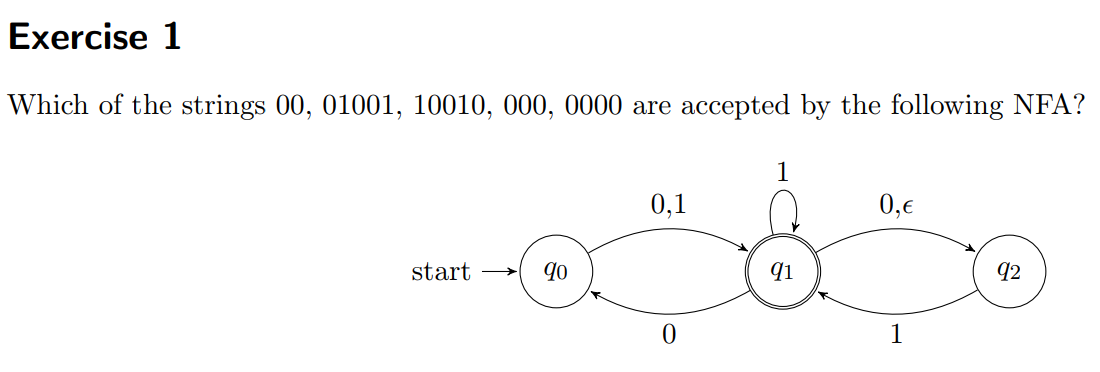
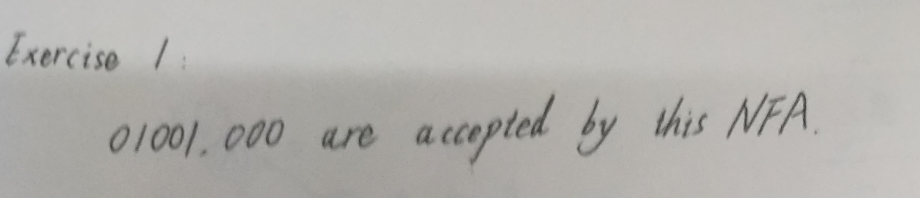
# Lab3：NFAs

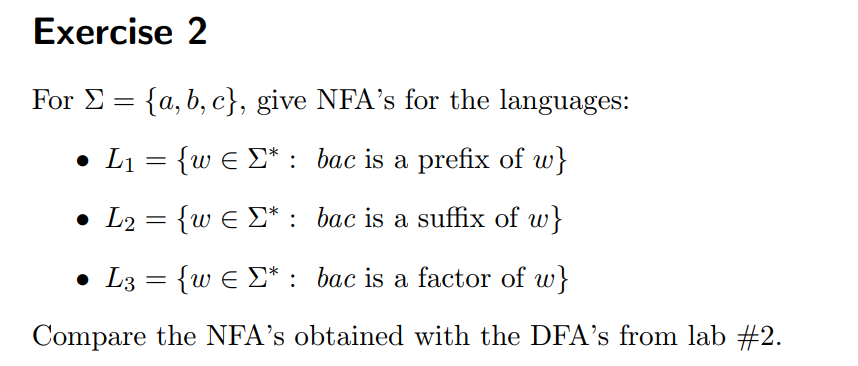
**NFA的画图：要求的字符串画一个确定路径就够了，在首/尾进行处理即可，中间不用加什么分叉。应该是把节点路径是死循环的部分都删除省略了，剩下的 节点都是有可能满足条件的。**

