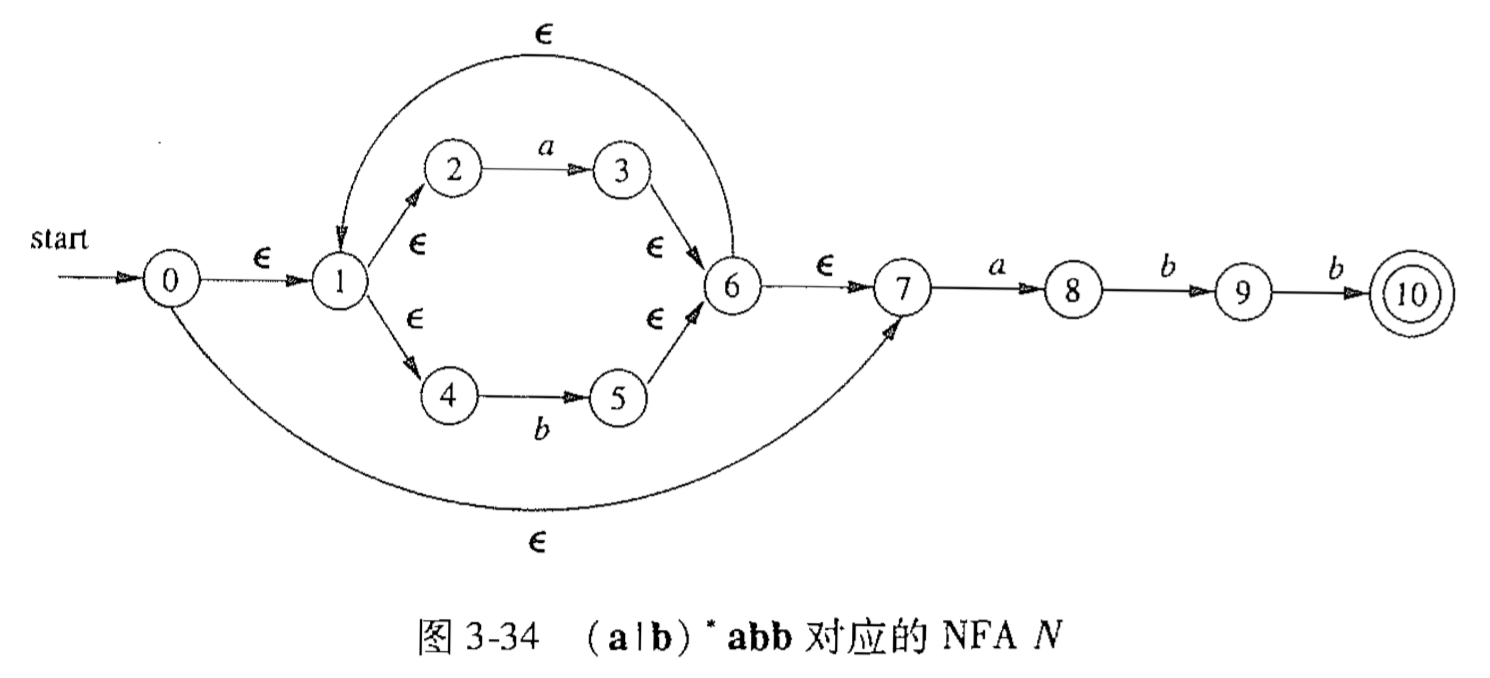
第三章 词法分析 →∪Σε

3.7.1 从NFA到DFA的转换



aaabb

初始值 ：当前指针指向第一个字符，即a

当前状态集合: ε-closure({0}) = {0, 1, 2, 4, 7} {A}

|  |  |  |
| --- | --- | --- |
| 当前指针 | 当前状态集合S | 操作 |
| aaabb | {0, 1, 2, 4, 7} {A} | ε-closure( {3, 8}) |
| aaabb | {1, 2, 3, 4, 6, 7, 8} {B} | ε-closure( {3, 8}) |
| aaabb | {1, 2, 3, 4, 6, 7, 8} {B} | ε-closure( move(S, ‘a’)) |
| aaabb | {1, 2, 3, 4, 6, 7, 8} {B} | ε-closure( {5, 9}) |
| aaabb | {1, 2, 4, 5, 6, 7, 9} {C} | ε-closure( {5, 10} ) |
| aaabb | {1, 2, 4, 5, 6, 7, 10} {D} | 接受, 返回yes |

