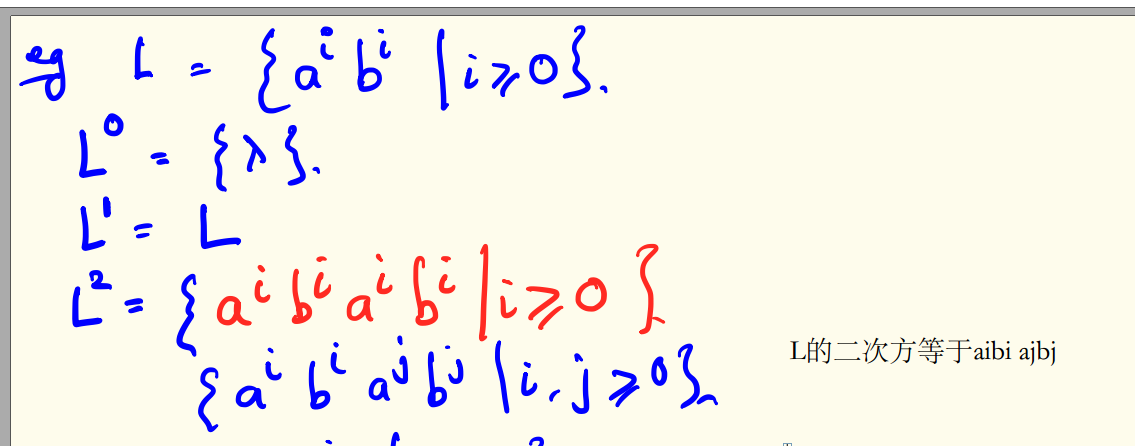
concatention 串联，

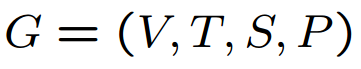
prefix前缀，

suffix 后缀





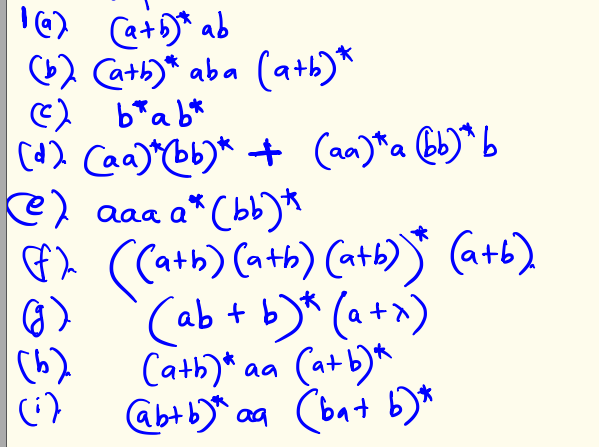
grammar

.

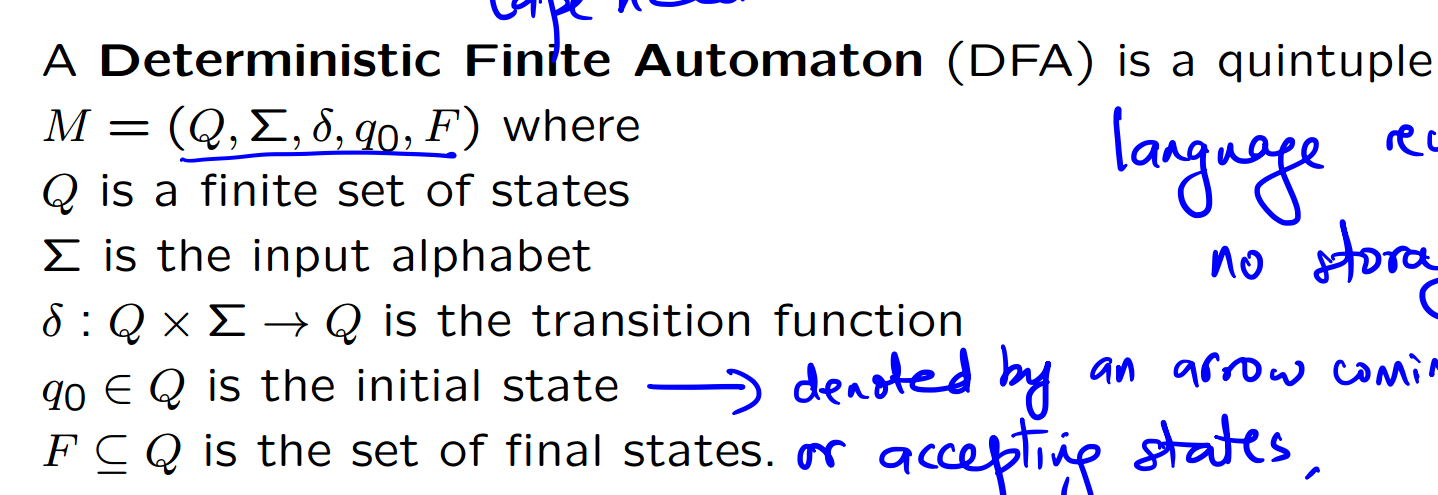
V是有限变量集，T是有限大写字母terminal集，S是开始变量,P是有限production/规则