**时刻注意：空串也是接收态Exercise3**



题意：判断字符串00，01001，000，0000是否能被该NFA接收





obtained：获得

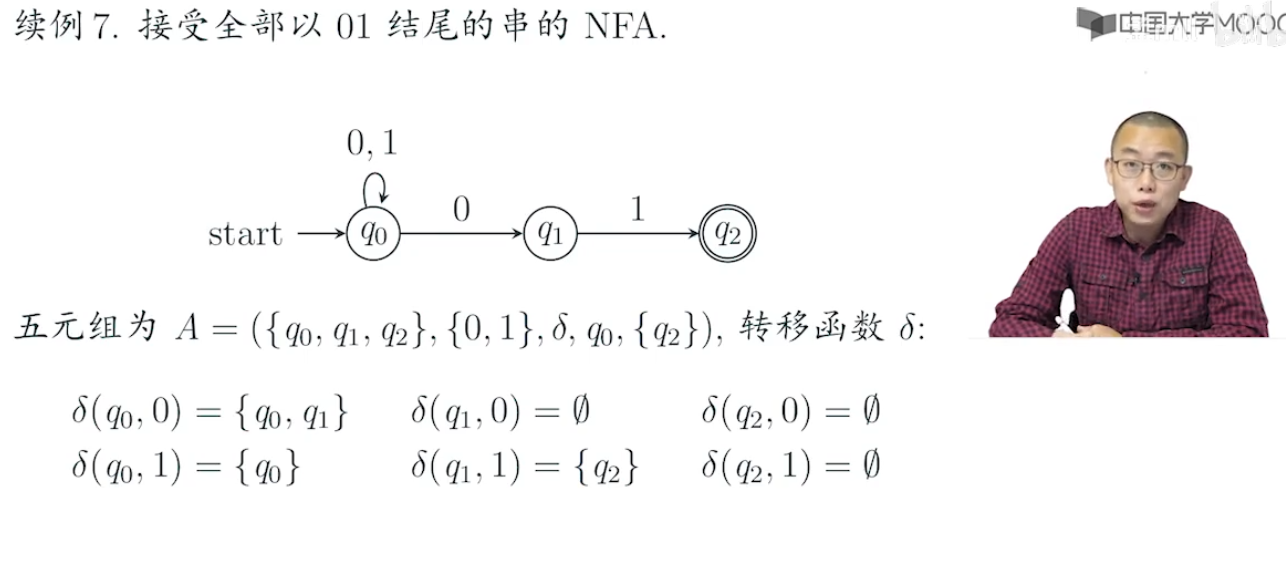
题意：写出下面语言的NFA：

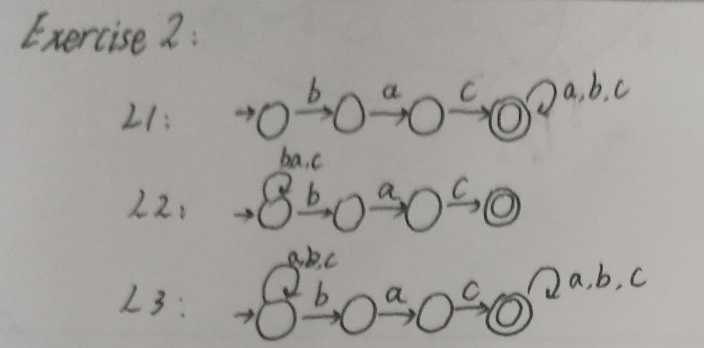
L1：bac是字符串w的前缀

L2：bac是字符串w的后缀

L3：bac是字符串w的一部分

比较获得的NFA和Lab2中的DFA

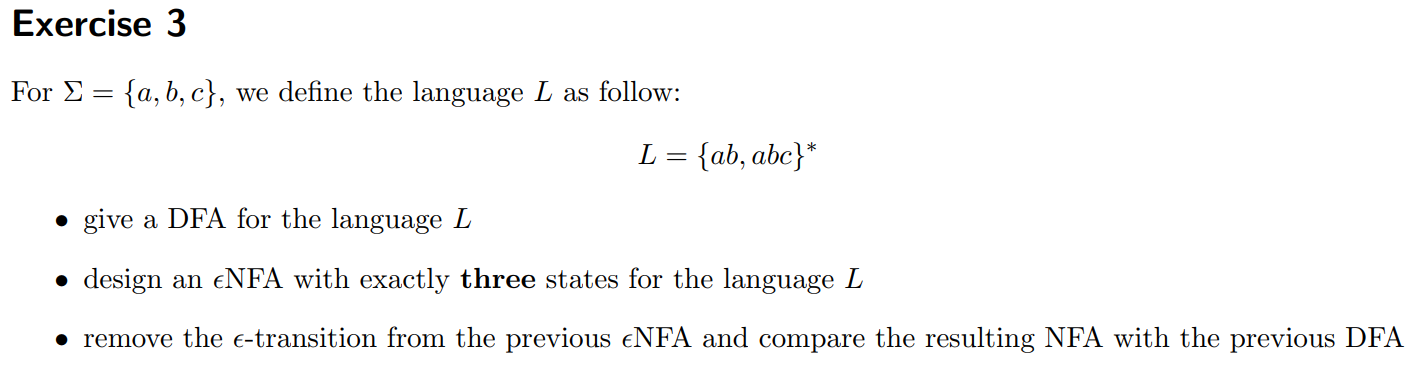
L2：



