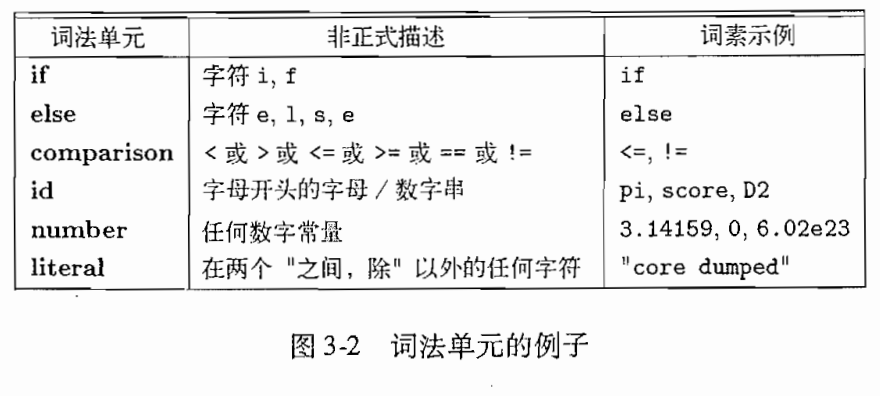
编译原理复习 #2 Lexical Analysis

**词法单元、模式和词素\***

* **词法单元（token）**由一个**词法单元名**和一个**可选的属性值**组成。词法单元名是一个表示某种词法单位的抽象符号，比如一个特定的关键字，或者代表一个标识符的输入字符序列。词法单元名字是由语法分析器处理的输入符号，**属性值可以是一个指针指向一系列的属性**
* **模式（pattern）**描述了**一个词法单元的词素可能具有的形式**。当词法单元是一个关键字时，它的模式就是组成这个关键字的字符序列。对于标识符和其他词法单元，模式是一个更加复杂的结构，它可以和很多符号串匹配
* **词素（Lexemes）**是**源程序中的一个字符序列**，它和某个词法单元的模式匹配，并被词法分析器识别为该词法单元的一个**实例**

**这里是一个示例：**

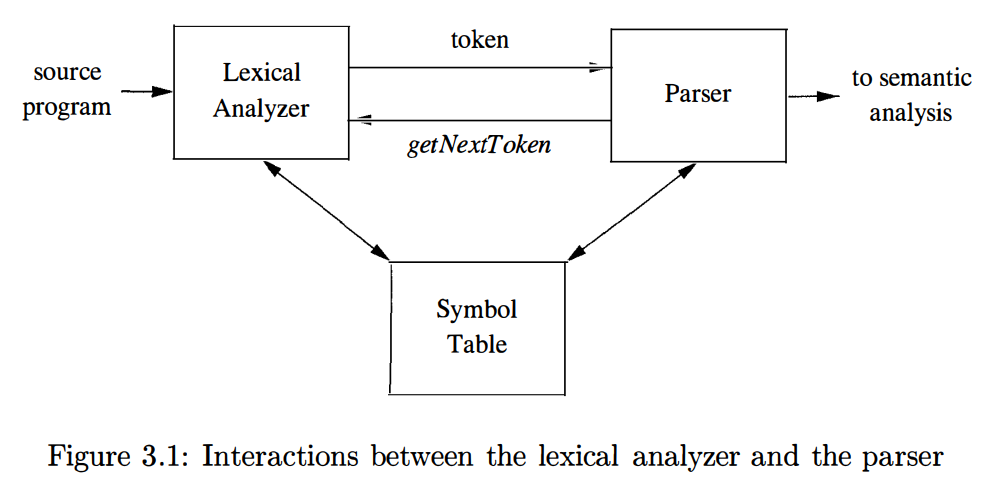


**常见词法单元分类：**

* 每个关键字有—个词法单元。关键字的模式就是该关键字本身
* 表示运算符的词法单元。它可以表示单个运算符，也可以表示一类运算符
* 一个表示所有标识符的词法单元
* 一个或多个表示常量的词法单元，比如数字和字面值字符串
* 每一个标点符号有一个词法单元，比如左右括号、逗号和分号

