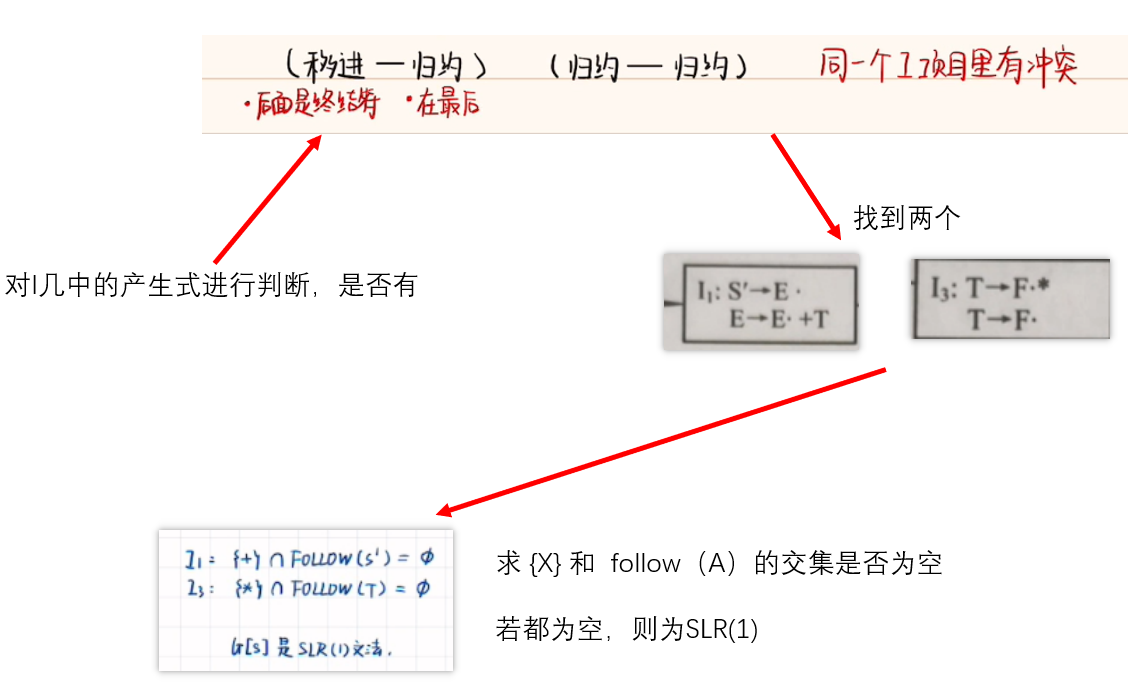
**简单 LR 技术（SLR）**

**LR(0) 由于没有查看处理处右边的句子所以有时候会存在冲突\***