L1：要求画前缀，所以前缀和中间不用动，只需要在后续加路径即可

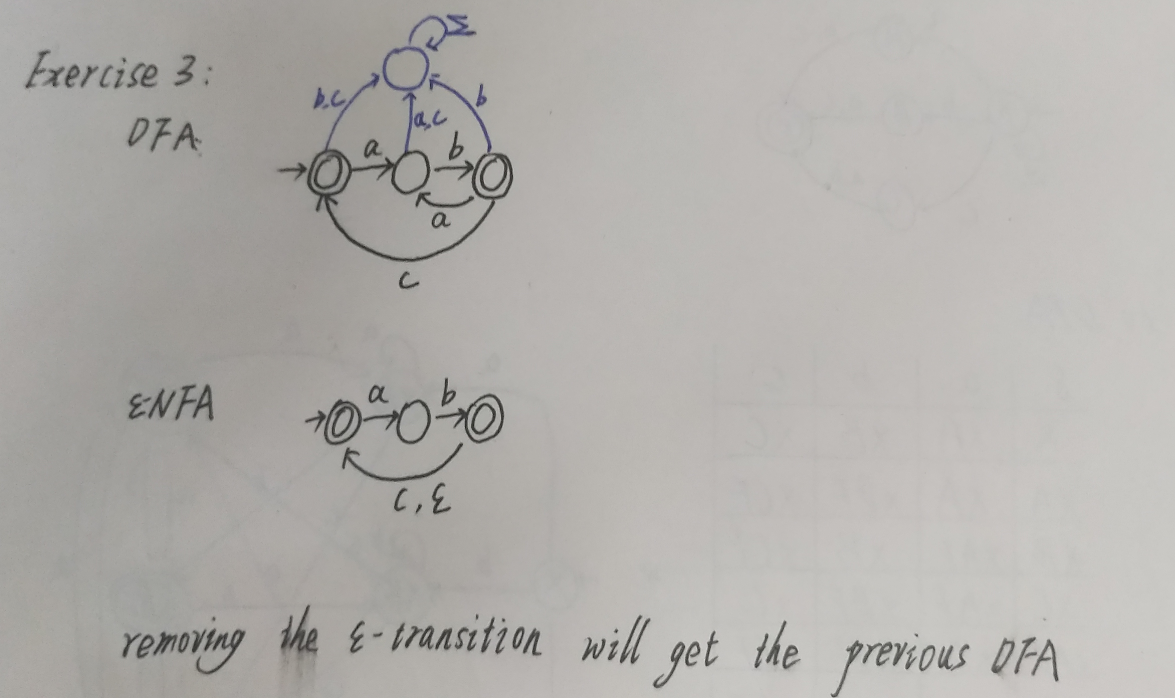
L2：要求画后缀，所以后缀和中间不用动，只需要在前面加路径即可

L3：要求画中间，所以只有中间不用动，前面和后面都要加路径

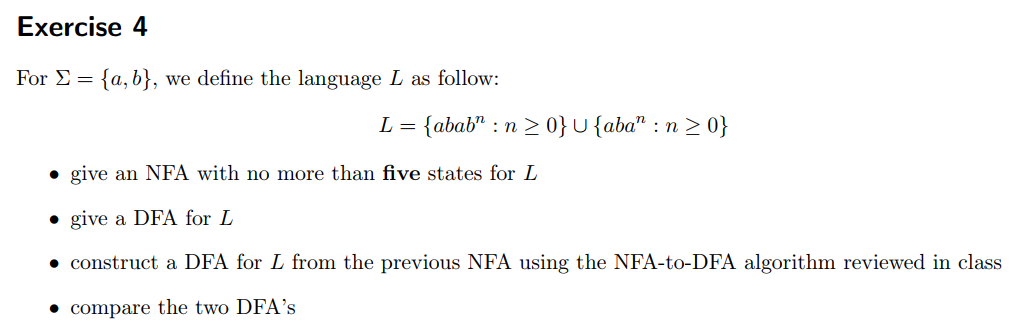


题意：语言L是由字符串ab，abc组成的克林闭包

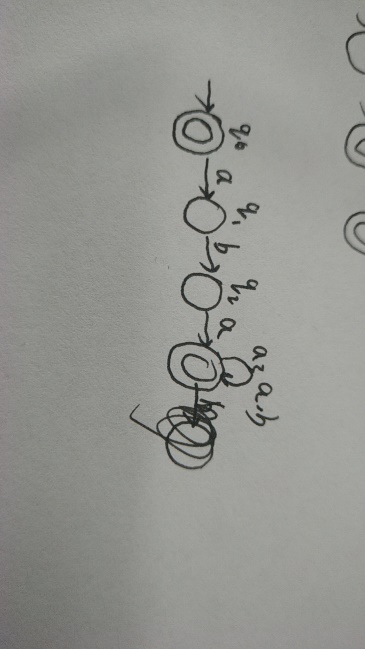
1. 画出语言L的DFA
2. 设计一个恰好由三个状态的空转移NFA
3. 删除2中的空转移。使得到的NFA与前面得到的DFA相比较



