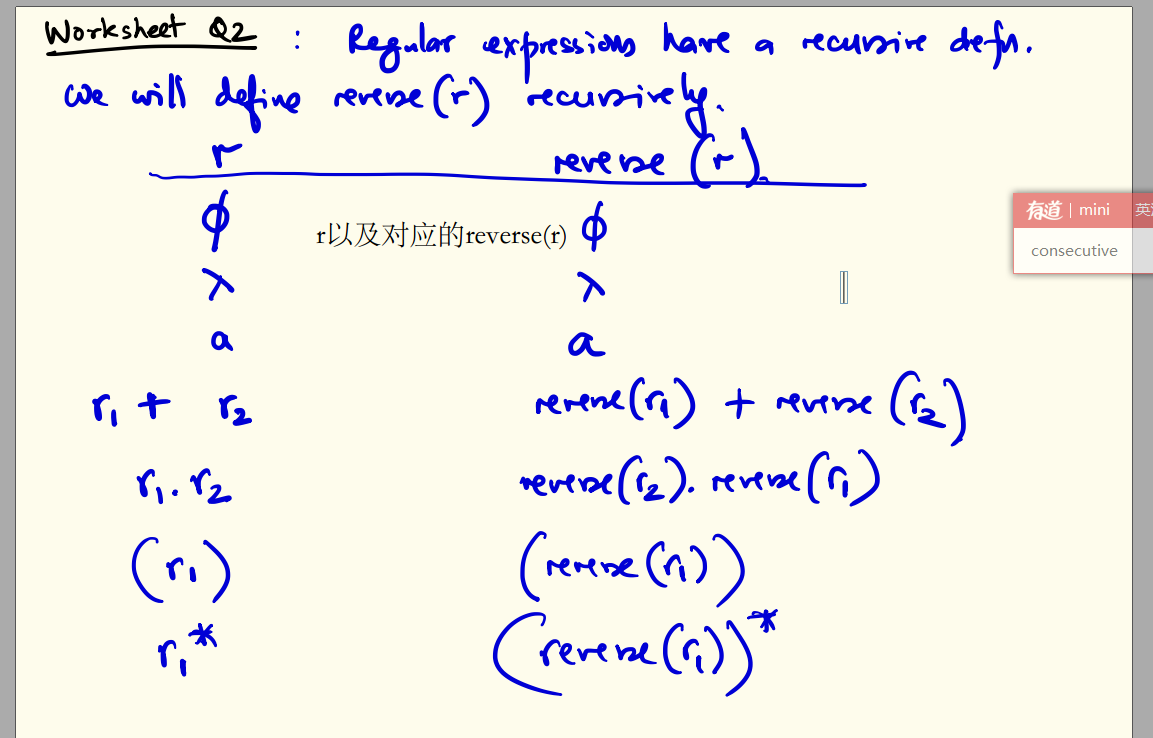
regular expression 表达even （aa）\*，写两遍就完事儿了

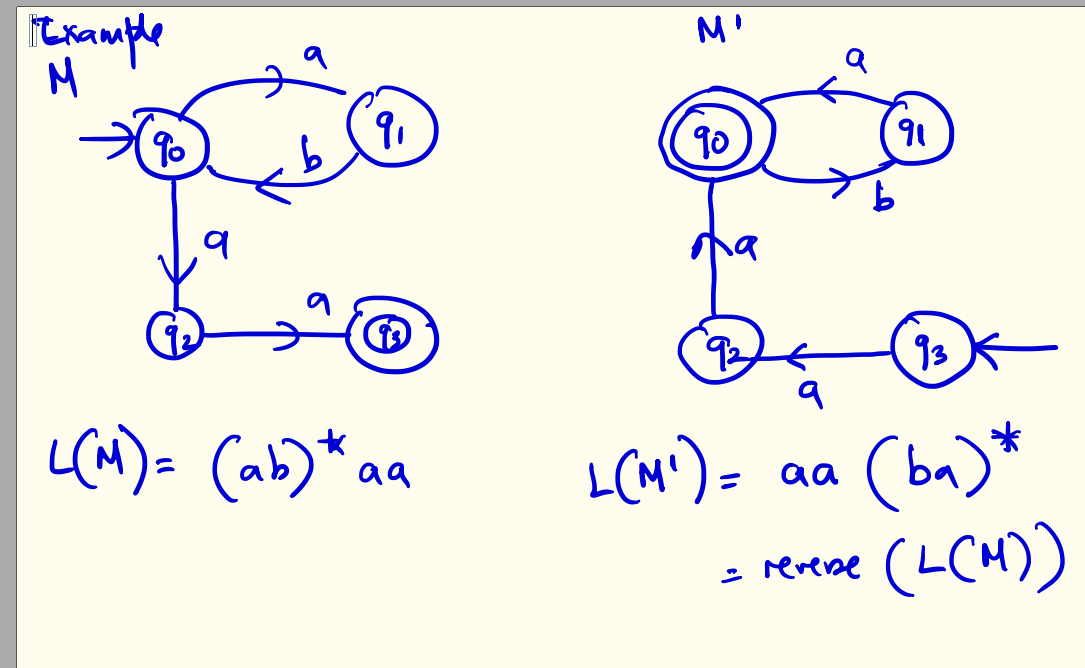
没有连续的A， abababa,bbbbba,

(ab+b)\*(a+b)

