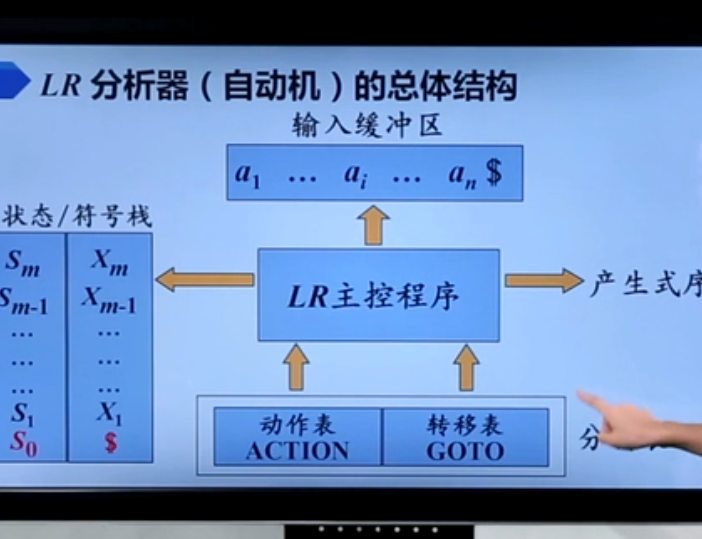


子地向下关键

•用来表示在什么状态，



LR分析器由一个输入带，

一个包含分析表的主控程序

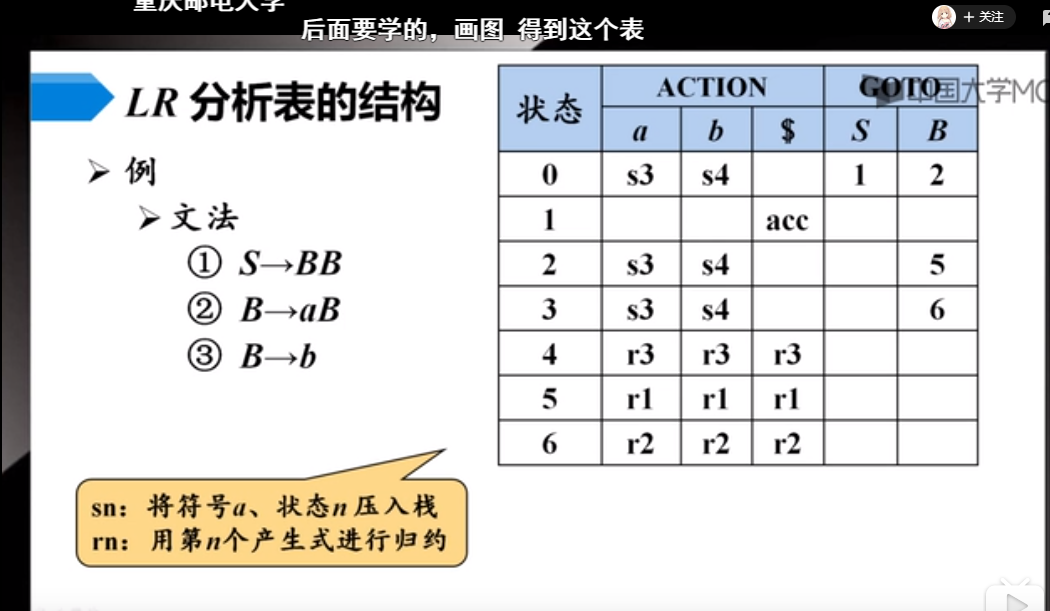
一个符号栈组成

符号栈由状态与符号构成

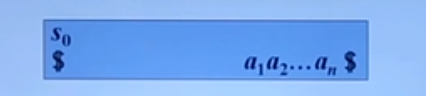