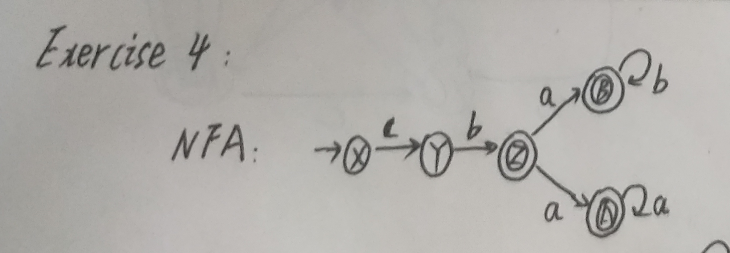
2中的空转移加在了接收态之间的转换上了

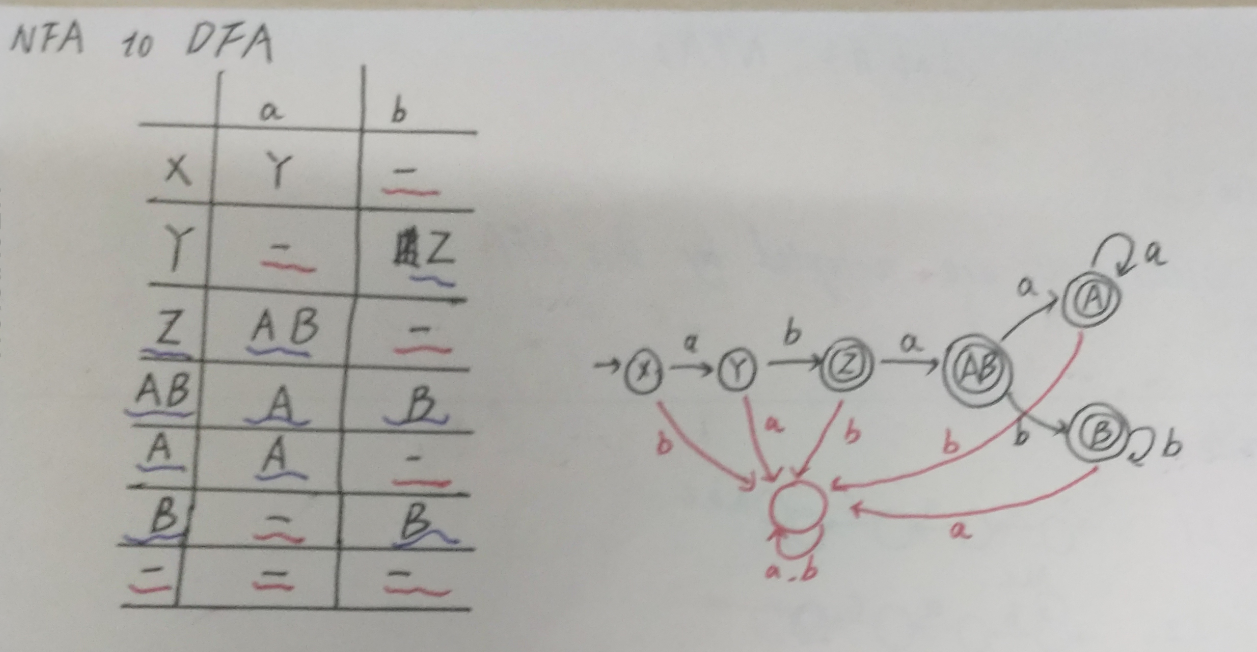


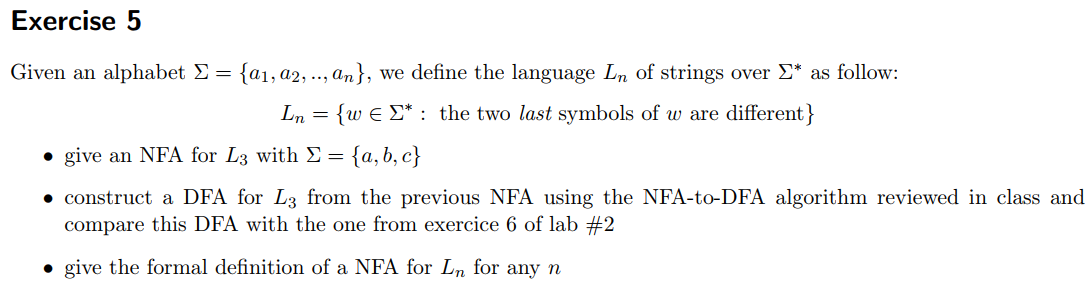
题意：

1. 给出L的NFA，要求NFA不超过5种状态
2. 给出L的DFA
3. 从之前得到的NFA中创建一个DFA，用到NFA-to-DFA的算法
4. 比较两个DFA
5. 不对：；