构建连续的a，左边ab+b\*,这样就是ababababbbbbbb，右边ba+b\*这样就是babababababbbbb



怎么reverse r:先把它转换成只有一个final的NFA,然后所有箭头倒转，START改成FINAL，FINAL改成START



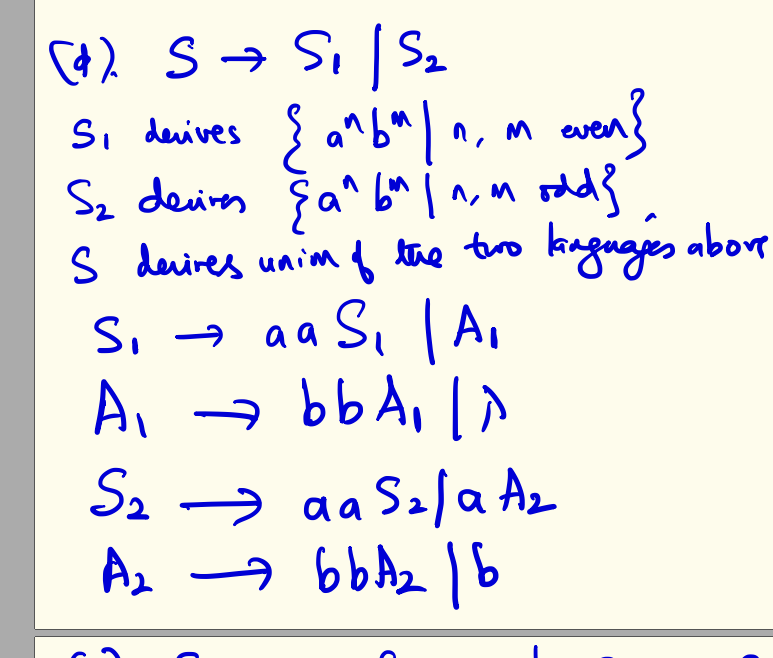
REGULAR EXPRESSION

以什么end aS|bS|xx aS|bS这个组合就是前面随便写

包含某一String 

前面随便写，后面随便写，但要确定通过另一字符A来保证转了aba这一步





n+M是even，要么NM都是even,要么NM都是ODD，分成两种情况

都是even就aaS1A1,bbA1入，都是Odd就aaS2|aA2，A2就bbA2|b

e

S->aS|aaaA

A->bbA|入

f S\_>aA|bA

A->aB|bB|入

B->aS|bS

g S->bS|aA|入

A->bS

h S->bS|aaA

A->aA|bB

B->bB|aaC

C->aC|bC|入

I bS|aaA

A ->aA|bB

B->bB|aac

C->aC|bD|

D->bD|入

j S->bS|aA

A->bA|aD

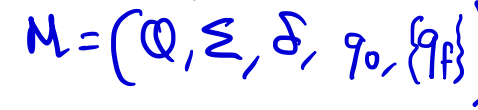
D->bS|入

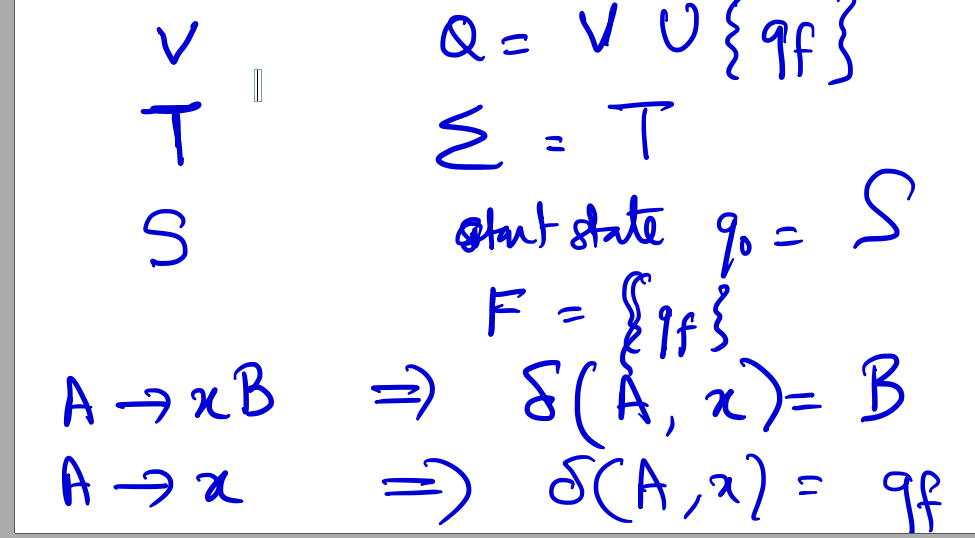
S->xxS,考虑需求成立下可以接受的循环

S->xxA,满足第一个要求

1. xxB,考虑需求成立下可以接受的循环

可以把任意regular grammar转换成一个NFA M，让L(G)=L(M)