分析表有两部分构成一个是ACTION一个是转移表TODO

SN是SHIFT转移

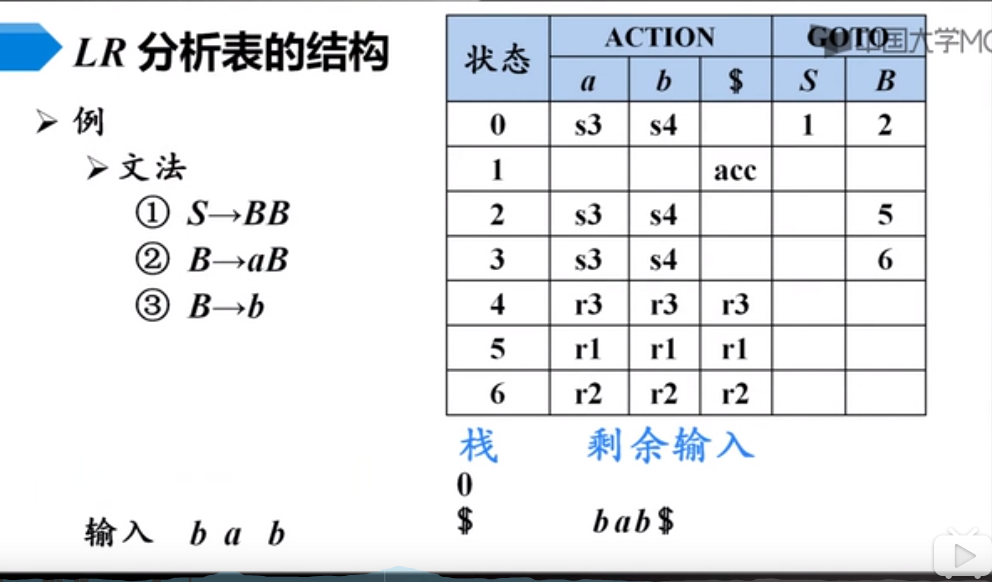
RN是reduce归约



GOTO表中每一列对应着一个非终结符，这里是SB，所以GOTO有SB



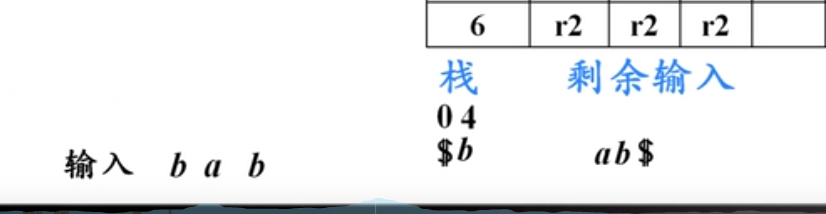
STEP1: INPUT BUFFER PUSH DOLLAR,push所有input符号