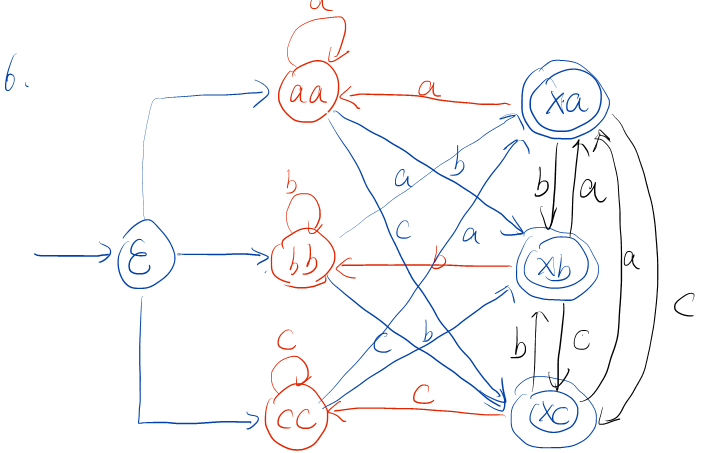
abababa就不对，空串也不能接收，不是闭包

正确：

HFA-to-DFA应该必考的

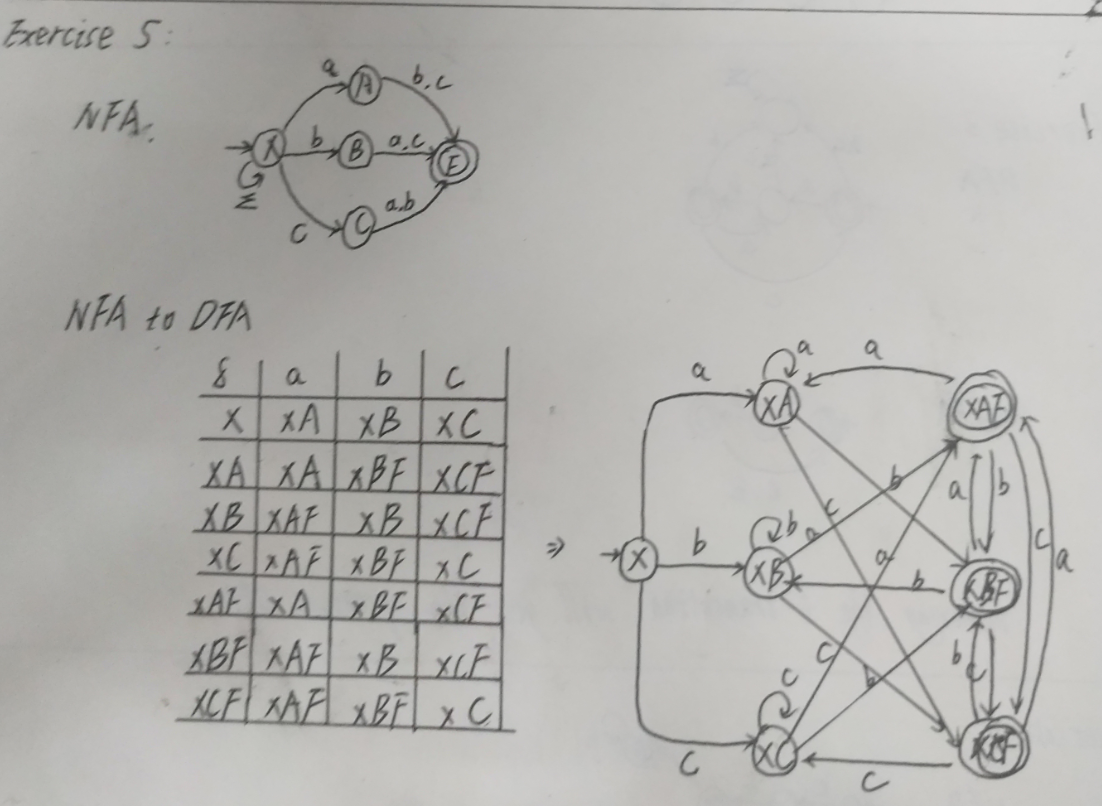


**有点不好理解了，多看两遍，与lab2的Exercise6**

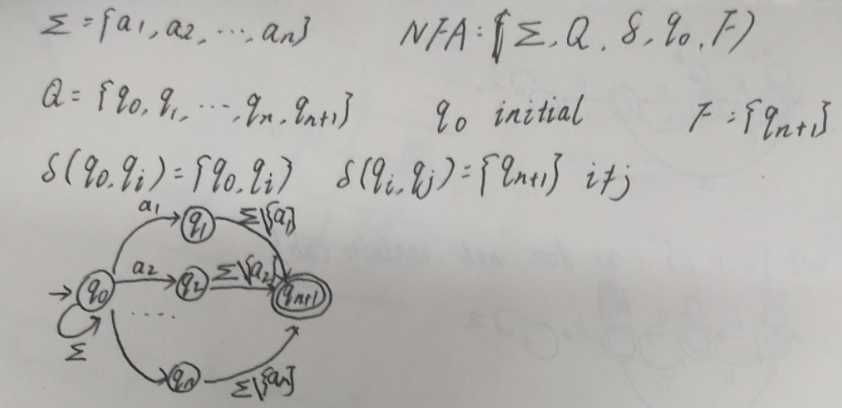