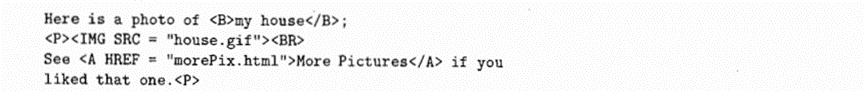
**词法分析器的作用**

**词法分析是编译的第一阶段**。词法分析器的主要任务是**读入**源程序的输入字符、将它们**组成词素（一个一个字符读入然后拼到一起成为一个词即词素）**，**生成并输出一个词法单元序列**，每个词法单元对应于一个词素



**例题**

**像 HTML 或 XML 之类的标记语言不同于传统的程序设计语言，它们要么包含有很多标点符号（标记），如 HTML，要么使用由用户自定义的标记集合，如 XML。而且标记还可以带有参数。请指出如何把如下的 HTML 文档划分成适当的词素序列。哪些词素应该具有相关联的词法值？应该具有什么样的值？**



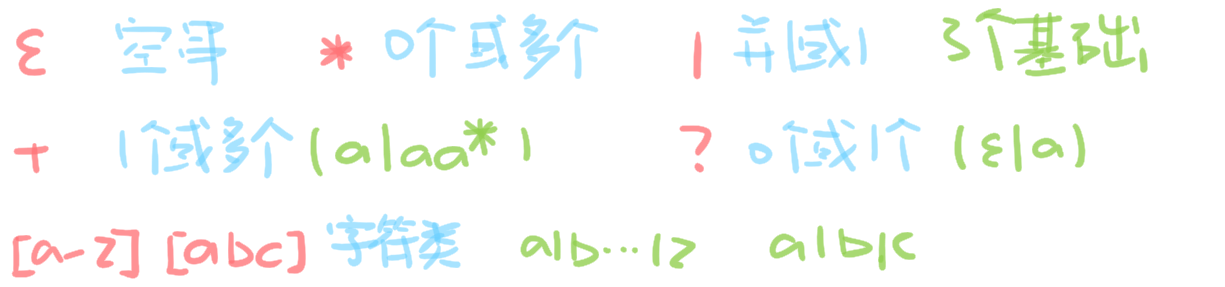
**解答：**

|  |  |  |
| --- | --- | --- |
| **词法单元** | **非正式描述** | **示例** |
| text | 标签间的文本 | Here is a photo of，my house，See，More Pictures，if you liked that one. |
| start\_tag | 开始标签 | <B>，<P>，<IMG>，<BR>，<A> |
| end\_tag | 结束标签 | </B>，</A> |
| attribute | 标签内的参数名 | SRC， HREF |
| value | 标签内的参数值 | “house.gif”，”morePix.html” |
| assign\_op | 赋值符号 | = |

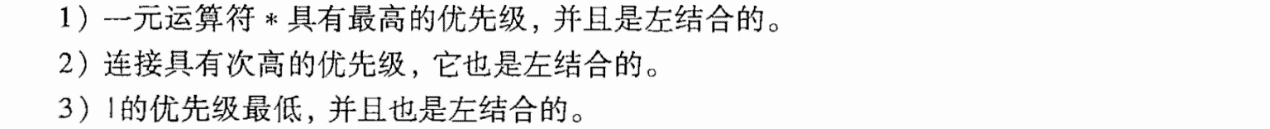
**另外一种可能**

* <string, Here is a photo of>
* <<>
* <tag, B>
* <>>
* <string, my house>
* <</>
* <tag, B>
* <>>
* <;>
* <<>
* <tag, P>
* <>>
* <<>
* <tag, IMG>
* <attr, SRC>
* <=>
* <“>
* <string, house.gif>
* <“>
* <>>
* <<>
* <tag, BR>
* <string, See>
* <<>
* <tag, A>
* <attr, HREF>
* <=>
* <“>
* <string, morePix.html>
* <“>
* <>>
* <string, More Pictures>
* <</>
* <tag, A>
* <>>
* <string, if you liked that one.>
* <<>
* <tag, P>
* <>>