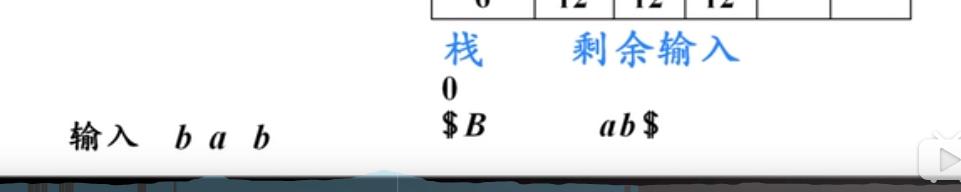
一开始符号栈是0dollar

根据状态栈0，以及当前输入小b， 我们决定s4，是移入移入动作

将小b移入到栈中，进行状态4

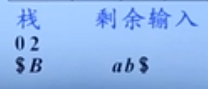


4与a进行r3,进行归约动作//第三条rule

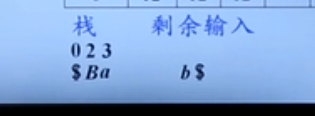


从右往左产生一个大B

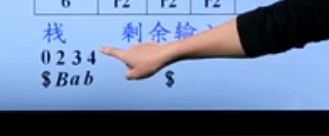
这时你会发现是不对称的，所有要check GOTO表，进入2号状态



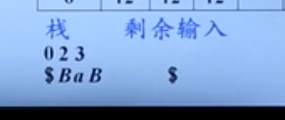
2与a，移入，进入3号状态



3遇到b，移入，4号状态

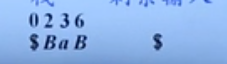


4号遇到dollar,归约

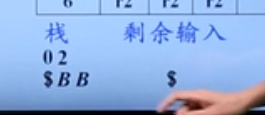


然后进入goTo,

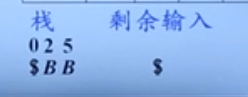
3遇到B，进入6



6遇到dollar，进行r2，aB转换成B //aB的状态也要同时移去

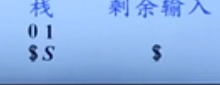


2B unMatch, goto,5

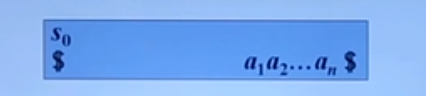


5与dollar，r1，进行rule1, BB转化成S，状态25移去

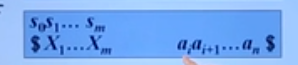
S遇上0，goTO1



1碰上dollar,action accept

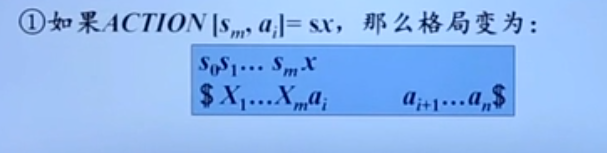


STEP1: INPUT BUFFER PUSH DOLLAR,push所有input符号



一般情况下，上下长度一样，

状态sm与ai匹配

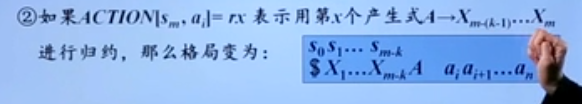


如果是sx shift

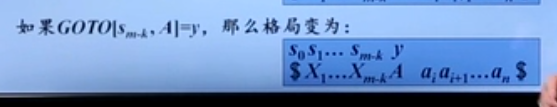
那就移入ai, 同时状态加上X