G=



Q就是V所有大写字母带个final

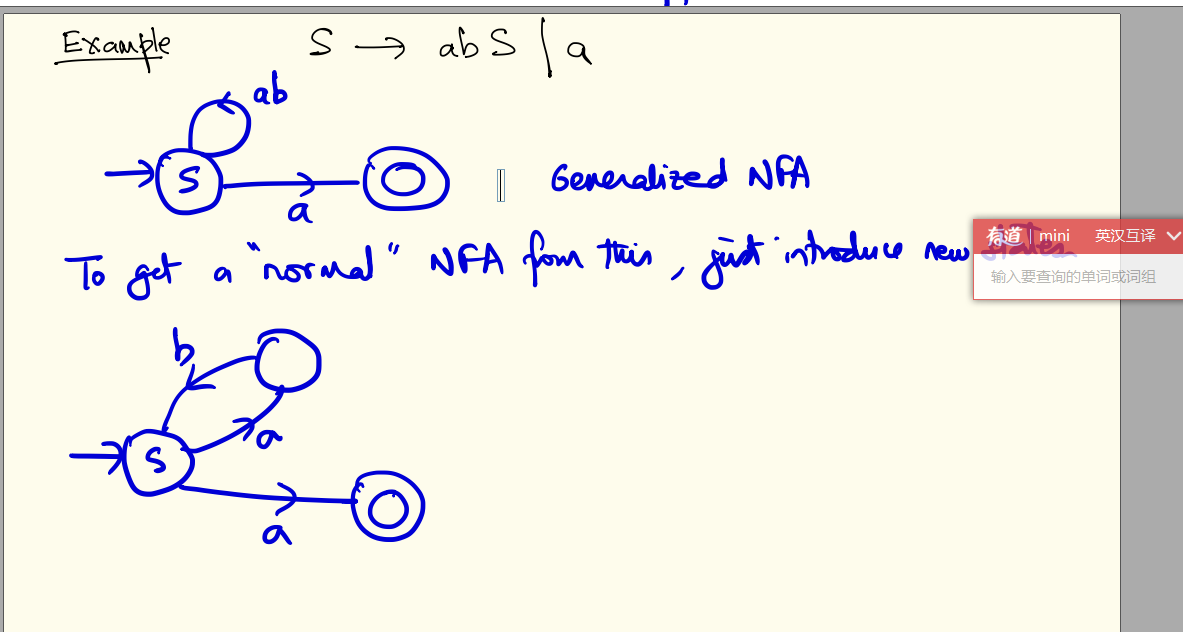
T与Input alphabet相等

S 就是startstate q0

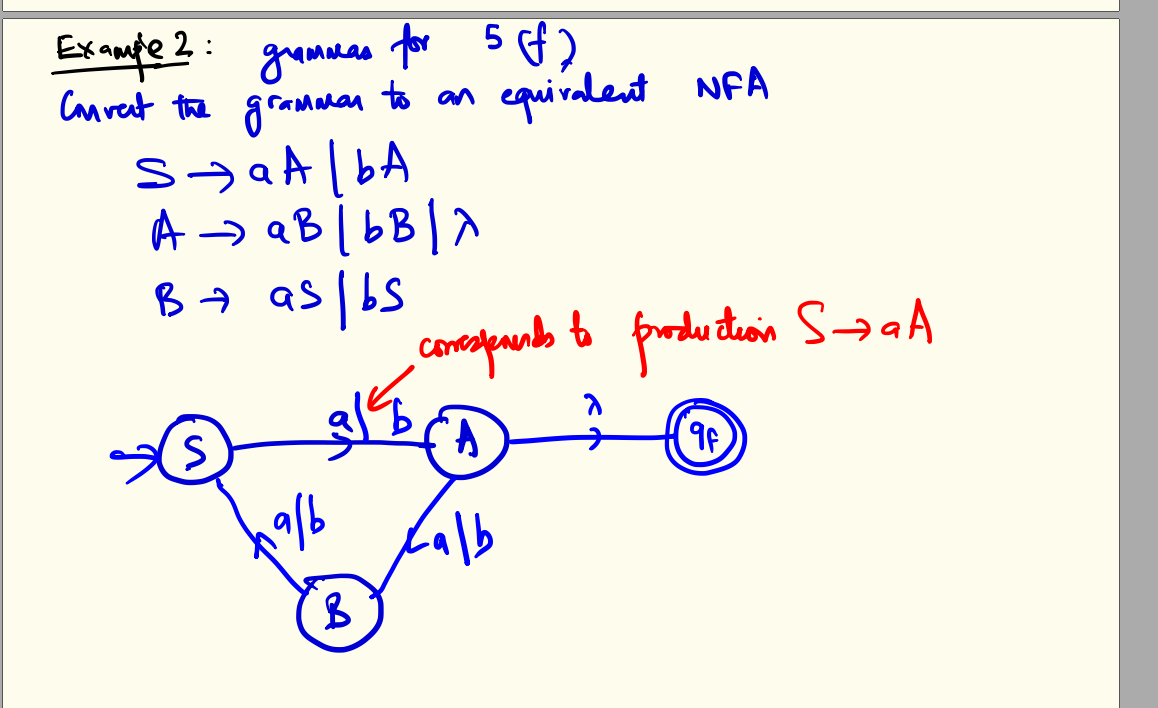
F=qf

A->xB描述的是一个中间路径，从A到Binput x得到B

A->x 描述的是直接到了final