**正则表达式及其扩展\***



**优先级：**



**例题**

**3.3.5 试写出下列语言的正则定义**

* **所有由按词典递增序排列的小写字母组成的串**



* **注释，即 / \* 和 \*/ 之间的串，且串中没有不在双引号（”）中的 \*/**

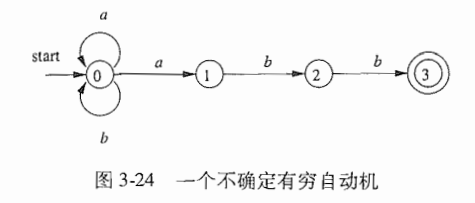
/\\*([^\*”]|\\*[^/]|\”([^”]\*)\”)\*\\*/

* **所有由 a 和 b 组成且不含子串 abb 的串**



**有穷自动机\*（Finite Automaton）**

**不确定的有穷自动机（Nondeterministic Finite Automata, NFA）**对其边上的标号没有任何限制。一个符号标记离开同一状态的多条边，**空串可以作为标号**



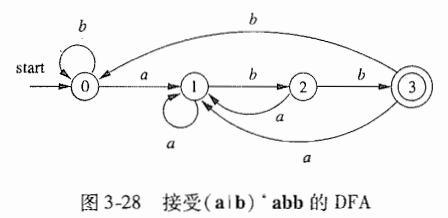
* **要有开始状态：**一个箭头上面带着 start 指向他
* **可接受状态**是双圆圈
* 箭头上是字符，可以使用空串。圆圈内是状态数字

**转换表 （transition table）：**

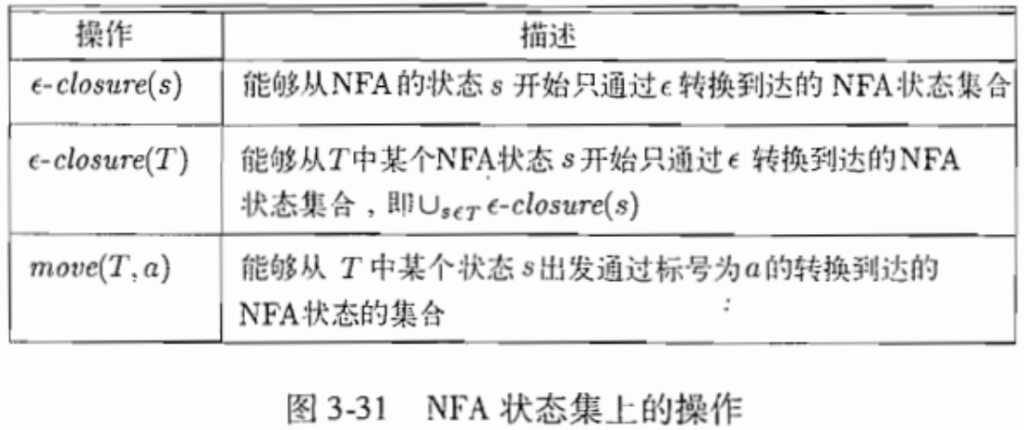


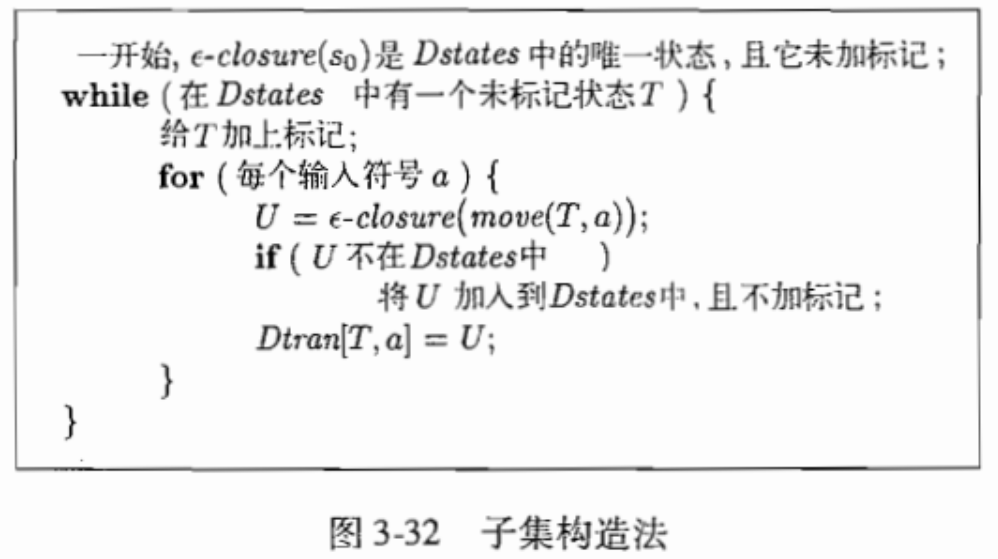
**确定有穷自动机（Deterministic Finite Automata, DFA）**