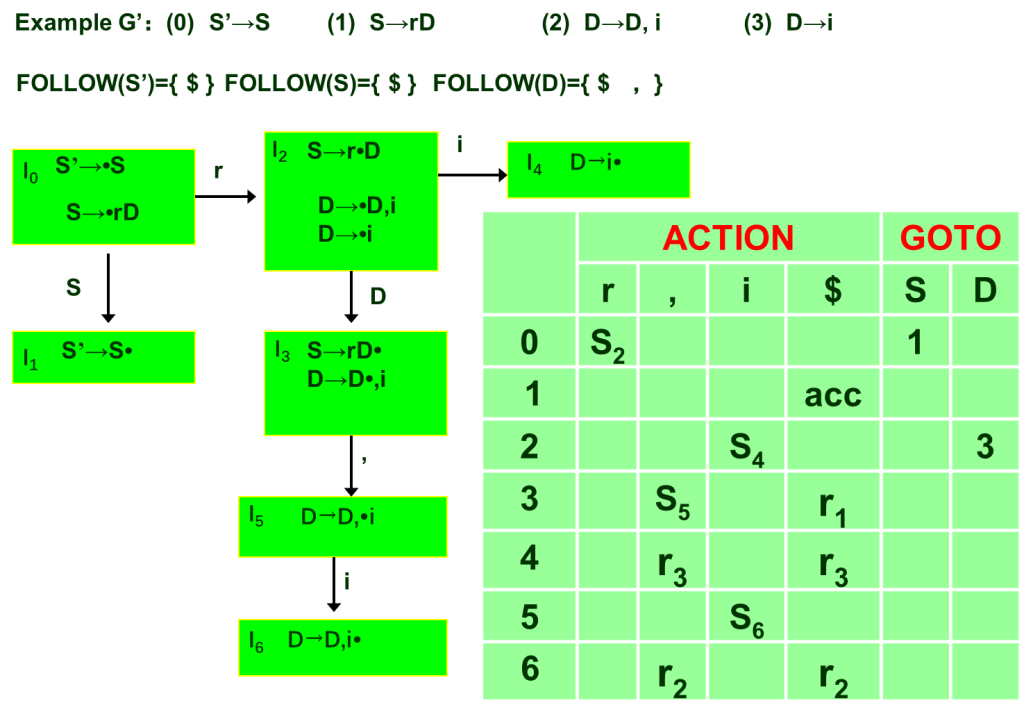
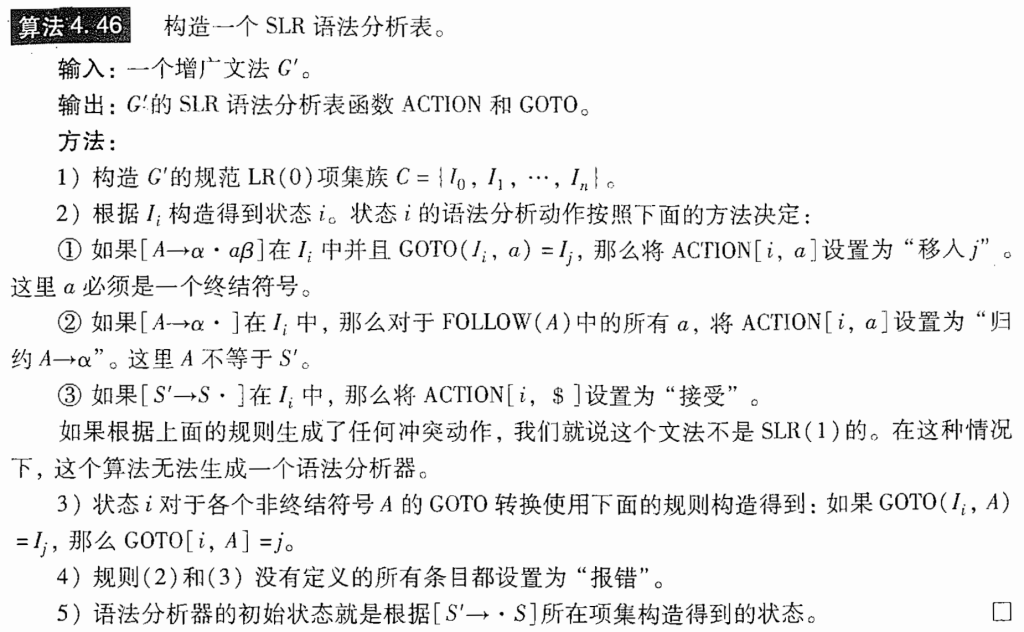