不确定性

DFA: 确定的

子集构造法

输入：一个NFA N

输出：一个接受同样语言的DFA D

基本思想 ： 让构造得到的DFA的每个状态对应于NFA的一个状态集合。

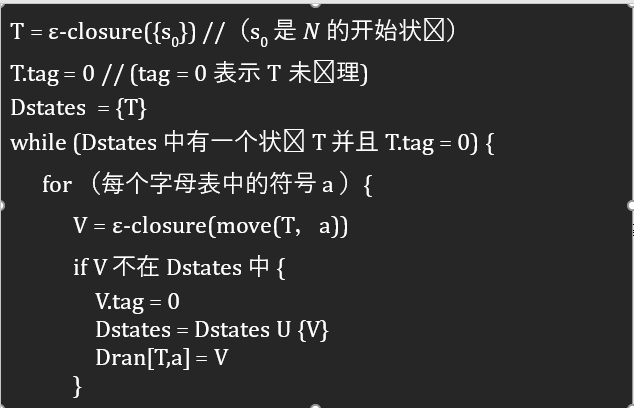
例如: 读入输入串a1 a2 … an后, 根据NFA我们可以到达一个状态集合, 在构造的DFA中，我们用一个状态来表示这个NFA的状态集合。

理论上来讲, DFA中的状态数有可能是NFA状态数的指数;

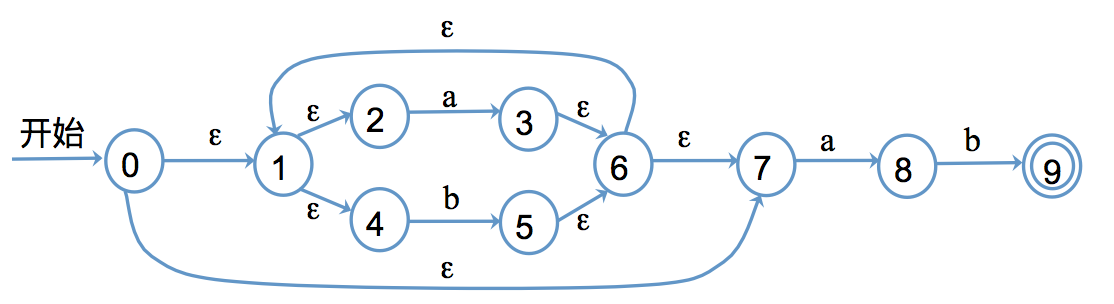
实践中, 对于一个真语言, 它的NFA和DFA的状态数量大致是相同的.

* move(S, c)
* ε-closure(S)

子集构造算法:



(a|b)\*ab通过MYT算法得到的NFA



|  |  |  |  |
| --- | --- | --- | --- |
| NFA状态 | DFA状态 | a | b |
| {0, 1, 2, 4, 7} | A | B | C |
| {1, 2, 3, 4, 6, 7, 8} | B | B | D |
| {1, 2, 4, 5, 6, 7} | C | B | C |
| {1, 2, 4, 5, 6, 7, 9} | D | B | C |