* **对于每一类字符有且只有一条离开该状态**、以该符号为标号的边，即同一类字符的转移不能去到两个地方
* **没有空串**



**从 NFA 到 DFA 的转换：子集构造（subset constmction）**

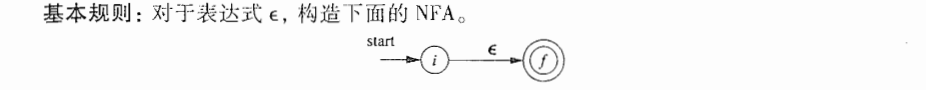


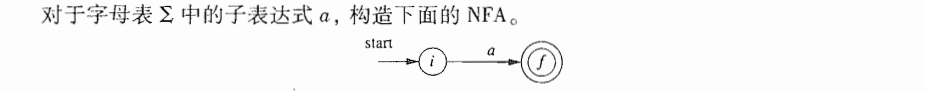


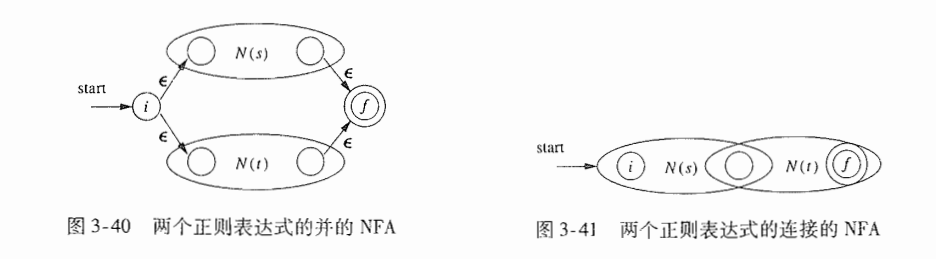
**算法总结：**先从 start（s1）开始把所有的可以**由空串到达**的位置合在一起**标为一个状态**，然后对于它的**每一种**输入所到达的位置 s2 – sn（及其空串可达位置）如果 s2 – sn 不是已经存在的状态，那么把他加入表格，依次对没有枚举完的状态（s2 – sn）再次做上述操作

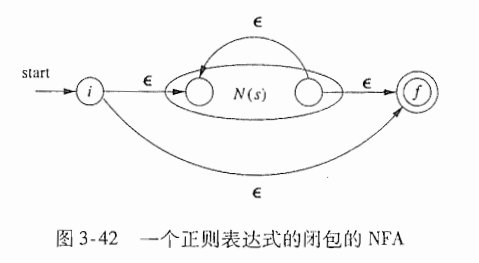
**人话总结：**先将相同的状态合并到一起，然后再将同类的字符转移到的状态合并成一个状态，反复执行（下一次执行就会再次合并相同状态即空串）