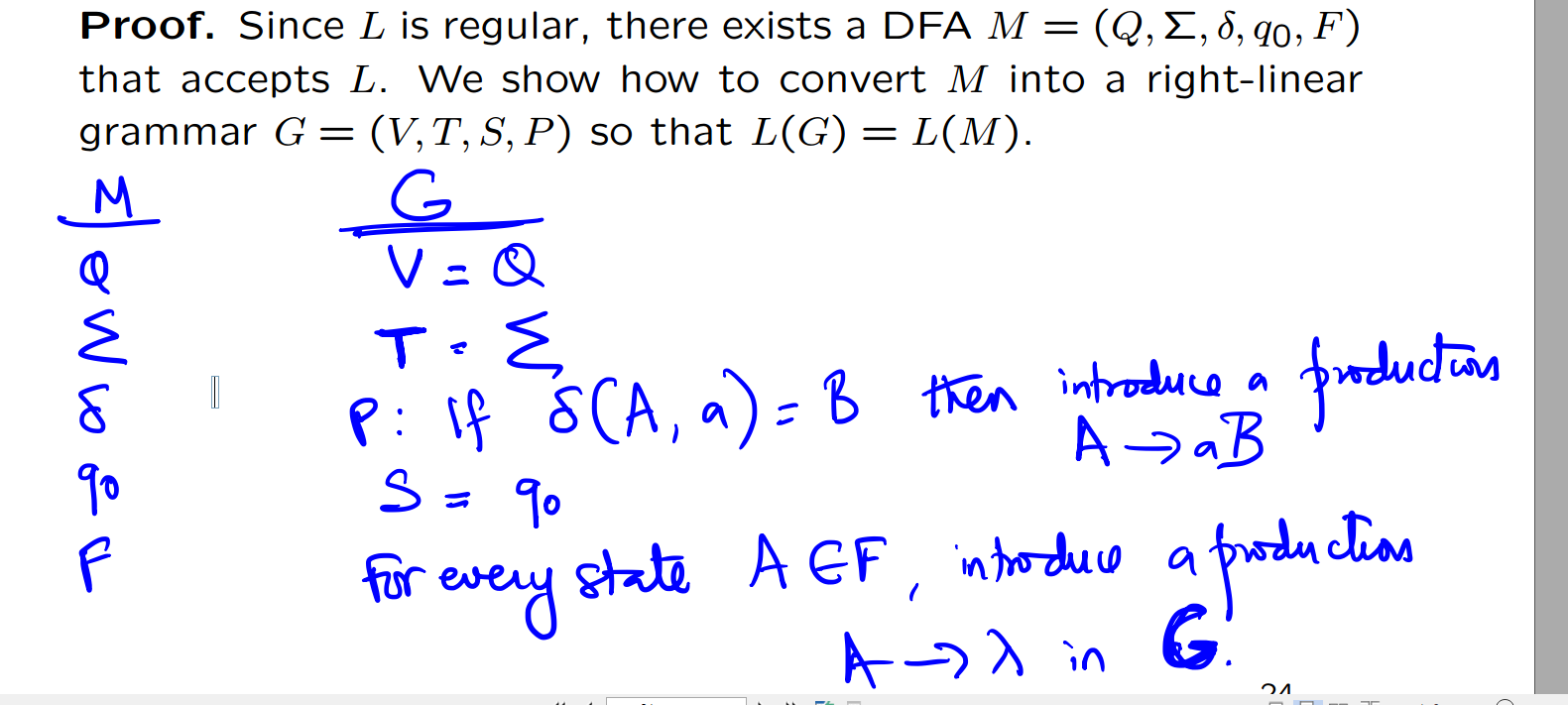


S->abS循环，S->abA,前进，S->a，到final

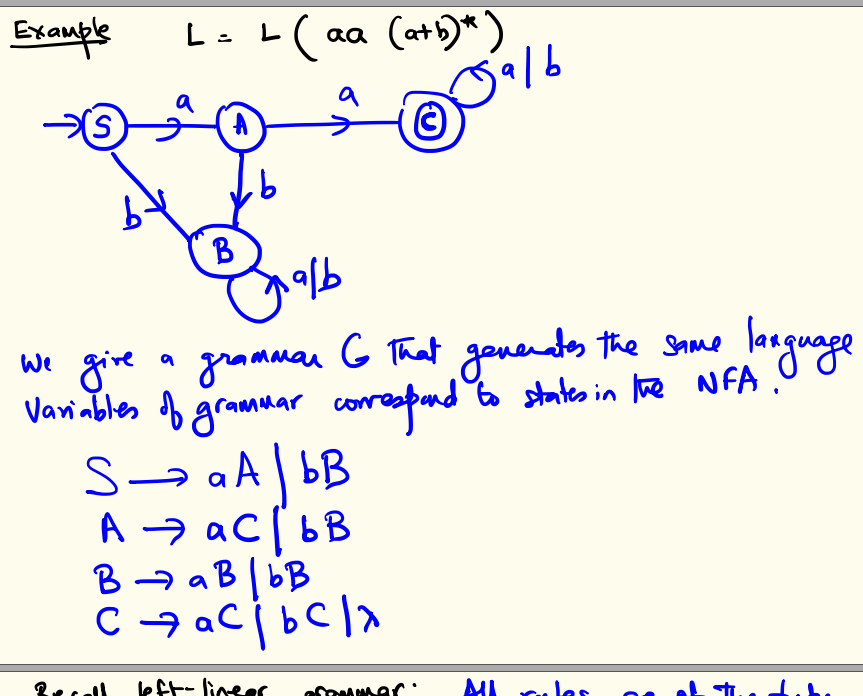


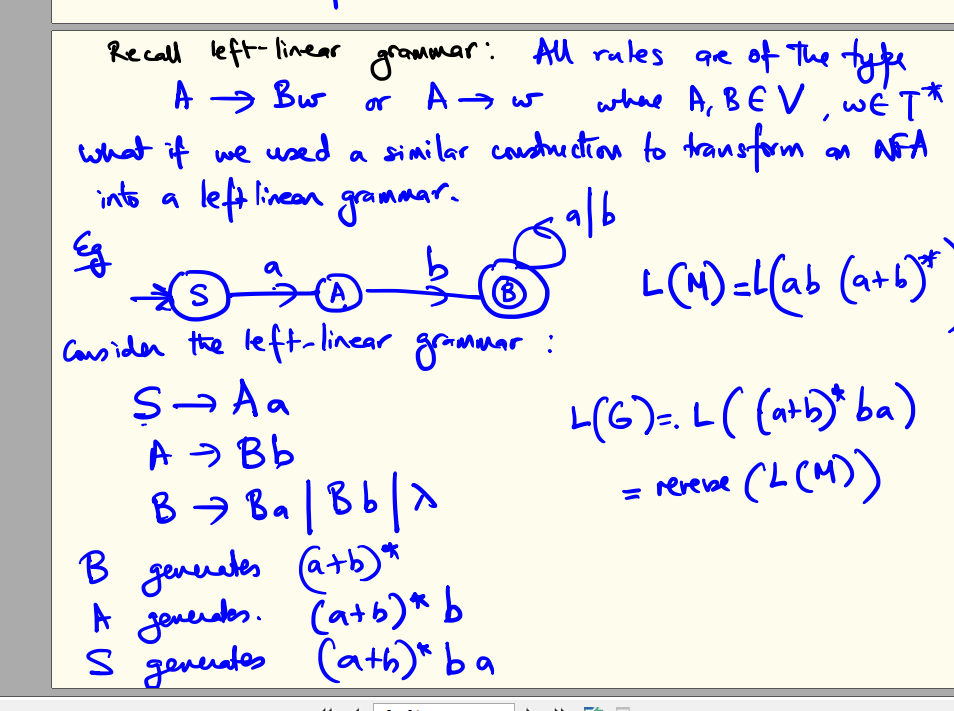
对于每一个Regular language L，都能找到对应right linear grammar G

因为L一定会有对应DFA，



