习题解答HW1

李晓奇

Fall 2023

2.1 a

题目 从 C 语言的参考手册确定它们形成输入字母表的字符集（不包括那些只可以出现在字符串或注释中的字符）。

源字符集（c语言源文件所用字符集）

- 26个大写字母: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
- 26个小写字母：a b c d e f g h I j k l m n o p q r s t u v w x y z
- 10个阿拉伯数字：0 1 2 3 4 5 6 7 8 9
- 特殊字符：! " # % & ' () * + , - . / : ; < = > ? [\ ] ^ _ { | } ~
- 空白字符：空格、水平制表符、垂直制表符、换页符、换行符

参见C (programming language) - Wikipedia

2.2

题目 在下面的 C 函数中按序列出所有的记号，并给每个记号以合理的属性值。

蓝色表示记号名，绿色表示符号表对应指针

<type, long>, <id, gcd>, <LeftParen>,
<type, long>, <id, p>, <Comma>,
<type, long>, <id, q>, <RightParen>,
<LeftBracket>, <if>, <LeftParen>, <id, p>,
<mod_op>, <id, q>, <relation, ==>,

```
long gcd(long p,long q) {
      if (p%q==0)
            /* then part */
            return q;
      else
            /* else part*/
            return gcd(q,p% q);
}
```

2.2 (Cont)

题目 在下面的 C 函数中按序列出所有的记号，并给每个记号以合理的属性值。

蓝色表示记号名，绿色表示符号表对应指针

<number, 0>, <RightParen>, <return>,
<id, q>, <Semicolon>, <else>, <return>,
<id, gcd>, <LeftParen>, <id, q>,
<Comma>, <id, p>, <mod_op>,
<id, q>, <RightParen>, <Semicolon>,
<RightBracket>

```
long gcd(long p,long q) {
      if (p%q==0)
            /* then part */
            return q;
      else
            /* else part*/
            return gcd(q,p% q);
}
```

2.3 a

题目 叙述由下列正规式描述的语言： 0 (0 | 1) * 0

由0或1组成，以0开头结尾，且长度至少为2的全部数字串。

2.4 h

题目 为下列语言写出正规定义：所有不含子串 011 的 0 和 1 的串

符合题意的正规式：$1^*(01|0)^*$

分析：

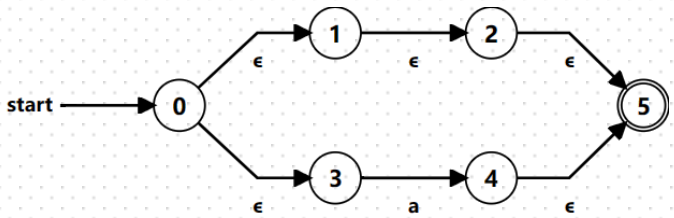
- $(01|0)^*$ 代表 0 后至多跟随一个 1
- 此外 0 前可以有任意个 1，所以有 1^*

在线测试：regex101: build, test, and debug regex

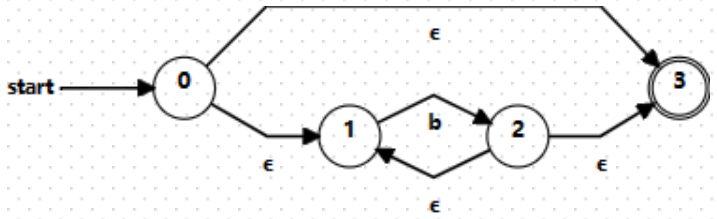
2.7 c

题目 用算法 **2.4** 为下列正规式构造不确定有限自动机，给出它们处理输入串 **ababbab** 的状态转换序列：$((\varepsilon \mid a)\, b^*)^*$