基本表示法：

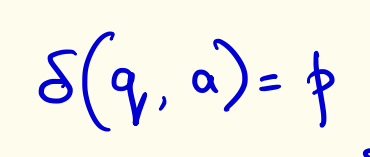


注意regular expression没有具体的几次方

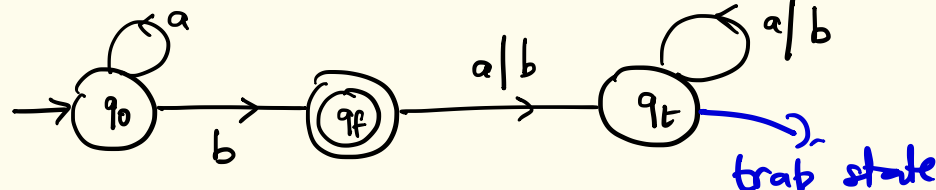
DFA



Q是状态集，alphabet集，路径，初始state,FINAL STATE



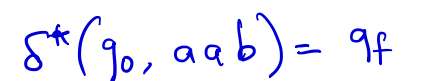
在state q，输入一个a，得到p

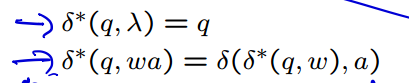


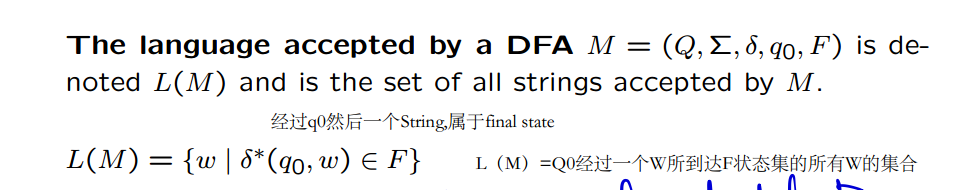
aib,

在DFA中，，每一个state要考虑a b两种情况，哪怕final，所以要加trap state