F那一步，对任意属于F集合里的final之1，加上一个A->入

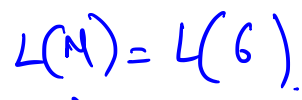




left linear是他的reverse

想把right linear转换成left linear

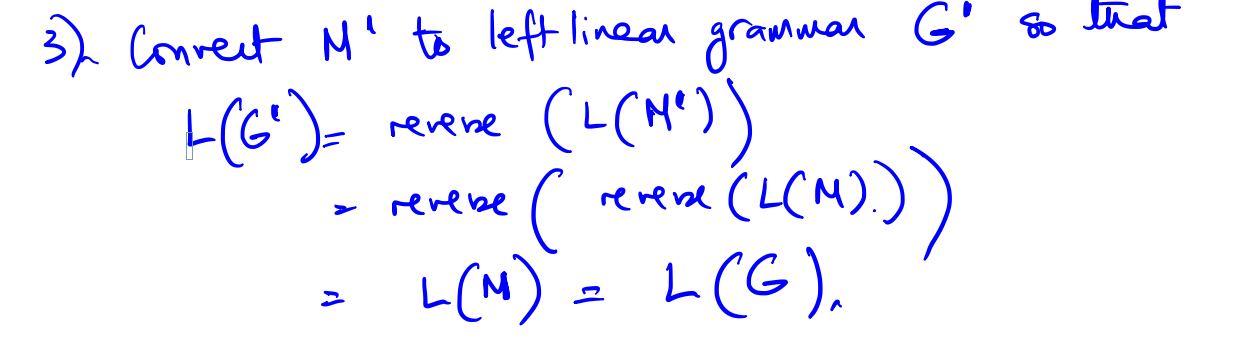
1.right-linearG找到对应的NFA M



2.把M转成M’