*通过观察后面的字符就可以在 LR(0) 中无缝升级 SLR(1)*

**SLR(1) 如何判断\***



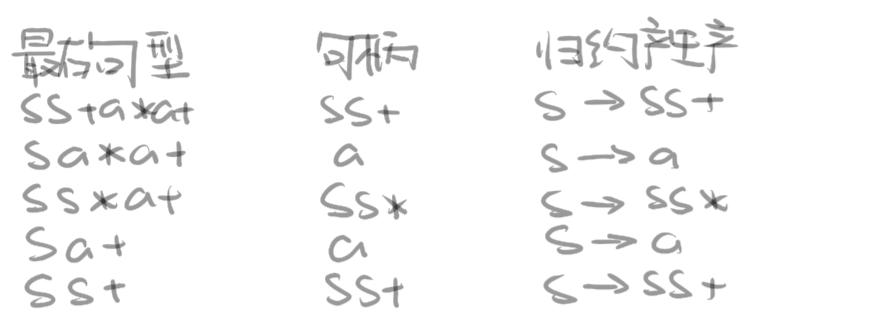
**构造 SLR(1) 语法分析表**



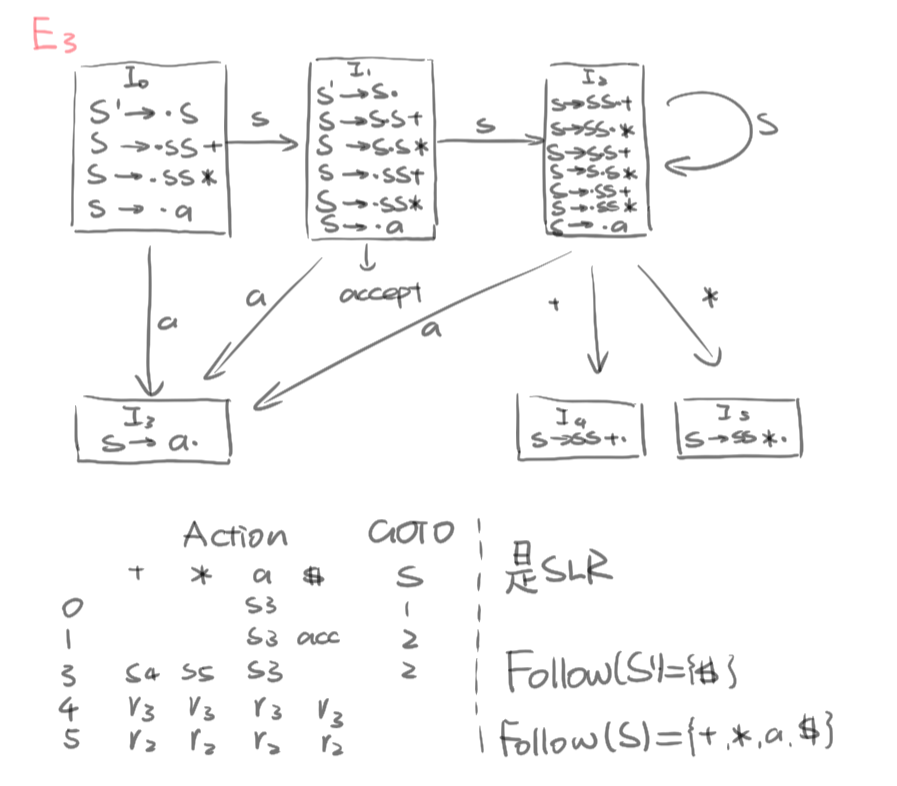
* 下一个字符在点的后面则 shift，后面跟的数字是进入的状态
* 如果点已经在最后面的时候下一个字符是 Fllow 中的字符则 reduce，后面的数字是表达式序号
* Action 对应于终结符，Goto 对应于非终结符