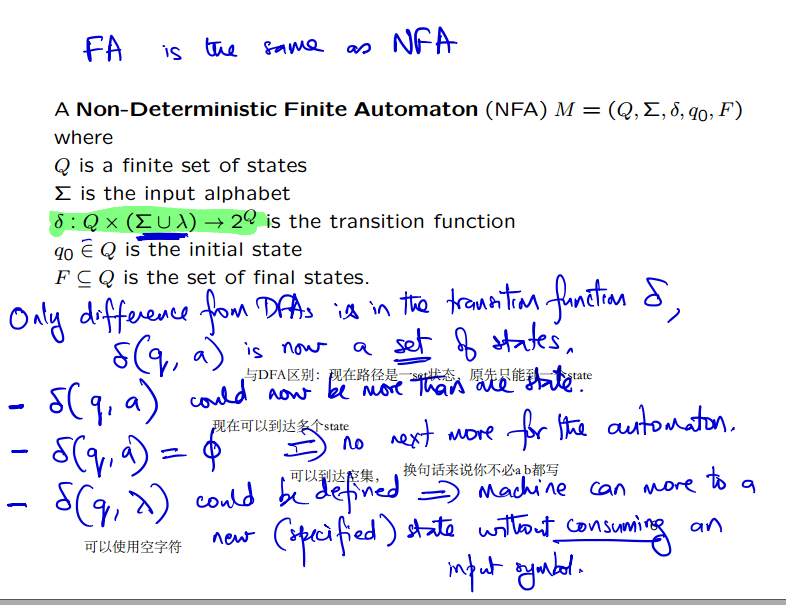
extended transition function

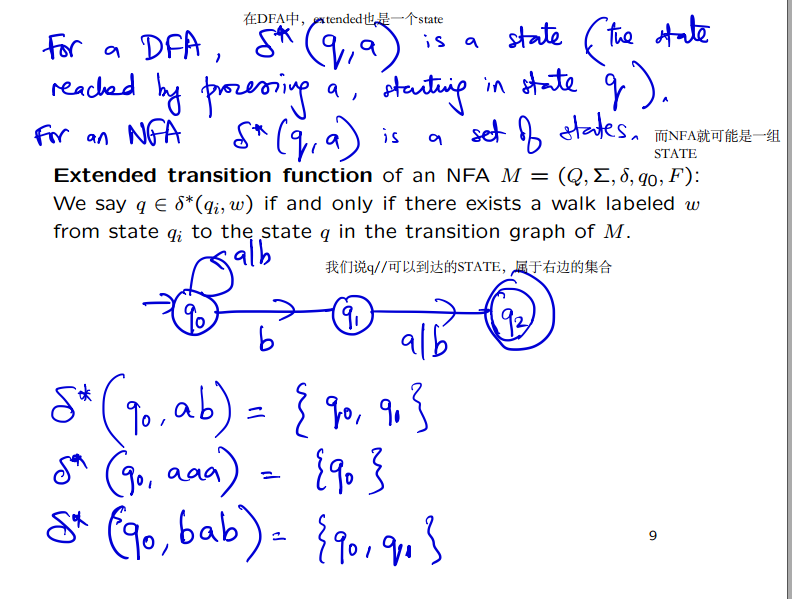




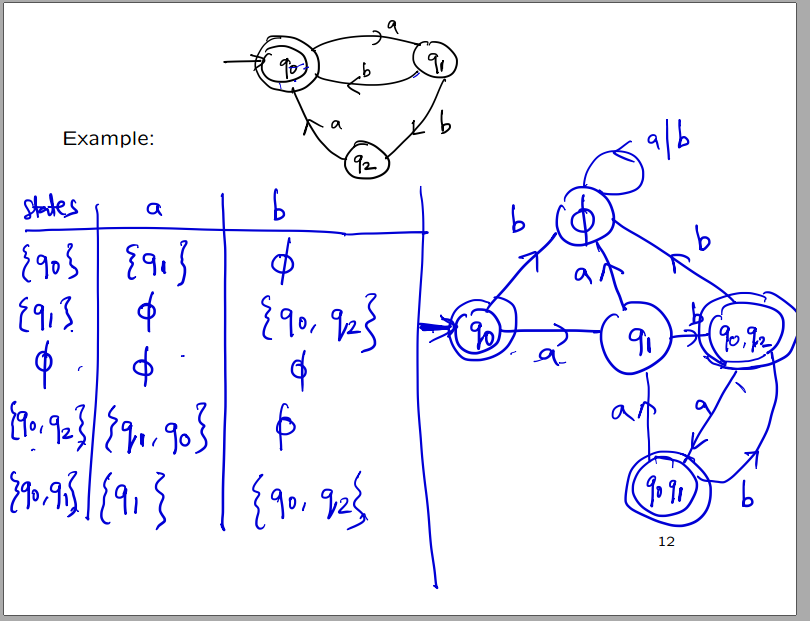


DFA接受的String，经过w到达final

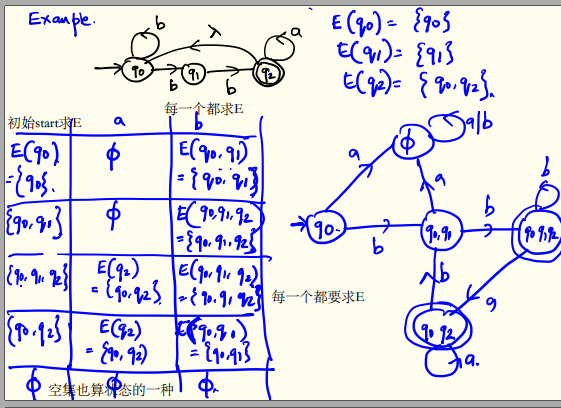




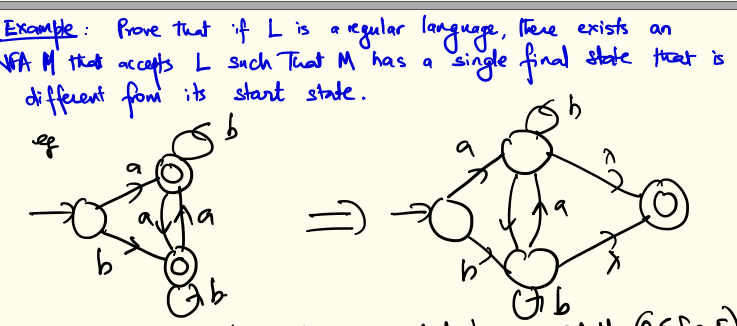
怎么把NFA转换成DFA：列表，从q0开始经过a能到达的所有集合，创建一个新state，空集也算一个state