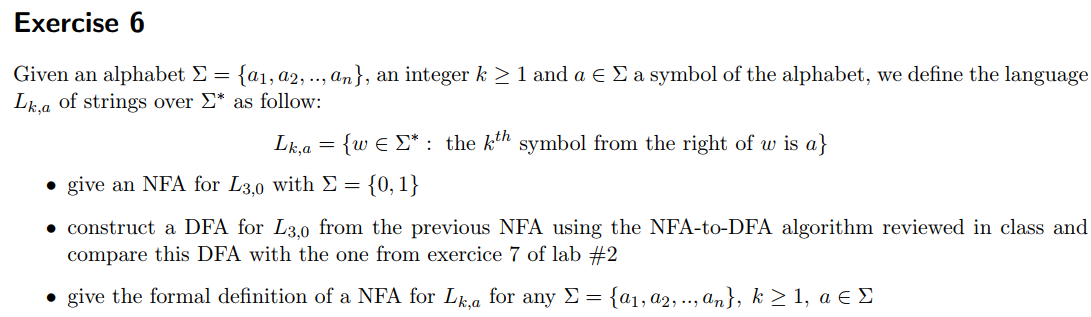
题意：题意：语言Ln，字符串w的最后两个字符是不同的

1. 写出L3的NFA
2. 从L3的NFA转换成DFA，同时比较。。。
3. 给Ln一个正式的NFA定义。the formal definition:正式定义，就是让写一个完整的五元组

