第三章 有限自动机与语法分析器

1. Pascal 和 C 语言的字母表是什么？

(答案)

pascal 语言的字母表是：
{0，1，¡¡ ，9}∪{a，¡¡ ，z}∪{A，¡¡ ，Z}∪{+，-，*，/，\，↑，?，?，_， (，)，[，]，;，:，=，<，
Enter，Space，Tab}

C 语言的字母表是：
_, 0¡¡9, a¡¡z, A¡¡Z, +, -, *, /, \, %, (,), [,], ., &, |, ！, =, #, {, }, ¡, ¡, ?, :，<，>， Enter，Space，Tab

(关闭)

2. 如果每个单词都以空白符结束(就象英文那样)，那么词法分析非常简单，但没有一个程序设计语言是定义的，

(答案)

(1) 写程序繁琐，给编程人员带来了负担，每次写一个单词都要写空白符，很容易出错。

(关闭)

3. 是 Pascal 程序段，试问词法分析阶段能发现哪些词法错误？

if a=1. then b: =1.0 else c: =1; a: =bc+d;

(答案)

if a=1. then b:=1.0 else c:=1; a:=bc+d;
if a=1.(1) then b:(2)=1.0 else c:(3)=1; (4)a:(5)=bc+d;
(1) 第一个等号右边的数字 1.写法不规范: 不符合实数的词法规范
(2) 字母 b,c 和第二个 a 后边的赋值符号不会被识别出来，因为：和＝之间有空格，会被识别为两个符号分别是
和等号¡ ＝¡

(关闭)

4．写出识别下列正则表达式定义的单词的 DFA：

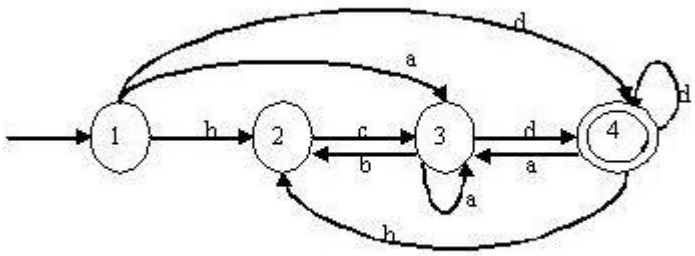
(1) $((a|bc)^*d)^+$

(2) $((0|1)^*(2|3)^+)|0011$

(3) (a Not (a)) aaa

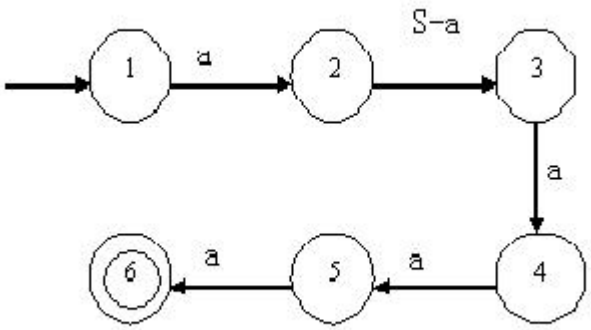
(答案)

(1)



(2)

(3)



(关闭)

5. 写出定义 Pascal 类定点 10 进制小数的正则表达式。其要求是前面和后面没有多余的 0，例如，0.0，0，123.01 和
是合法的；而 00.0，002456.1000 和 001.00 是非法的。

(答案)