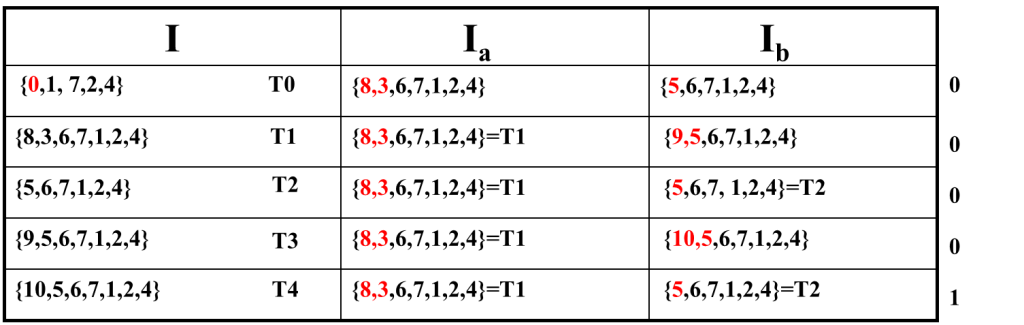
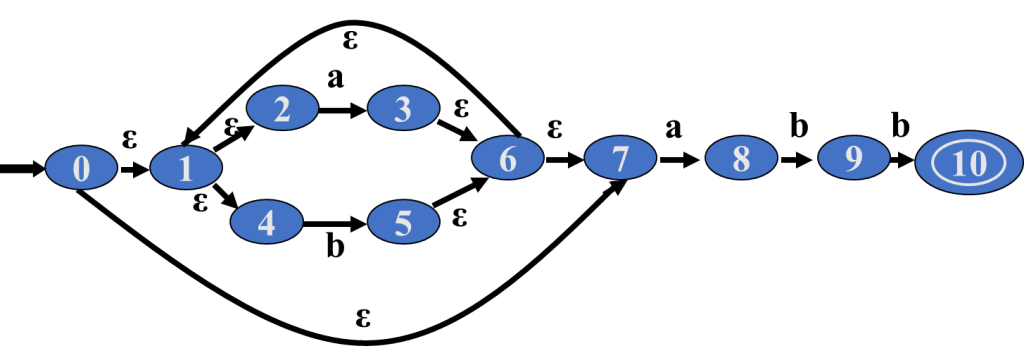
**如何将正则表达式转换为 NFA：**