注：NFA-to-DFA要写根据DFA的节点名称写，而不是根据NFA的节点名称写，因为NFA中节点个数要比DFA中节点个数要小，会写不全  


注意五元组的写法：（考前写几遍）

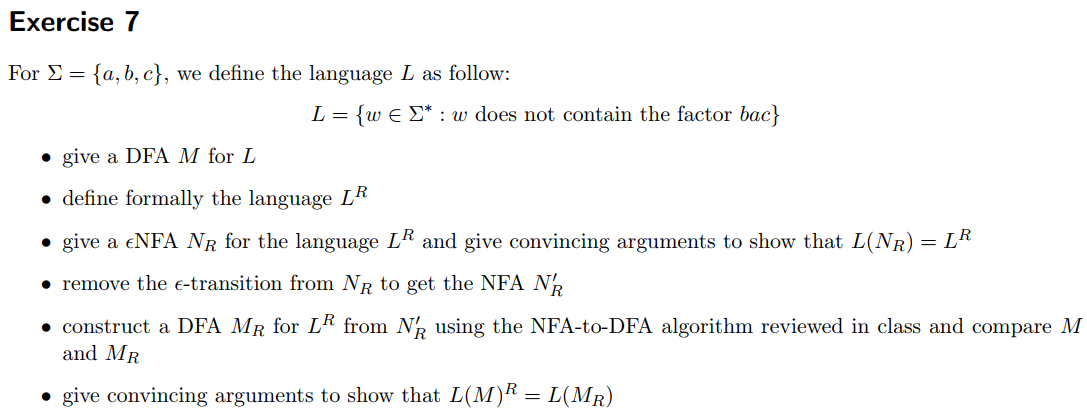




题意：Lk,a表示字符串w右边数第k个符号是a。

1. 给出L3,0的NFA
2. 将NFA-to-DFA
3. 给出Lk,a的NFA正式定义（五元组）

把五元组的书写补上。

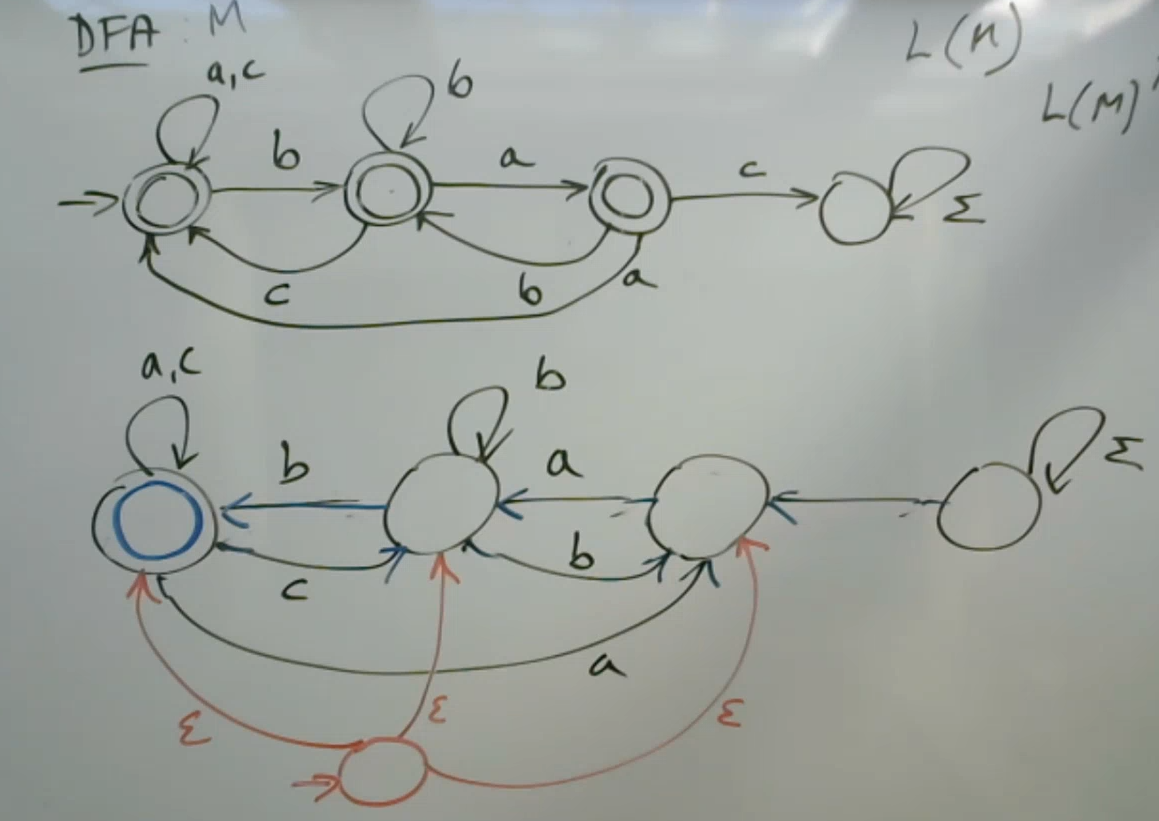


题意：语言L中不包含bac

1. 给出L的DFA
2. 正式的定义语言LR（L的逆置）
3. 给出语言LR带空转移的NFA
4. 删除空转移得到的NFA
5. 由第四问中的NFA构建一个DFA
6. 证明结论

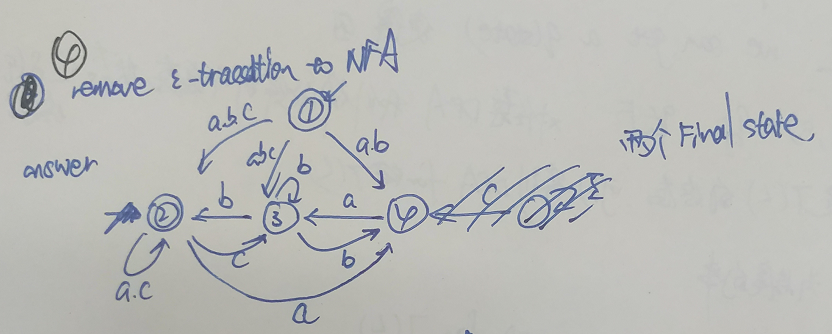
L DFA—>LR ∈NFA的固定操作：

1. 把所有箭头反着画，上面的字母保留
2. L DFA的起始状态（即start处的箭头所指的节点）变成LR ∈NFA的接收状态
3. L DFA的接收状态变成LR ∈NFA的起始状态
4. 但由于起始状态有3个，所以用到空转移