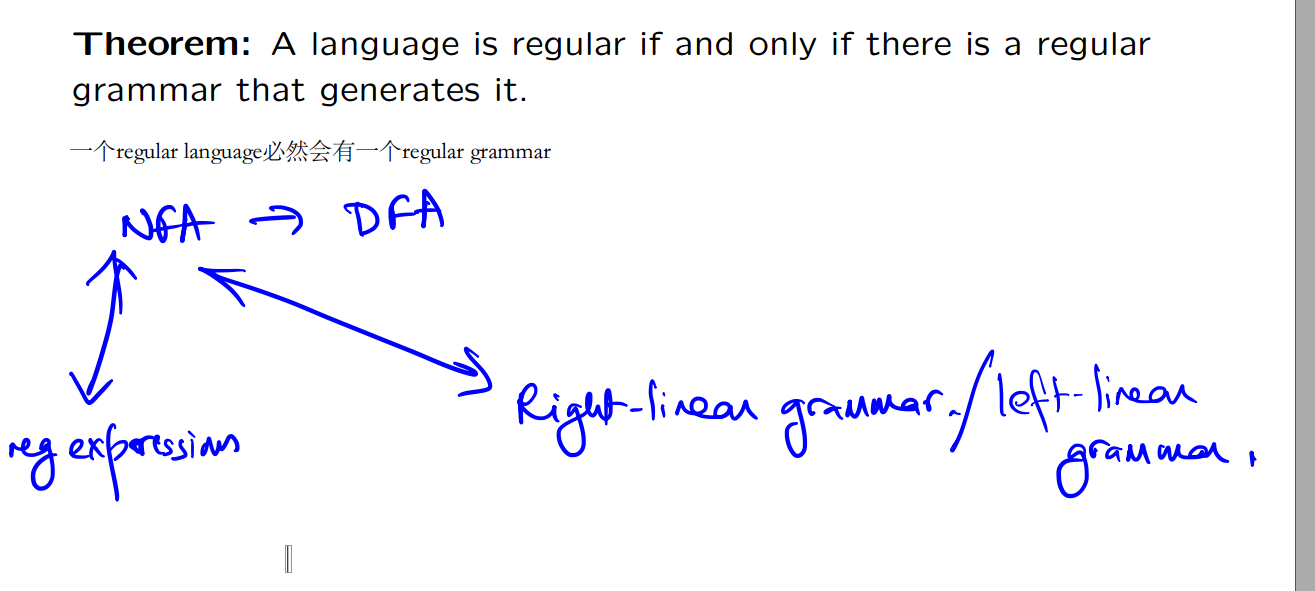
3.M’转成left linear



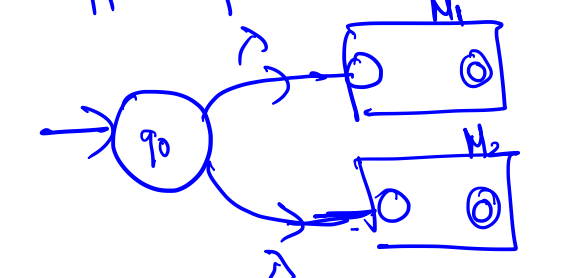
这个G’实际上是reverse L M’

reversereverse L(m)