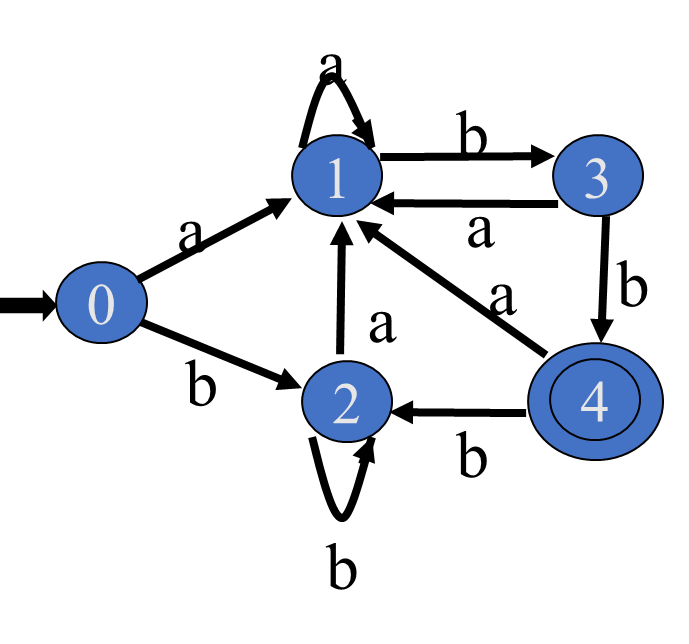










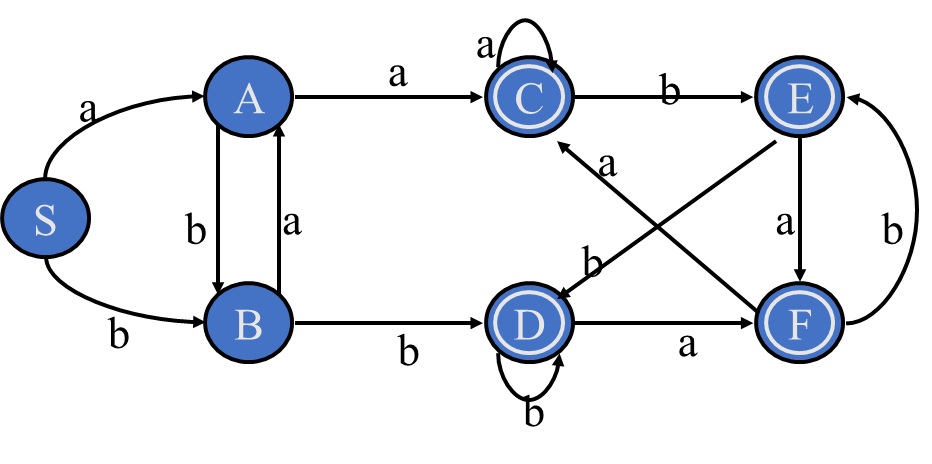


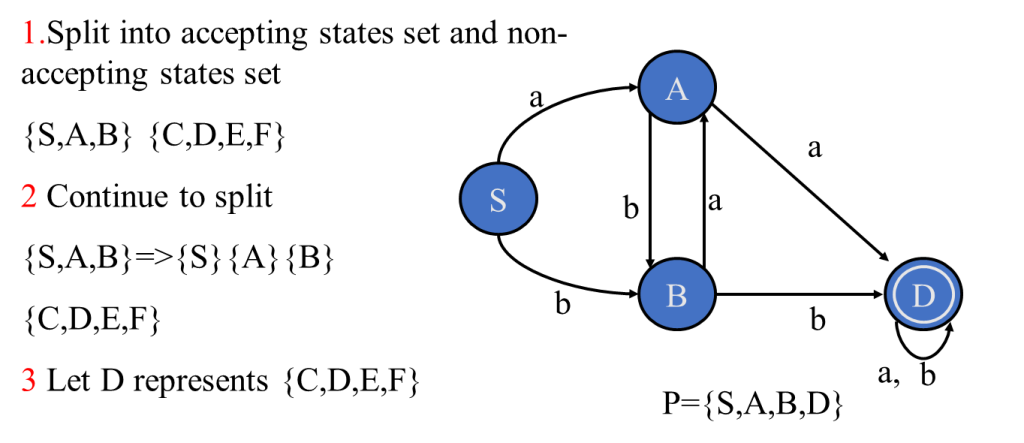
**最小化 DFA：**

* 先将 DFA**分成结束组和非结束组**（或者仅结束组）
* 我们现在**假定一组内的状态都是等价的**，那么这个**组内每一个状态对于任何输入**得到的**结果应当是一样的**，对一组内的每一个状态**输入字符**（只考虑从某状态出而不考虑入某状态）查看其输出是否一致（注意此时已经将另外**组内的状态看成是同种状态**，所以如果有输入导致的输出落在同一组不同状态内也算是一致的）然后**如果不一致则将所有不一致的分出去成为一组**
* 依次**执行直到没有组可分**，则现在就是最简的