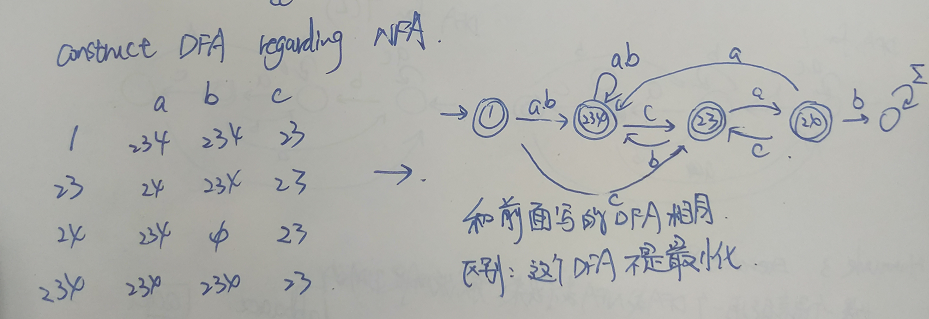
这个图少画了个接收态（起始位置处）

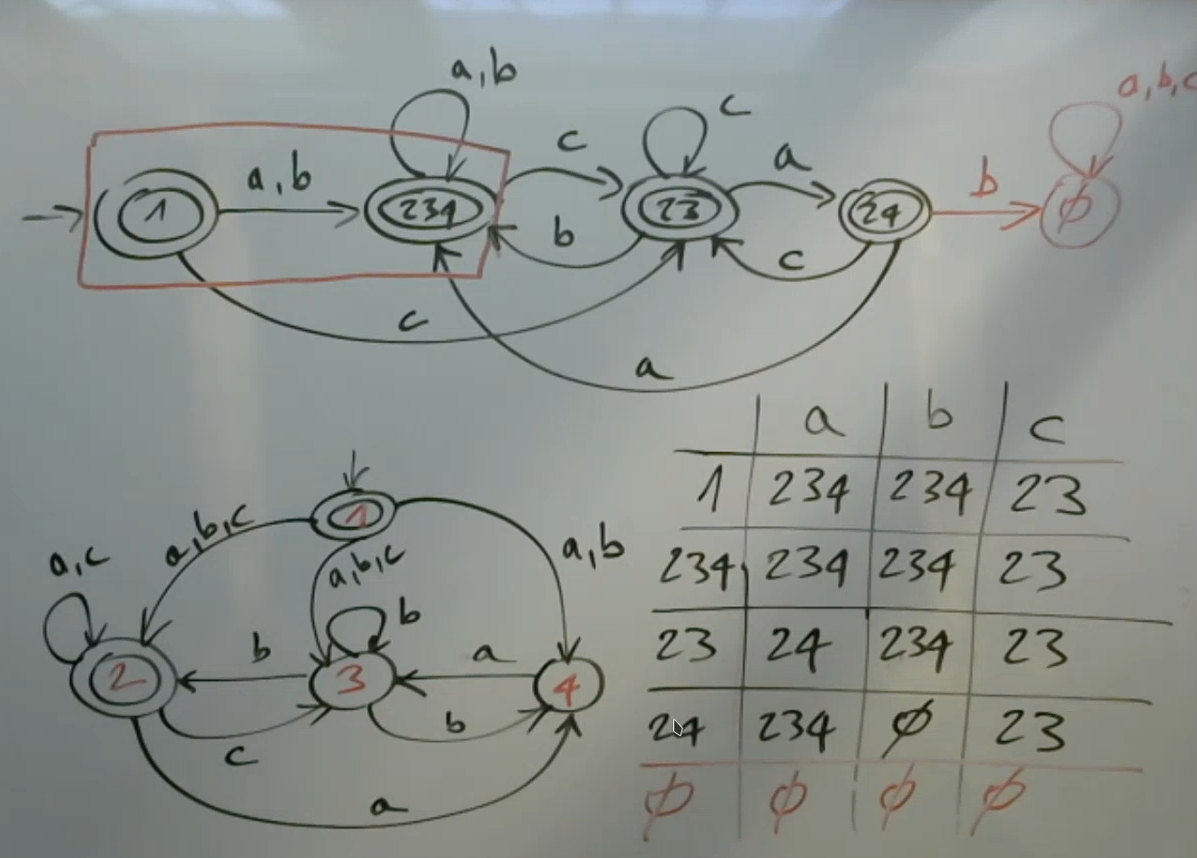
删除空转移的操作：

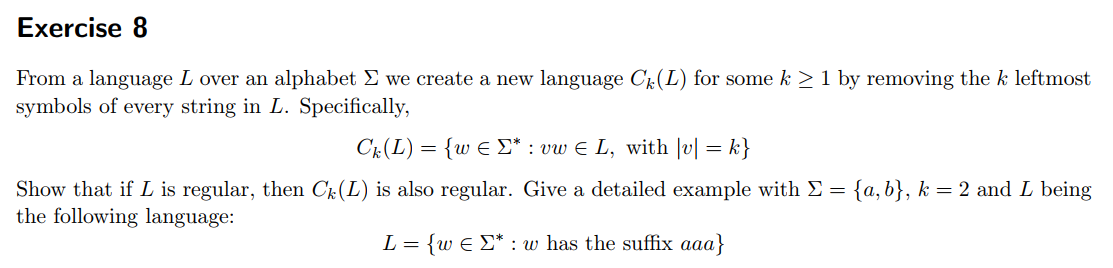
找到空转移指向的节点，看看由起始指针经过那个字母能到达那里，那把那个字母填到那个路径上



DFA-to-NFA







题意：