就是L（m）就是L（g）

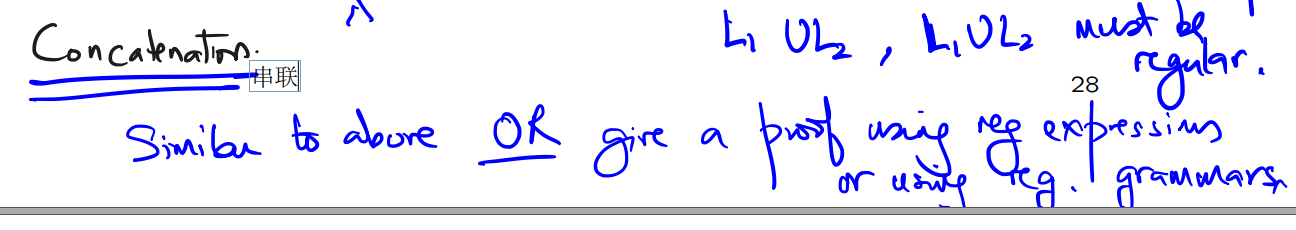


如果L1 L2 regular



L1UL2也regular，因为可以构造单开头单final的NFA，然后我们联合起来构造出新NFA，证明他是regular的

Concatenation串联

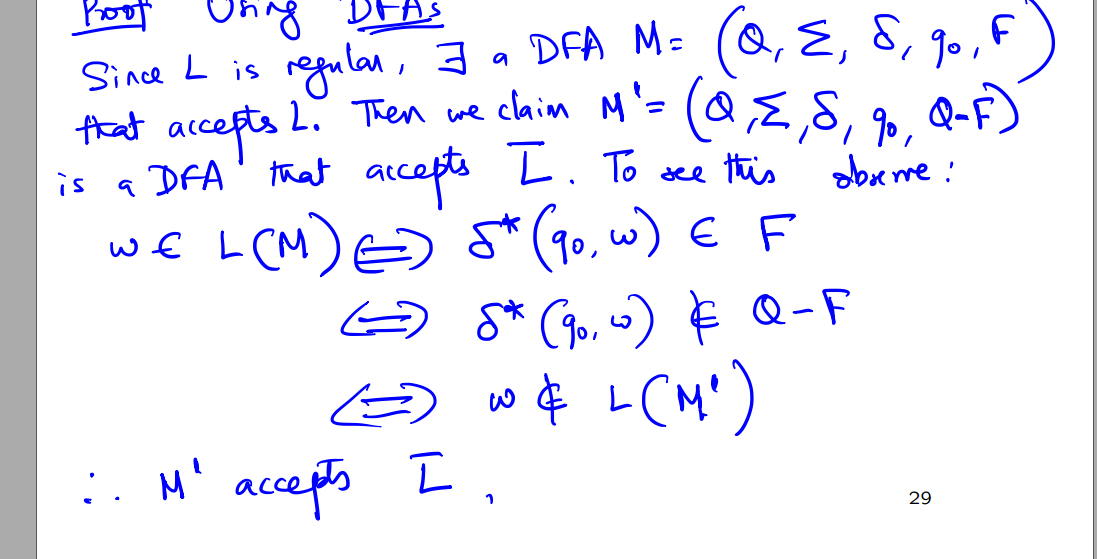


同上，不过是串起来了

Complement

如果L regular,L’也regular

用DFA

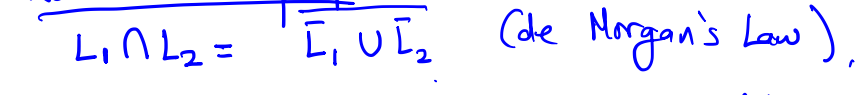


Q 与别的啥都相同，但是原来的F改为正常，其他的正常改为FINAL，不改变方向

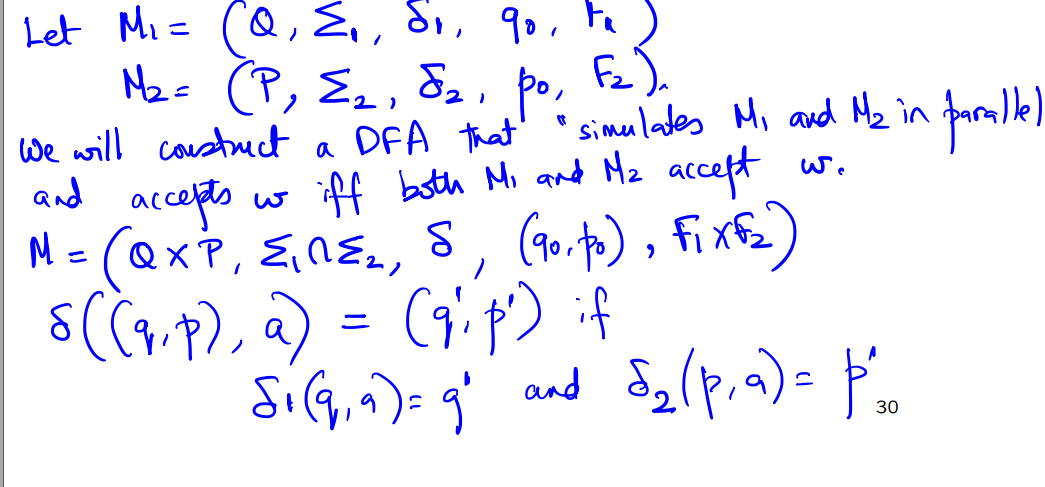
意味着这个W原来经过Q0都属于F，现在都不属于F

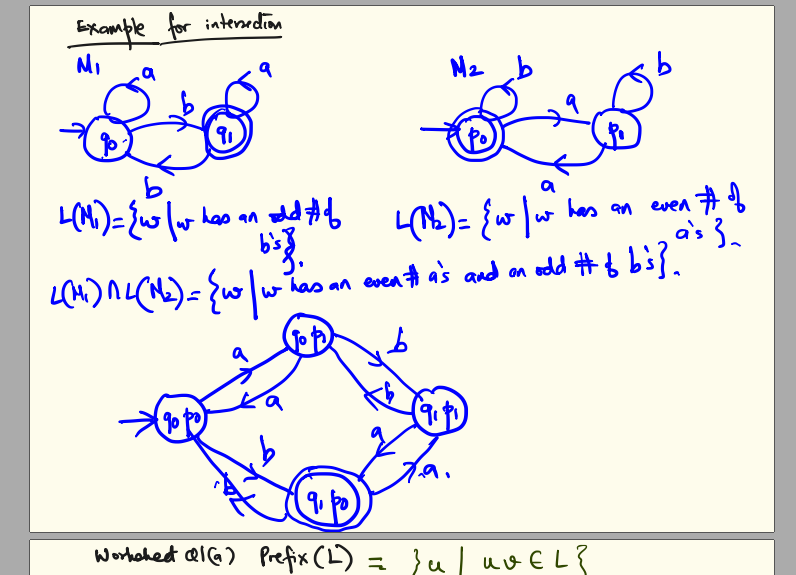
INtersection,交集

第一种方法达默根法则，

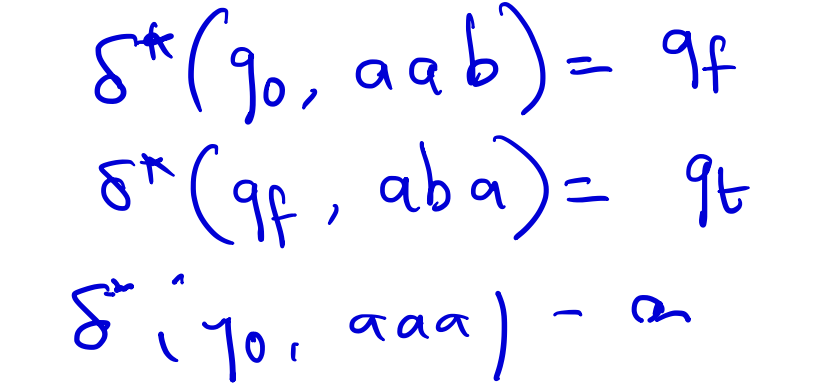


第二种方法





DFA可以AB指向同一个，A|B，不能伸出两个AB，必须有AB



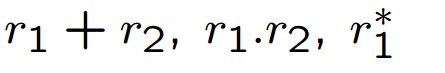
Language 只有有且只有DFA时是regularde