如果有λ路径：要对每个路径再E一次 ，既如果经过λ有新的，要加入

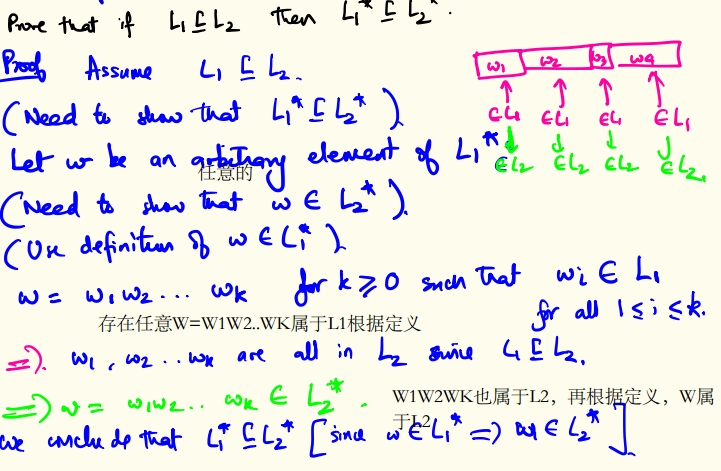


如果L是个regular，那么必然存在一个NFA M接受L，这个M存在一个final state并且与start state不同



原因，如果L是regular，那么必然存在一个DFA，这时我们可以创造一个NFA，让DFA的final state都伸出一条λ线到新final

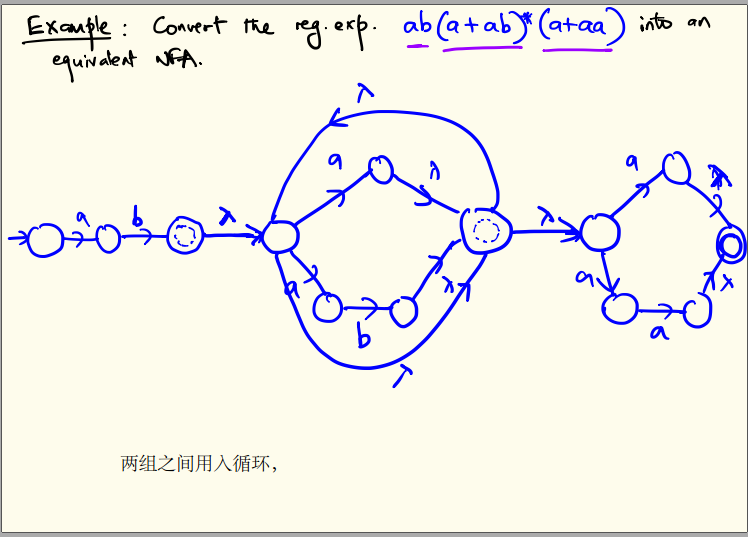
如果



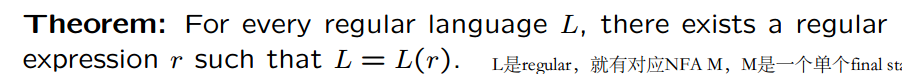
L1星的任意string可以表示为，由l1的w组起来的串联，而这些小w右属于L2，根据定义，他们都属于L2星.