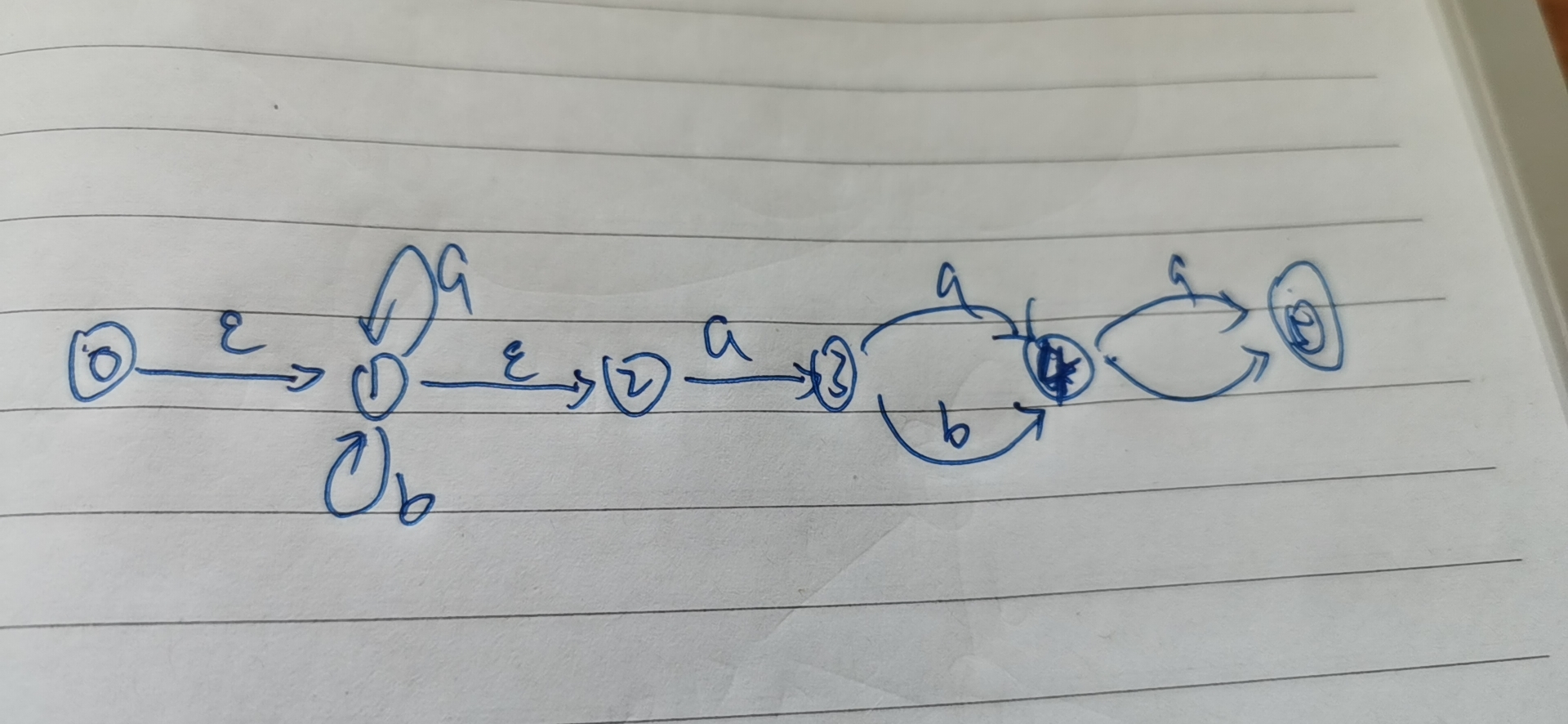
练习:

(1) 为正则表达式(a|b)\*a(a|b)(a|b)构造NFA。

(2) 并将NFA转换为DFA。

注: 使用MYT(自底向上)和自顶向下法分别生成NFA, 然后再分别转换为DFA, 并比较两个DFA是否一样.

下法得到NFA:



将上图NFA通过子集构造算法生成DFA

|  |  |  |  |
| --- | --- | --- | --- |
| NFA状态集合 | DFA状态 | a | b |
| {0, 1, 2} | A | B | C |
| {1,2, 3} | B | D | E |
| {1, 2} | C | B | C |
| {1,2,3,4} | D | F | G |
| {1,2,4} | E | H | I |
| {1,2,3,4,5} | F | F | G |
| {1,2,4,5} | G | H | I |
| {1,2,3,5} | H | D | E |
| {1,2,5} | I | B | C |