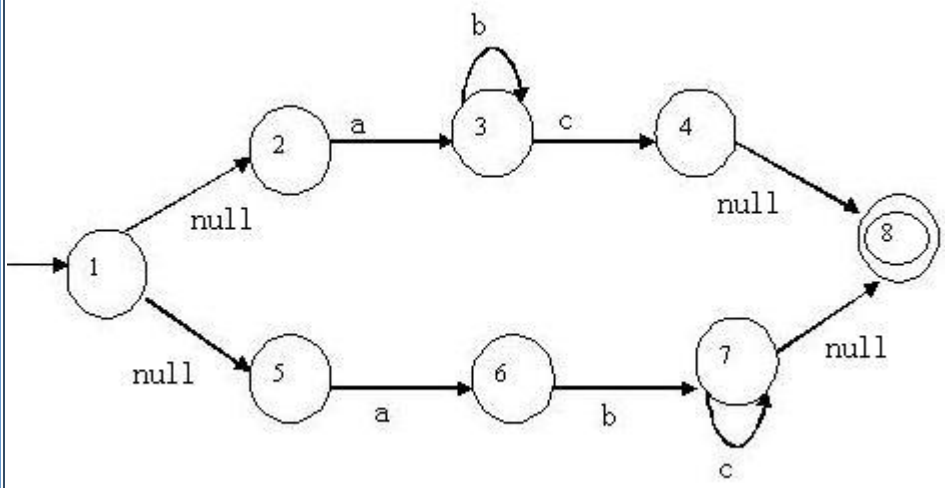
D = {0, 1, ¡, 9}
Decimal = (0|(D¨ {0})D*).(D*(D¨ {0})|0)

(关闭)

6. 构造一个 DFA，它接受的符号串集合等于正则表达式 $(ab^*c)|(abc^*)$ 所示的字符串集合。要求先构造 NFA
换成 DFA，最后加以极小化 。

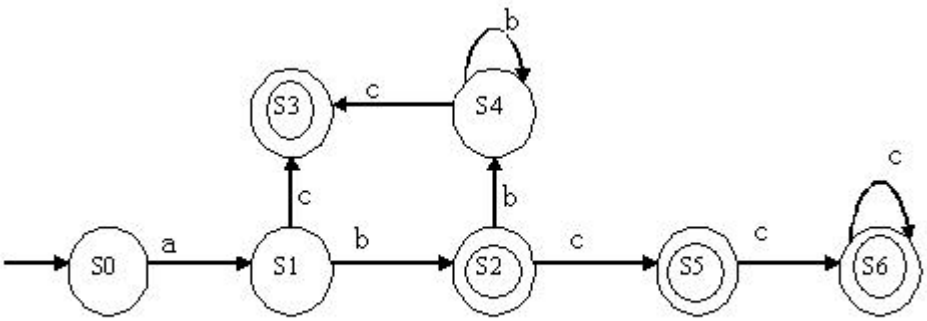
(答案)

(1) NFA 为:



(2)转换成 DFA:

DFA.State	a	b	c	NewStates
s0: [1,2,5]	[3,6]			{[3,6]}
s1: [3,6]		[3,7,8*]	[4,8*]	{[3,7,8*],[4,8*]}
s2*: [3,7,8*]		[3]	[4,7,8*]	{[4,8*],[3],[4,7,8*]}
s3*: [4,8*]				{[3],[4,7,8*]}
s4: [3]		[3]	[4,8*]	{[4,7,8*]}
s5*: [4,7,8*]			[7,8*]	{[7,8*]}
s6*: [7,8*]			[7,8*]	{ }



(3)化简 DFA:

DFA 的状态转换表:

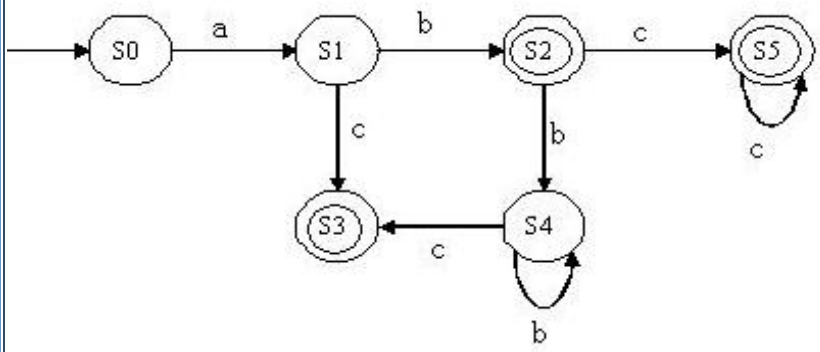
	a	b	c
S0	S1		
S1		S2*	S3*

S2*		S4	S5*
S3*			
S4		S4	S3*
S5*			S6*
S6*			S6*

其中状态 S5*与 S6*等价，化简得

	a	b	c
S0	S1		
S1		S2*	S3*
S2*		S4	S5*
S3*			
S4		S4	S3*
S5*			S5*

最后得 DFA:



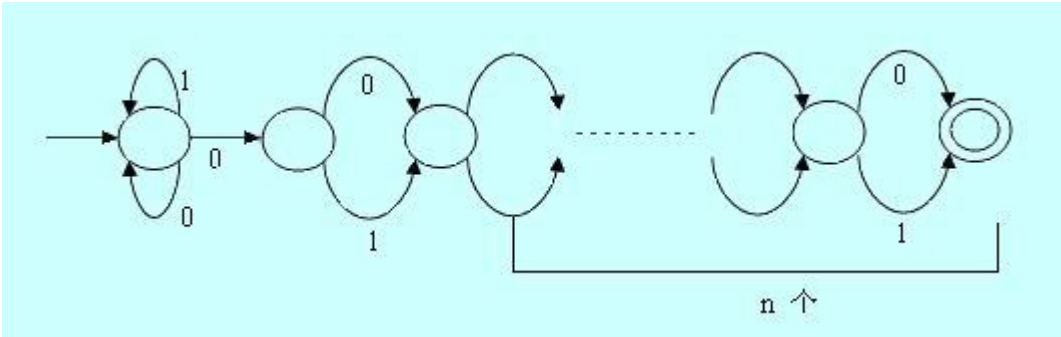
(关闭)

7. 出对应正则表达式 $(0|1)^*0(0|1)^n$ 的 NFA 示意图，其中 n 是大于等于 0 的整数。证明 NFA 的等价的

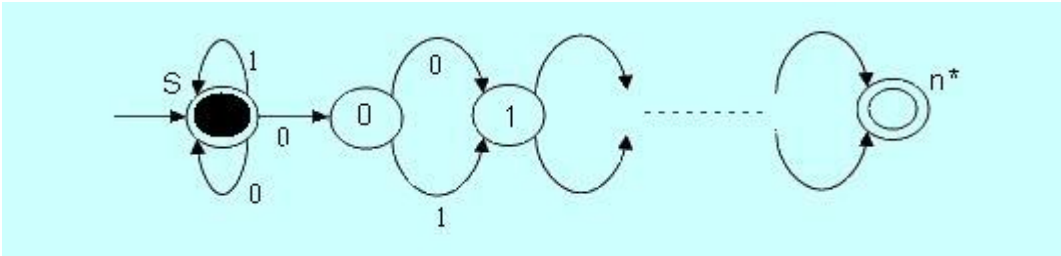
态个数将按指数级增大，即假设 NFA 的状态个数为 M，则 DEA 的状态个数是 M^n 级的。

(答案)

NFA



证明：



将 NFA 按书中算法转换成 DFA

DA.States	0	1	New States
[S]	[S,0]	[S]	{[S,0]}
[S,0]	[S,0,1]	[S,1]	{[S,0,1],[S,1]}
[S,1]	[S,0,2]	[S,2]	{[S,0,1],[S,0,2],[S,2]}
[S,2]	[S,0,3]	[S,3]	{[S,0,1],[S,0,2],[S,0,3],[S,3]}
...	...	...	...
[S,n-1]	[S,0,n*]	[S,n*]	{[S,0,1],¡,[S,0,n*],[S,n*]}
[S,n*]			{[S,0,1],¡,[S,0,n*]}
[S,0,1]	[S,0,1,2]	[S,1,2]	
[S,0,2]	[S,0,1,3]	[S,1,3]	
...	...	...	