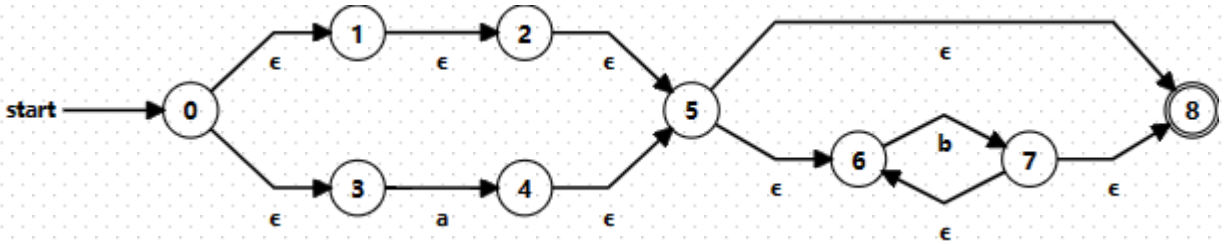
构造NFA的过程：



$\varepsilon \mid a$



b^*

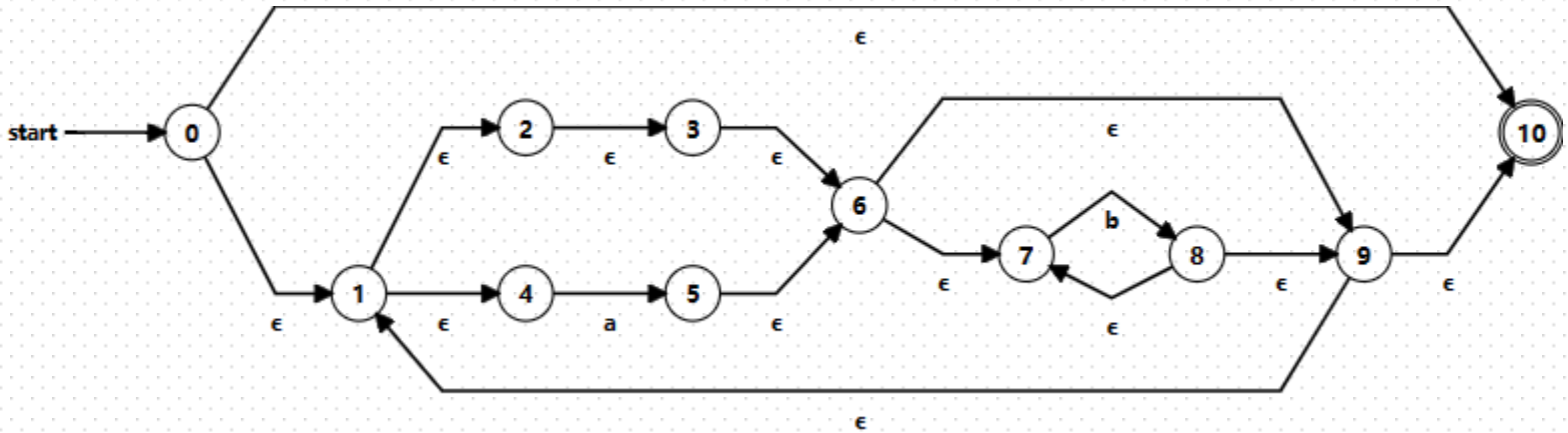


$(\varepsilon \mid a)\, b^*$

2.7 c (Cont)

题目 用算法 **2.4** 为下列正规式构造不确定有限自动机，给出它们处理输入串 **ababbab** 的状态转换序列：$((\varepsilon \mid a)b^*)^*$

构造NFA的过程：



$((\varepsilon \mid a)b^*)^*$

说明：根据题目要求，需要按照算法 2.4 构造，考试中如果未要求，可以构造更简单的NFA

2.7 c (Cont)

题目 用算法 **2.4** 为下列正规式构造不确定有限自动机，给出它们处理输入串 **ababbab** 的状态转换序列：$((\varepsilon \mid a)b^*)^*$