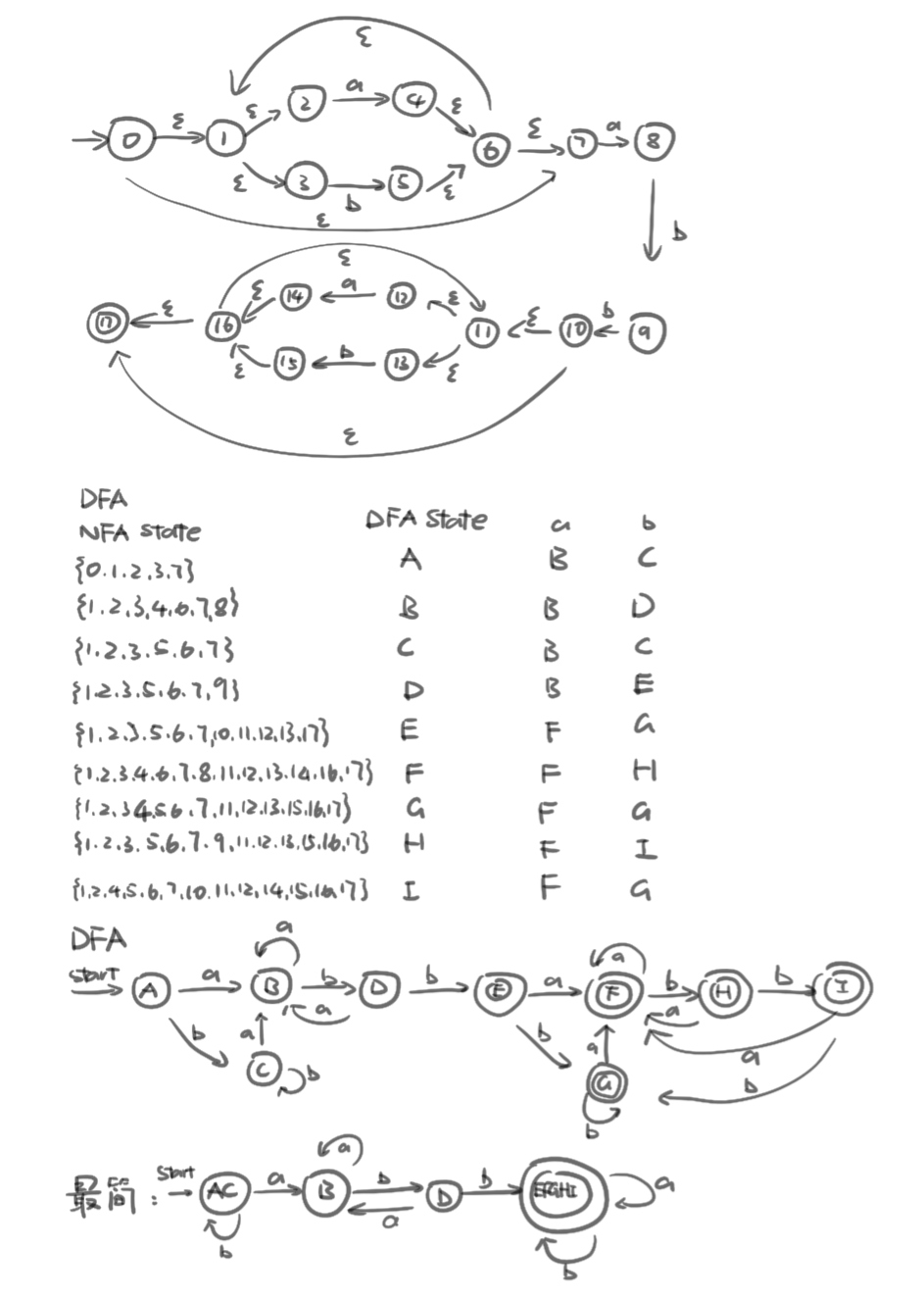
**例题\***

**化简 DFA**

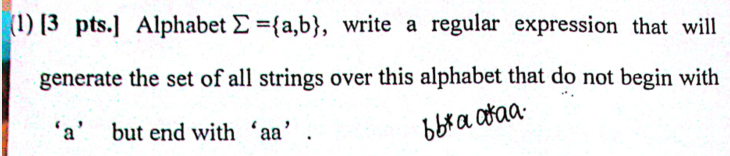
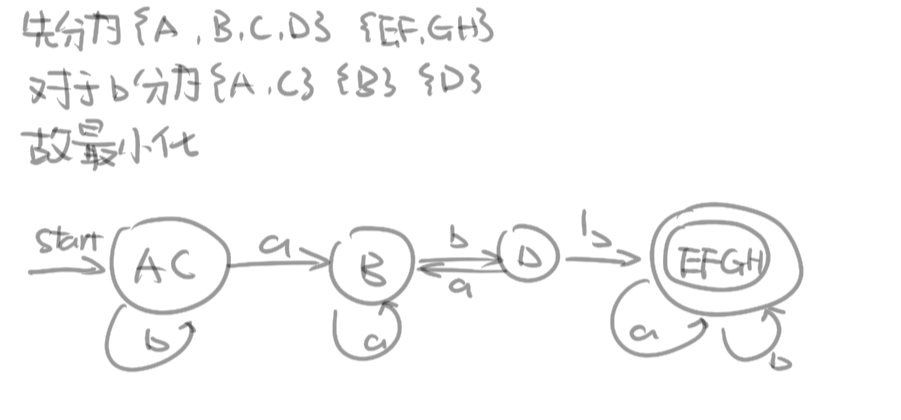
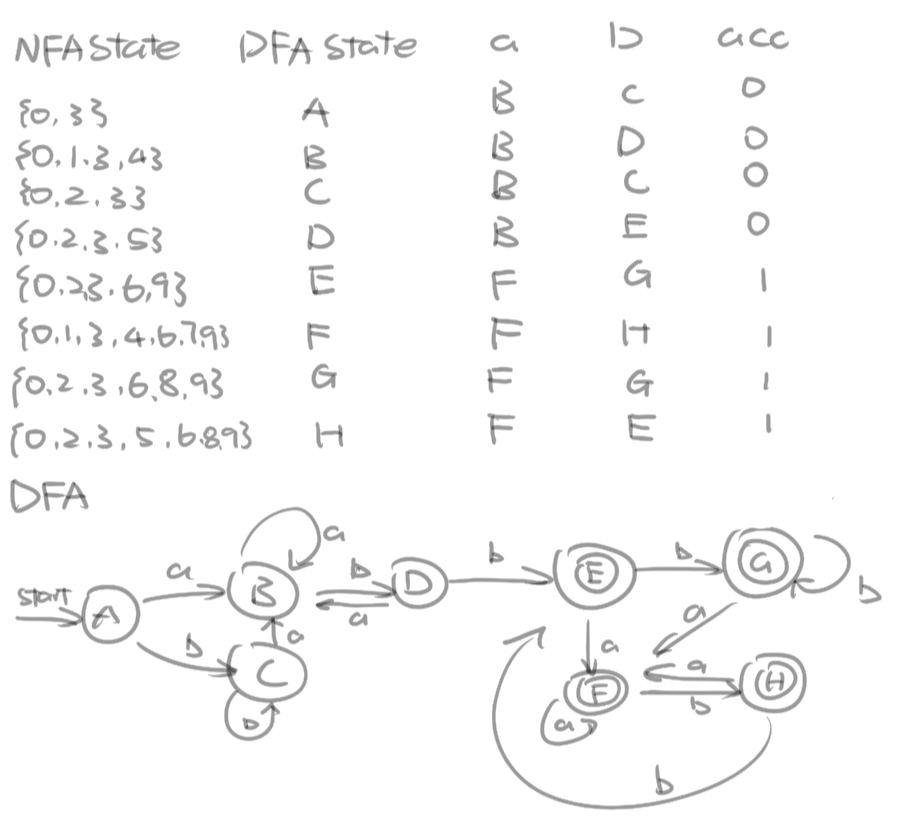