如果是rx，归约，

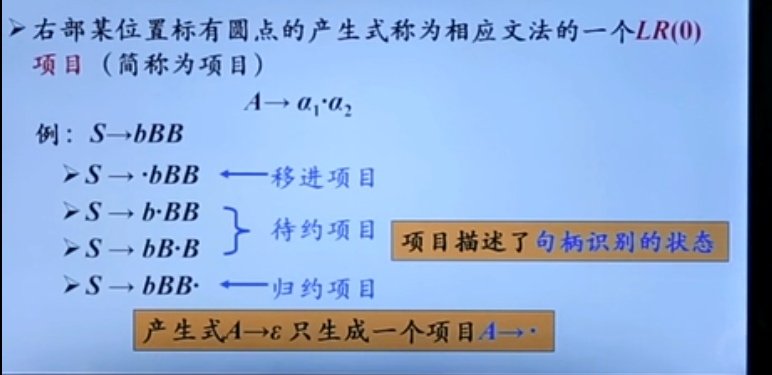
一直把下面的符号栈替换掉，同时替换对应状态栈，然后将左边记录到符号栈中，



并根据站定状态与A，补上对应状态



右边某位置标有圆点的产生式称为相应文法的一个项目



terminal是移进项目

non-terminal是待约

最后是归约

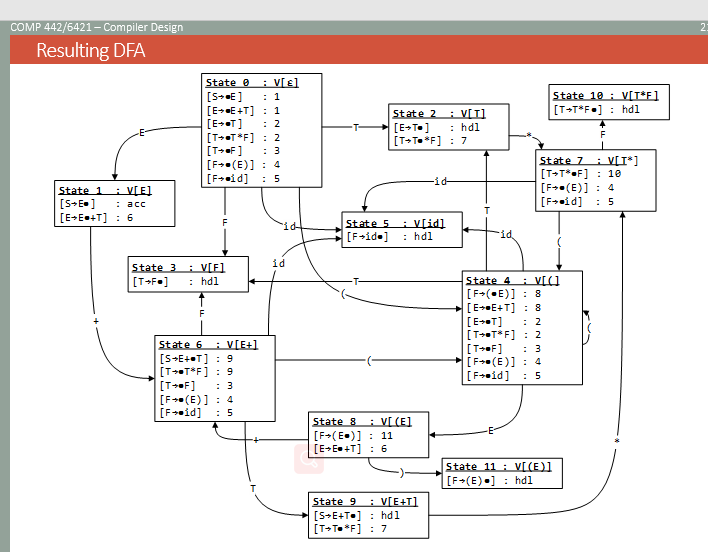
shift action实际上代表着点右移一格，进入到DFA中的一个新状态

reduce action代表着我们已经到了当前rule的底端，一旦handle到了底端，右边的所有element都应该被从stack上移走，并且用左边symbol代替， 然后我们跳刀下一个state，对应着下一个handle被处理，对应着table中的GOTO

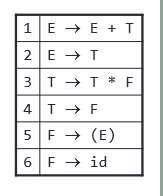
Initial Iterm: 一个dot为起点的item

Completed item:一个以dot为终点的item， 当我们有一个completed item，handle就会到底，然后我们把handle一直减到LHS non terminal

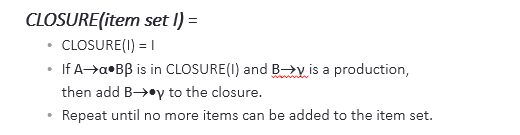
从state 0开始， 计算closure。 我们会发现所有可能state transition从



先把初始项目加上S->E 放入i0

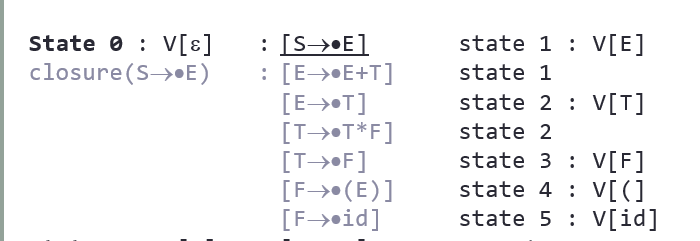


closure计算



如果句柄右边是一个非终结符，

那么把B->xx加入closure，一直重复



例如·这里

句柄右边是E

加入E->•E+T

E->•T

右边是T

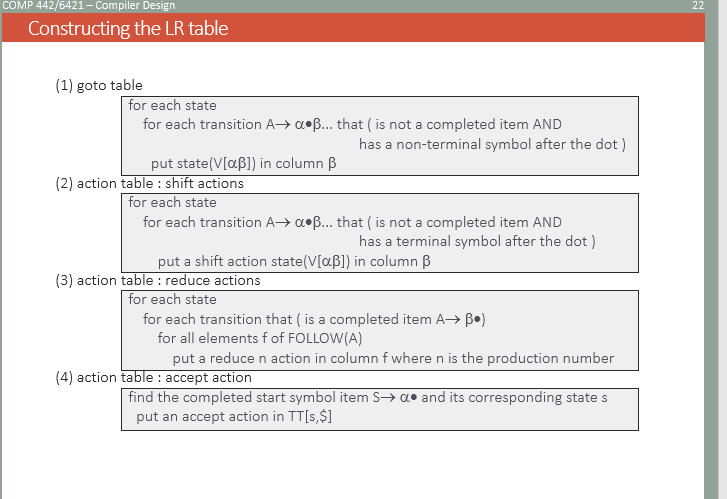
T=>•T\*F

T->•F  
F->•(E)

F->id

建立好了closure，我们需要知道这个state能到那些state

.

如果右边是non-trminal且not completed

把V【aβ】放在格子β里

shift

如果右边是terminal

把shift放在格子β里

如果completed了，找到当前rule, reduce当前rule，

如果