换句话说任意L1星的STRING属于L2星

怎么把regular expression转换成NFA ，两组之间用循环，加号用空集



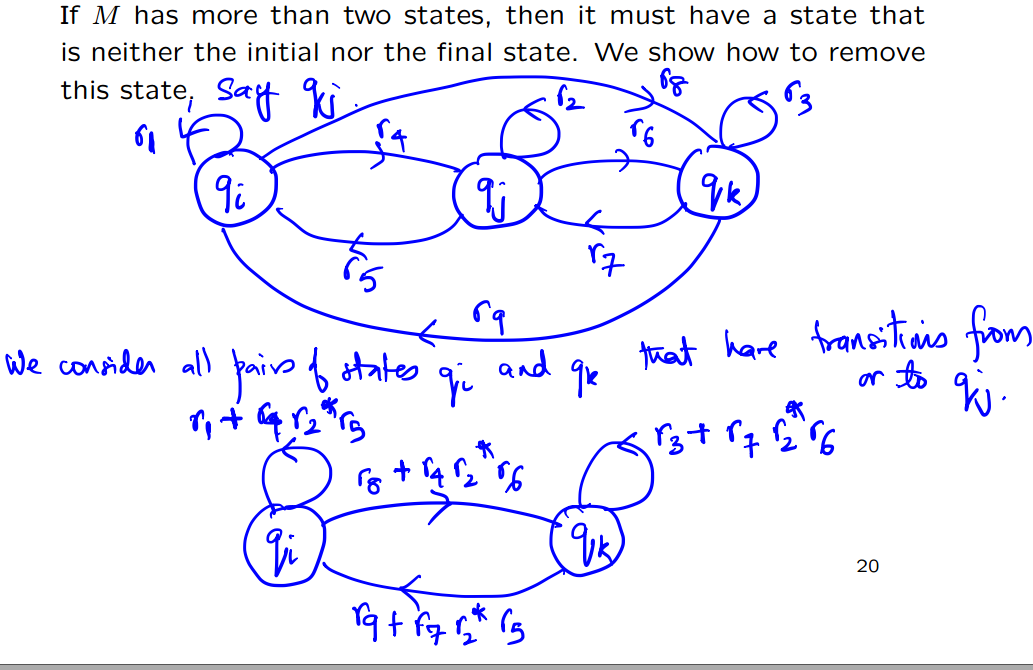
对于每个regular language，必有对应expression r



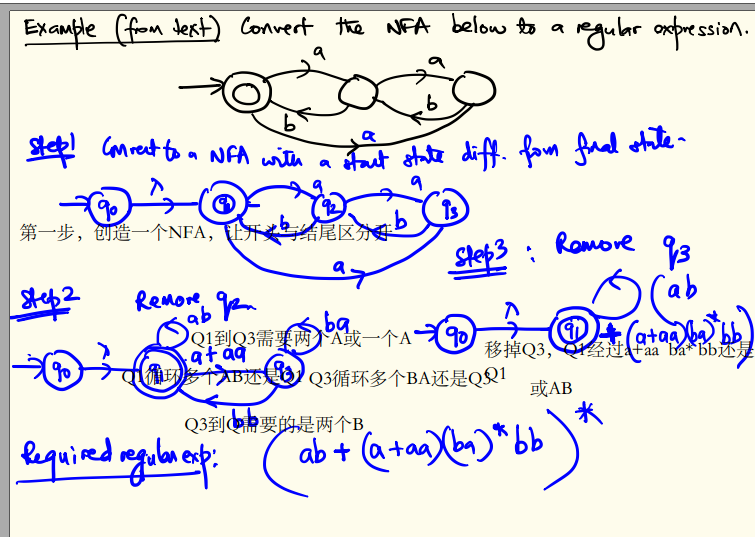
因为我们可以吧L转换成一个有唯一final state的NFA

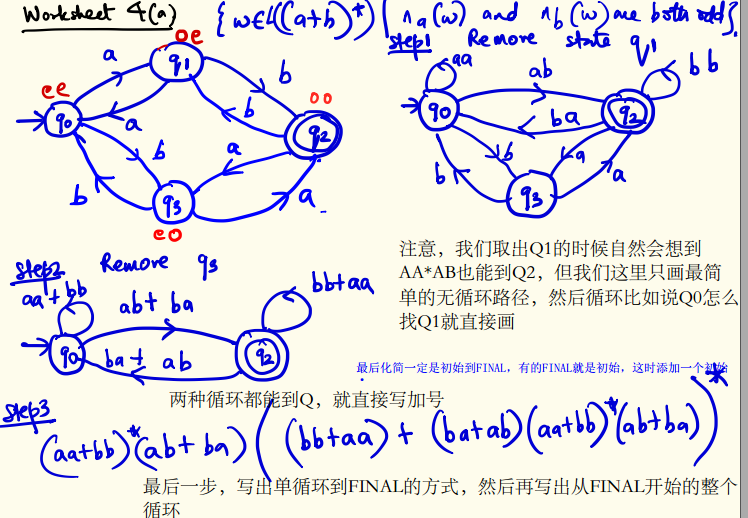
然后可以把这样的NFA转换成regular expression

：转换过程



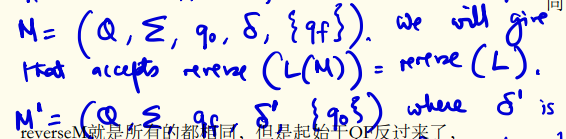
第一步：创造一个NFA，让开头与结尾明显区分，并且只能有一个final state,然后尽量缩减通过remove某个非final非start状态