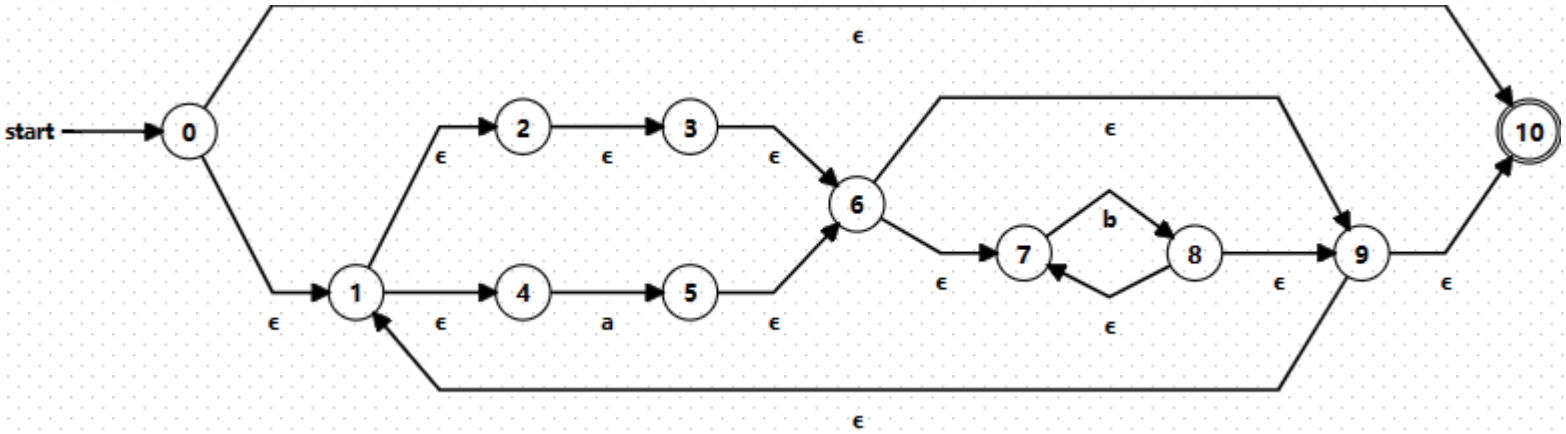
输入 ababbab 时的状态转换序列：

0 → 1 → 4 → 5 → 6 → 7 → 8 → 9　　// 识别 ab

→ 1 → 4 → 5 → 6 → 7 → 8 → 7 → 8 → 9　　// 识别 abb

→ 1 → 4 → 5 → 6 → 7 → 8 → 9　　// 识别 ab

→ 10

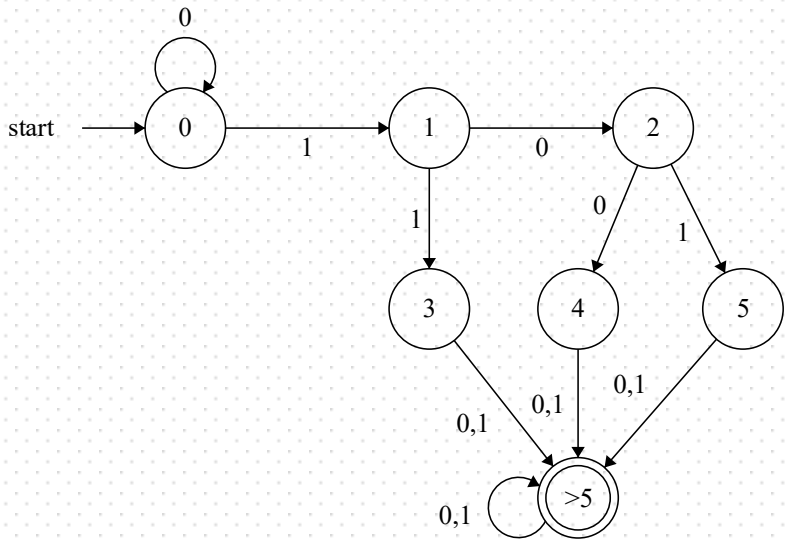


2.15

题目 构造一个最简的 **DFA**，它接受所有大于**101**的二进制整数。

根据输入，可以判断当前得到的数字，

手工构造如下DFA，状态名代表了当前得到的数字：



2.15 (Cont)

题目 构造一个最简的 **DFA**，它接受所有大于**101**的二进制整数。

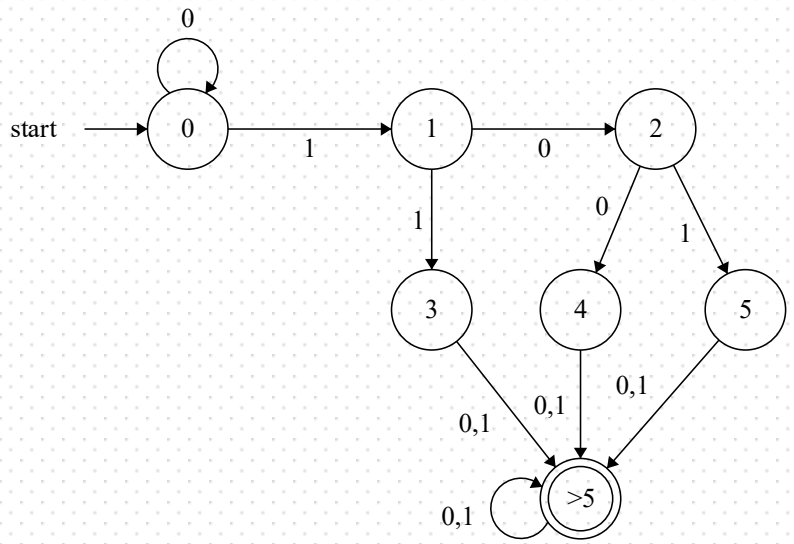
然后根据算法 2.3 得到极小DFA

1. 首先检查状态转换函数是否为**全函数**:

   任意状态都存在对 0 或 1 的转换

2. 划分状态子集，得到 **F** 和 **S - F**

   F = { “>5” } S – F = { 0, 1, 2, 3, 4, 5 }



2.15 (Cont)