NFA中extended是一组STATE

对于每个NFA有对应DFA

如果是regular language，必能构造出有一个final state的NFA，并且与start不同

Regular expression r,如果r是regular expression

都是

对于每个Regular expression一定有对应NFA

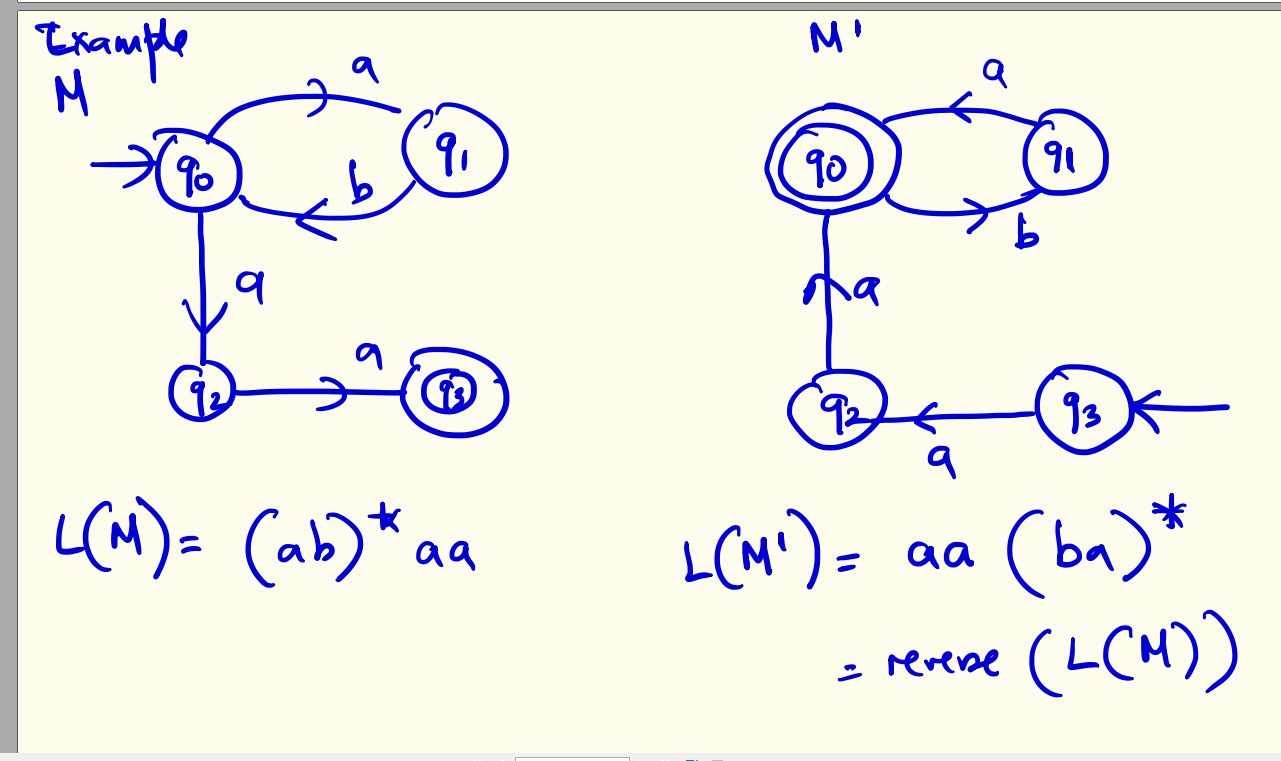
+就左边两个入，右边两个入

✖就直接连起来，中间加入

星在盒子外面·加上两个state，用入链接，两个state入循环

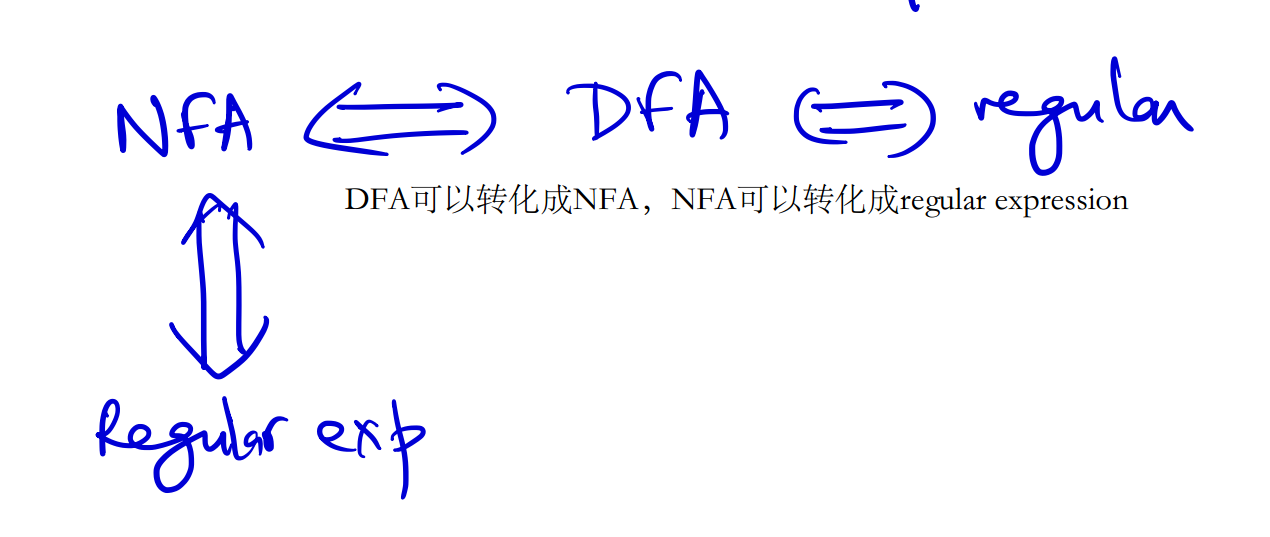
对于每个regular language，if and only if一定有对应regular expression

对于Regular expression 的 reverse



箭头相反，初始与FINAL互换

补给的话就要不改箭头，FINAL与中间兑换



Right linear grammar G 一定能转化成 对应NFA M

对于任意regular grammar，一定有对应RIGHT LINEAR

把right linear转换成LEFT LINEAR

创造一个RIGHT LINEAR对应的M 倒转，在用LEFT表达

对于任意regular language，if and only if regular grammar