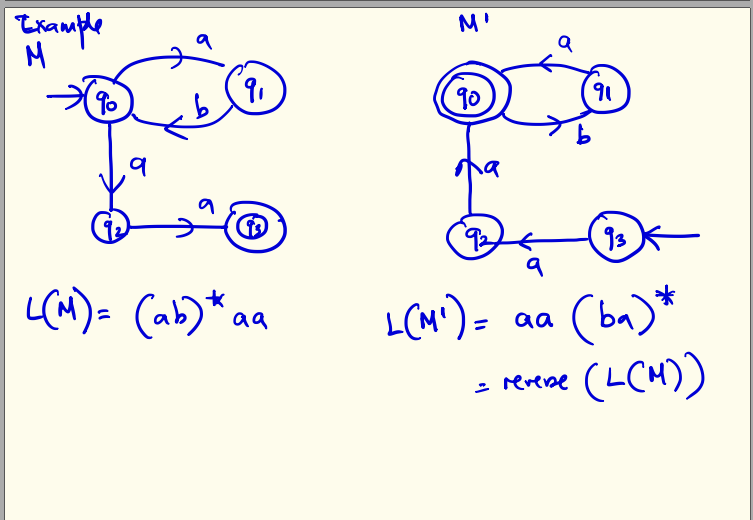


最终一定是初始到final，只剩初始到final的时候就可以表达了

如果L是regular的，那么reverse L也必然是regular



你只要把q0 qf互换，然后路径翻转就行



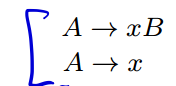
方向翻转，初始与final翻转

注意补给和reverse的区别，reverse只要翻转start与final并改变箭头方向，起始变了

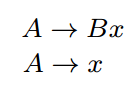
而补集并不改变箭头方向，只是把一个圈改成两个圈，两个圈改成一个圈（起始不变）

regular grammar

right-linear

x可以是一个小写字母也可以是一个小写字母串分为两种情况甚至是空集，但是大写字母只能有一个

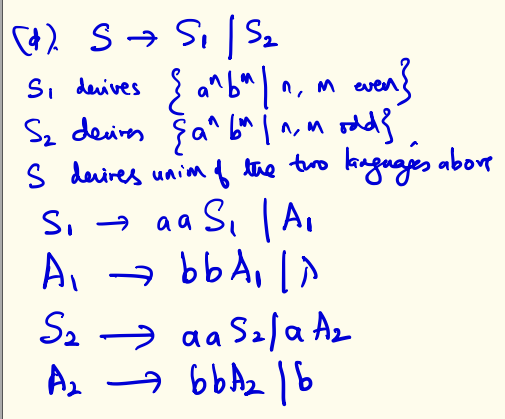
left-linear



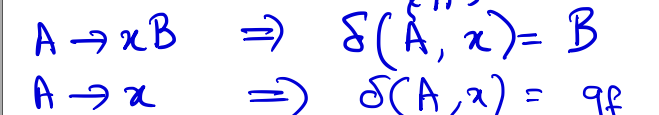
如果一个GRAMMAR 是right linear/left linear，就是regular grammar