题目 构造一个最简的 **DFA**，它接受所有大于**101**的二进制整数。

然后根据算法 2.3 得到极小DFA

3. 构造新的划分

F 中只有一个状态，不可再分：{ “>5” }

考察 S – F = { 0, 1, 2, 3, 4, 5 }

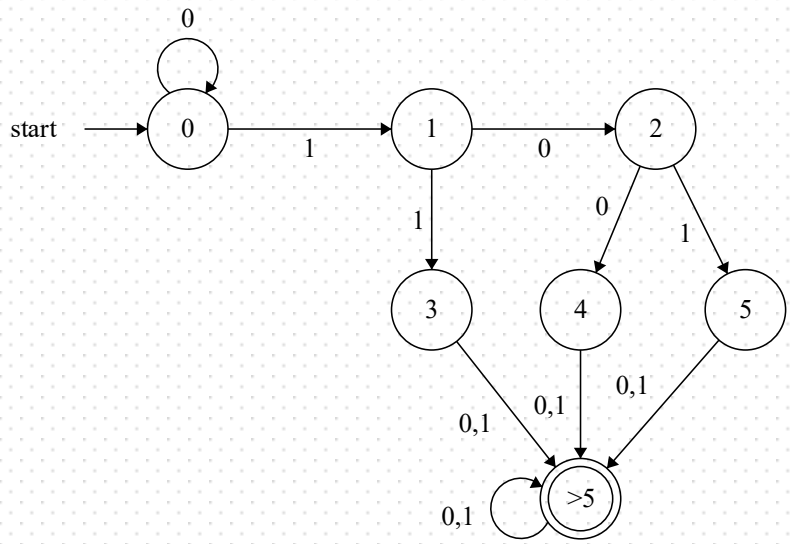
对于输入 0，状态 0、1、2 分别转换到 0、2、4

而状态 3、4、5 均转换到 “>5”

所以将 S – F 划分为：{ 0 }{ 1 }{ 2 }{ 3, 4, 5 }

状态 3、4、5 的 0 转换与 1 转换都相同（均到达状态 “>5”）

故子集 { 3, 4, 5 } 不可再分



2.15 (Cont)

题目 构造一个最简的 **DFA**，它接受所有大于**101**的二进制整数。

然后根据算法 2.3 得到极小DFA

4. 极小 DFA 的状态转换表

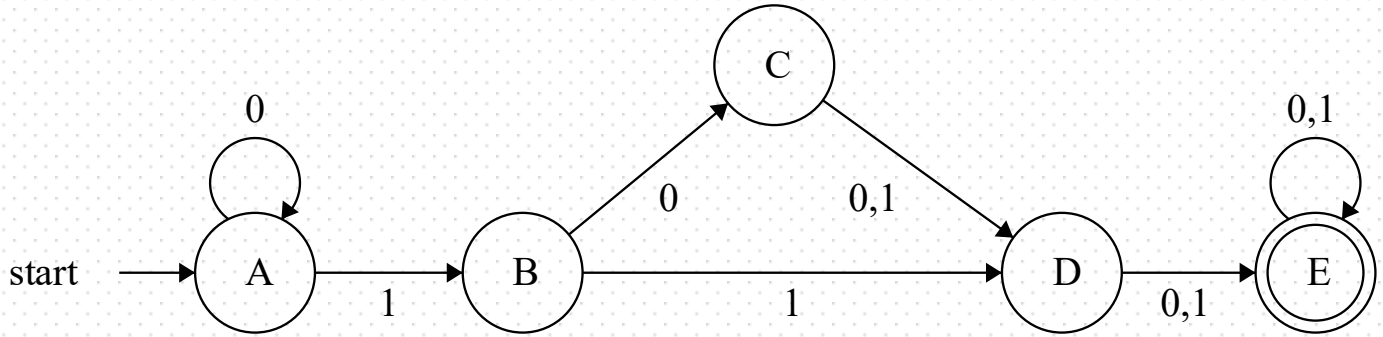
状态	输入符号	
	0	1
A({0})	A	B
B({1})	C	D
C({2})	D	D
D({3,4,5}	E	E
E({6})	E	E

2.15 (Cont)

题目 构造一个最简的 **DFA**，它接受所有大于**101**的二进制整数。

然后根据算法 2.3 得到极小DFA

5. 极小 DFA 的状态转换图



2.15 (Cont)

题目 构造一个最简的 **DFA**，它接受所有大于**101**的二进制整数。

方法二 先写出正则式，再根据算法依次构造NFA、DFA、最简DFA

从字符串的左侧开始匹配，只要匹配到特定序列以及长度的的数字，即可判断出大于101：

- 如果在左侧匹配到11，且长度大于等于3，则该数字大于101
- 如果在左侧匹配到10，且长度大于等于4，则该数字大于101
- 此外可以有任意多个前导0

由此可以写出正则式：$0^*1\,(\,00\mid 01\mid 1\,)(\,0\mid 1\,)^+$。

然后按照教材算法可以构造出相应的NFA、DFA以及最简DFA