到转换结束 DA.State 中有

[S]

[S,i] i = 0¡n

[S,i,j] i = 0¡n -1; j = i+1¡n

[S,i,j,k] i = 0¡n -2; j = i+1¡n -1; k = j+1¡n

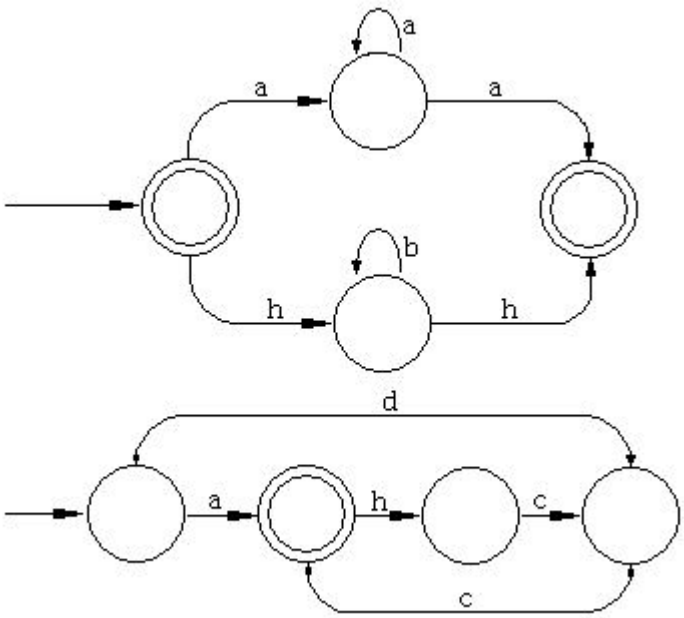
i

[S,t0,t1,¡,tn] t0 = 0,¡, tn = n

故 DFA 状态是 M^n 数量级。

(关闭)

8. 写出与下面 DFA 等价的正则表达式。



(答案)

(1)正则表达式为： (ε | a+a | hb*h)
(2)正则表达式为： a((hc|c)da)*

(关闭)

9. 当构造词法分析器时，根据单词的正则表达式定义首先构造 NFA，之后将 NFA 转换成 DFA，并用其 DF
法分析。从正则表达式到 NFA 的转换块，而 NFA 到 DFA 的转换比较慢， 因此在某些场合下，用 NFA 并
DFA 的转换思想进行词法分析可能是很有意思的。就是说，在 NFA 的一个状态下，遇到一个字符时，用 NFA 到
的构造技术确定下一状态。试写出算法。

(答案)

```
procedure NextState (instate: in State, a: in char, outState: out State)
  BEGIN
    for (元素 ss inState)
    BEGIN
      ss1:= close (MoveSet(ss, a, NFA.TT));
      outState := outState ∪ss1;
    END
  END
```

(关闭)

10．设 DFA1 和 DFA2 是同一字母表上的不同确定自动机，写出等价性判定算法。

(答案)

算法：

(1)将 DFA1 和 DFA2 分别化简得最简自动机 DA1 和 DA2，若 DA1 和 DA2 状态数不等，则不等价，停止；则转(2)

(2)将 DA2 的初状态改为与 DA1 的初状态相同，设为 S1_0；令 ss={S1_0},News={S1_0}

(3)从 News 中任选一个状态 Sn,对每一个字符 a，比较 DA1.TT(Sn,a)和 DA2.TT(Sn,a)

[1]若二者其中一个为空，不等价，停止

[2]若二者均不为空，但 DA2.TT(Sn,a)是 DA1 中状态且不等于 DA1.TT(Sn,a),不等价，停止

[3]若二者均不为空，DA2.TT(Sn,a)不是 DA1 中状态，则将 DA2.TT(Sn,a)改为与 DA1.TT(Sn,a)相同,若 DA1.TT(Sn,a)不在集合 ss 中，则将其加入 News;

(4)从 News 中删去 Sn,若 News 为空，二者等价，停止；否则转(3)

(关闭)

<<上一章◎ ◎回页首◎ ◎下一章>>