**练习题\***

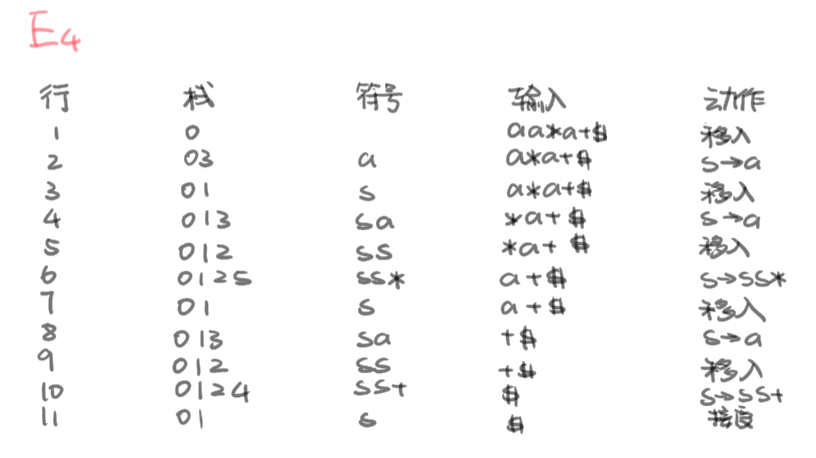
* **对文法 S -> SS+ | SS\* | a，求出右句型 SS + a \* a + 的句柄**



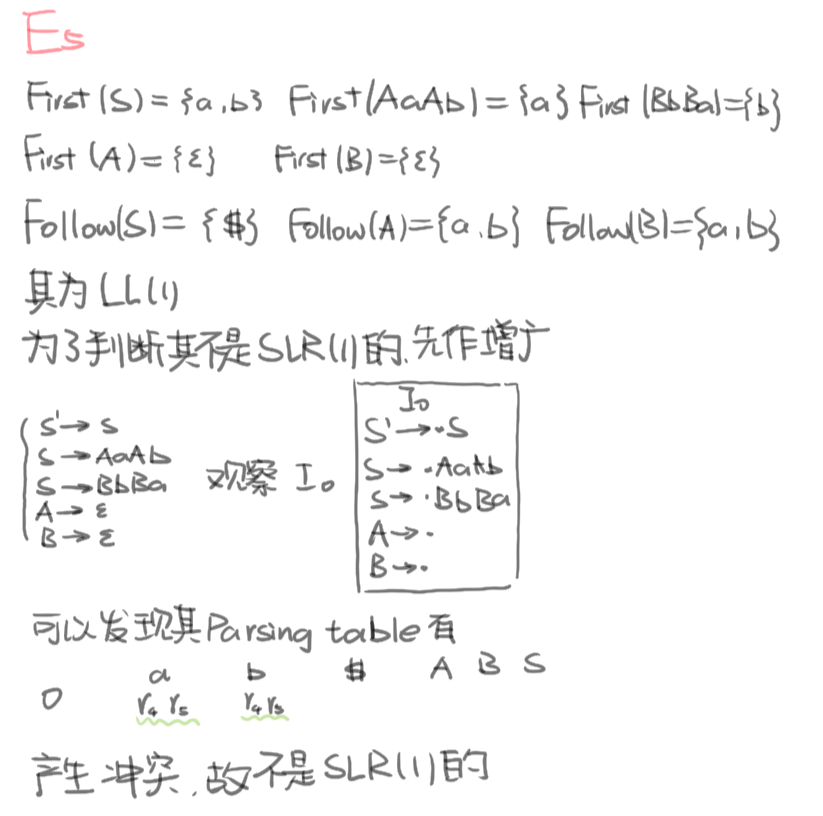
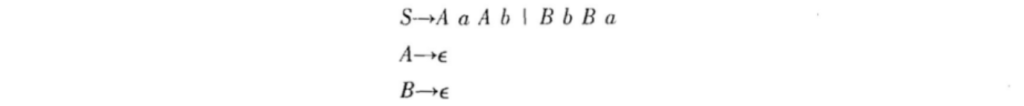
**对文法 S -> SS+ | SS\* | a， 求出 Follow set、求出该文法基于 LR(0) item 的 DFA、Parsing Table，并判断是否 SLR(1)**



**针对输入串 aa \* a +，依据上述所求 parsing table，构造表格完成解析过程**



**如下面文法，证明其是 LL(1) 但不是 SLR(1)**



* **规约 – 规约：**发现有两个可以规约项，那么他们的 Follow 不能相交
* **移入 – 规约：**当前规约的 Follow 不能和其他的 First 相交