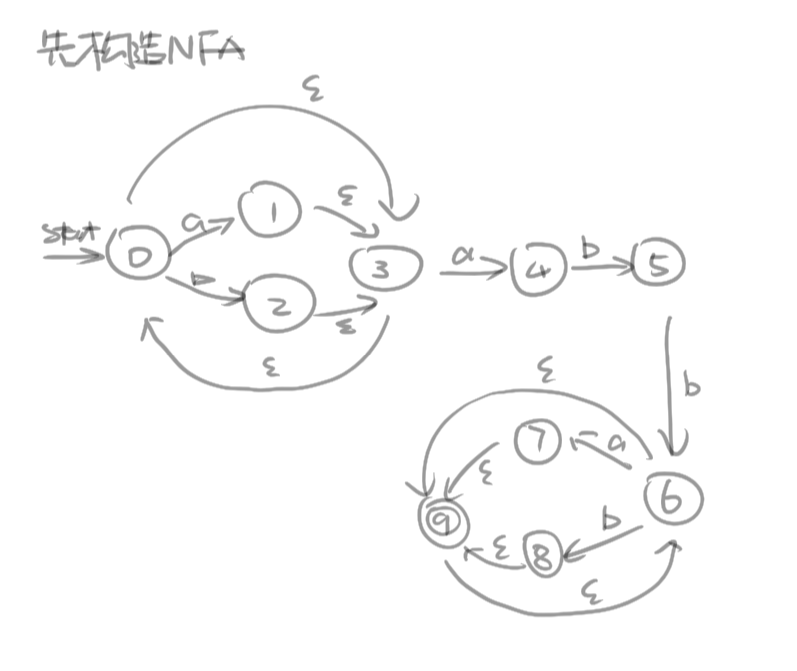




**3.7.3: d（完成子集构造法，和 DFA 最小化）(a|b)\*abb(a|b)\***



**其他解法（在绘制 NFA 时减少空串可以有效降低后面的步骤）：**



b(a|b)\*aa