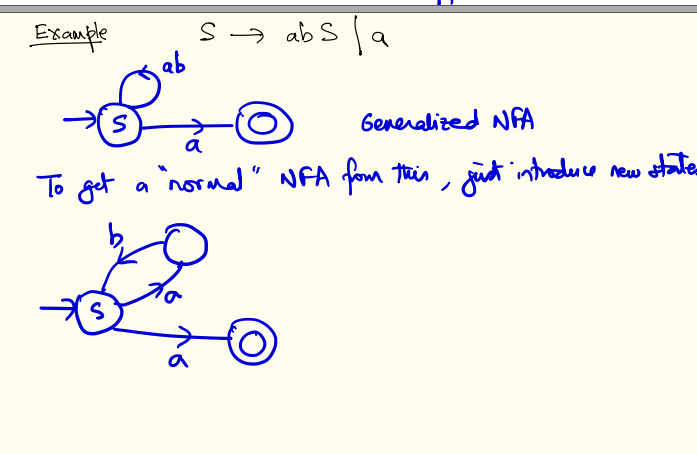
n+m是even

分为两种情况nm皆为odd，nm皆为even

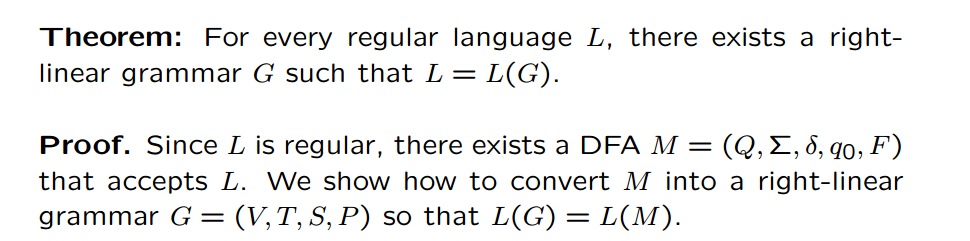


每一个right linear 都有一个NFA，





我们最终只要把ab化开来就行了

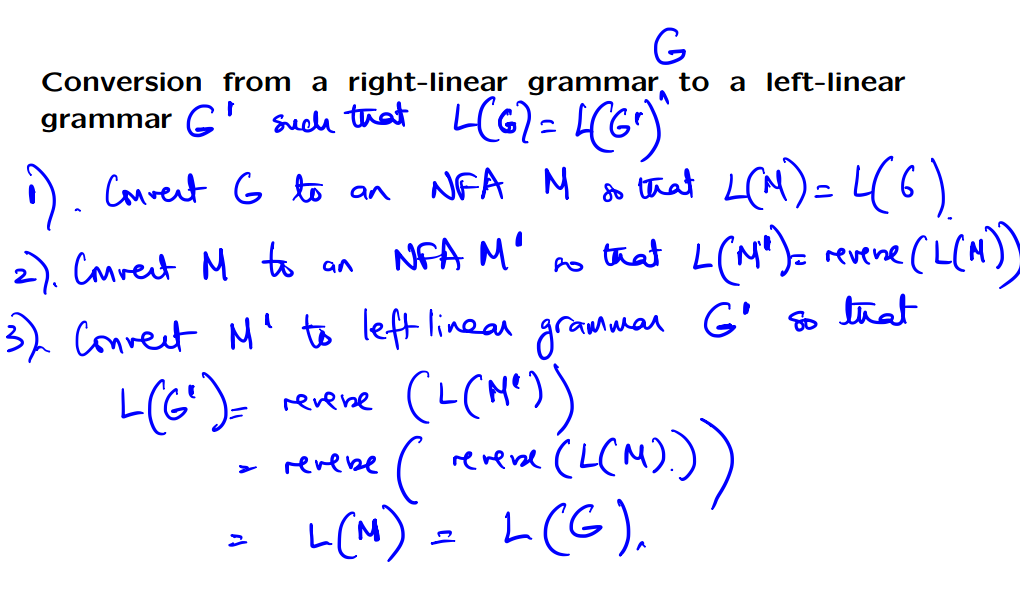


对于任意L，总能找到right linear grammar

因为他必定能转化成对应DFA，

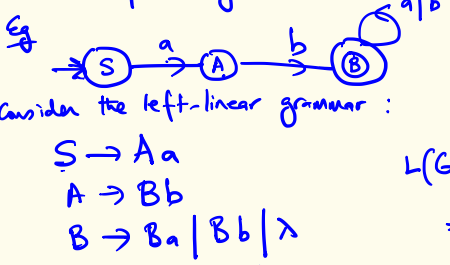
我们只要把路径些出来就行

怎么把right linear转换成left linear重要

把G转换成NFA M，

然后把Mreverse，既start final互换，箭头翻转

然后M'用left linear表示



closure性质

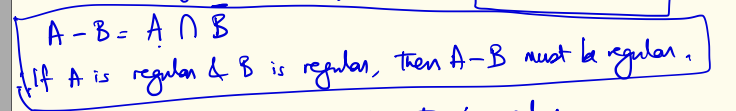
L1 L2为regular那么

并联：可以制造NFA

串联：同上

补集：可以制造DFA，DFA翻转生成新DFA，代表补给

交集，均是regular，大摩根



wwr:

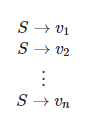
ambbam

xyz,xy<m,y>=1

a^m-ibbam

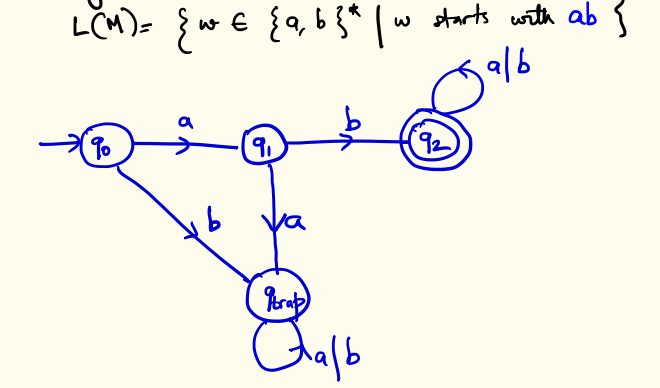
每个finite language都是regular的

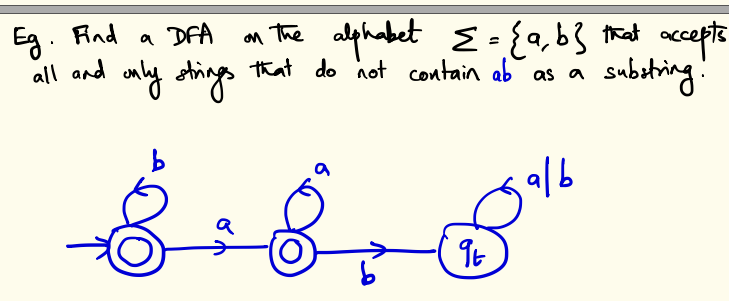
因为他们可以写成以下形式



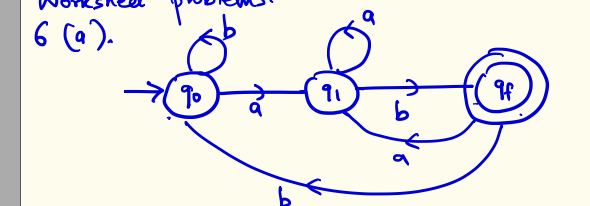
例子合集

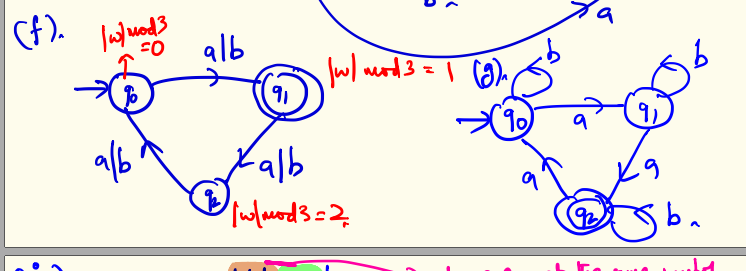
dfa





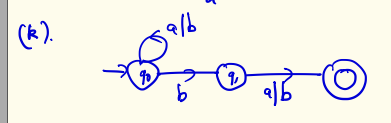






NFA

倒数第二个是b



不是regular

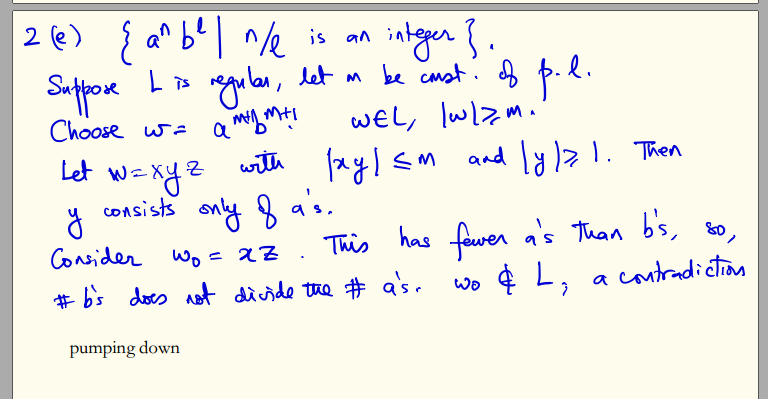


.









我们可以设置的野性一点

复杂关系

双向关系

