第十章 有限状态自动机

前言

- 输入一个字符串，判断其是否是合法的C语言标识符；
- 输入一个字符串，判断其是否是 $a^n b^n c^n$ 形式(即先输入a、再输入b、最后输入c，且输入的a、b、c的个数相同）；
- 读取C源代码，去除注释；
- ……
- 针对类似的字符串识别、处理问题，建立有限状态自动机模型，可以为分析、求解带来很大的帮助。

什么是有限状态自动机？

6.1 基本概念

例1：打电话 (自动机在通信领域的应用)。

在一次呼叫中，从建立连接到通话完毕，要经历摘机，拨号，应答，进行通话等过程，话机的状态及状态迁移如下所示。

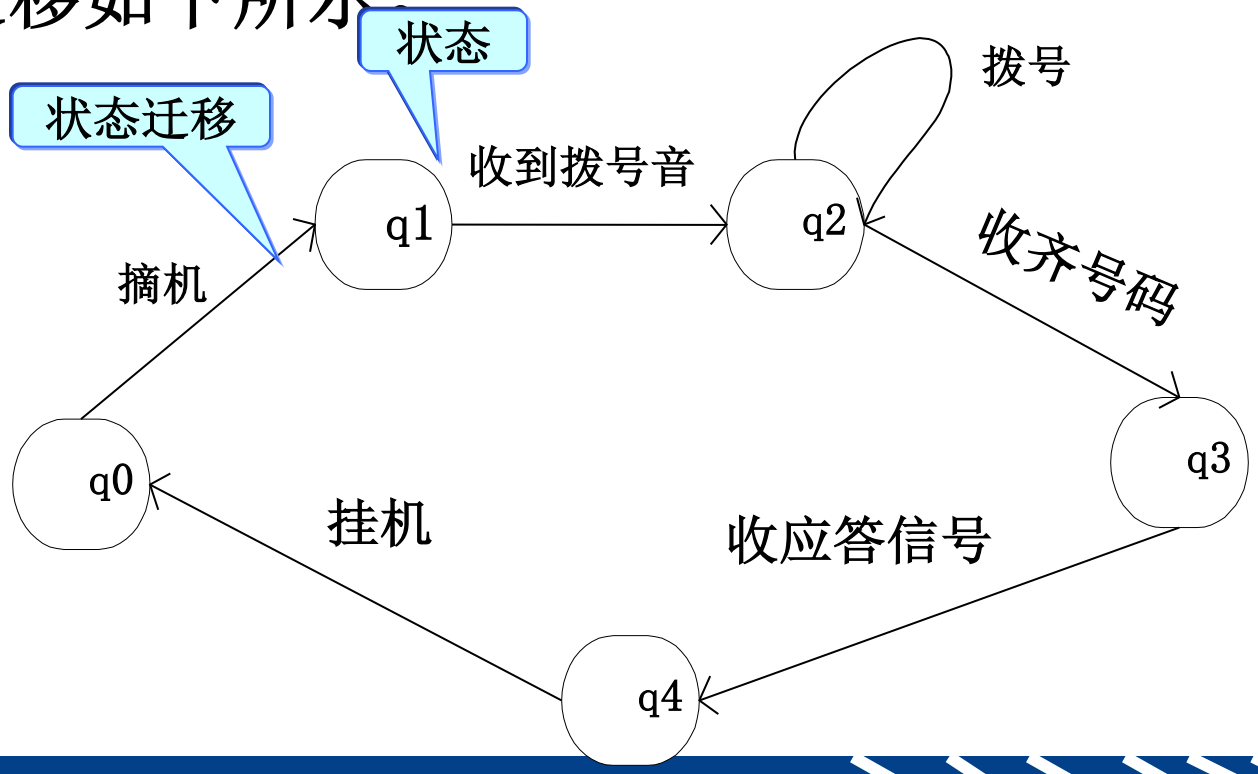
q0:空闲状态

q1:等待拨号音状态

q2:可以拨号状态

q3:等待应答状态

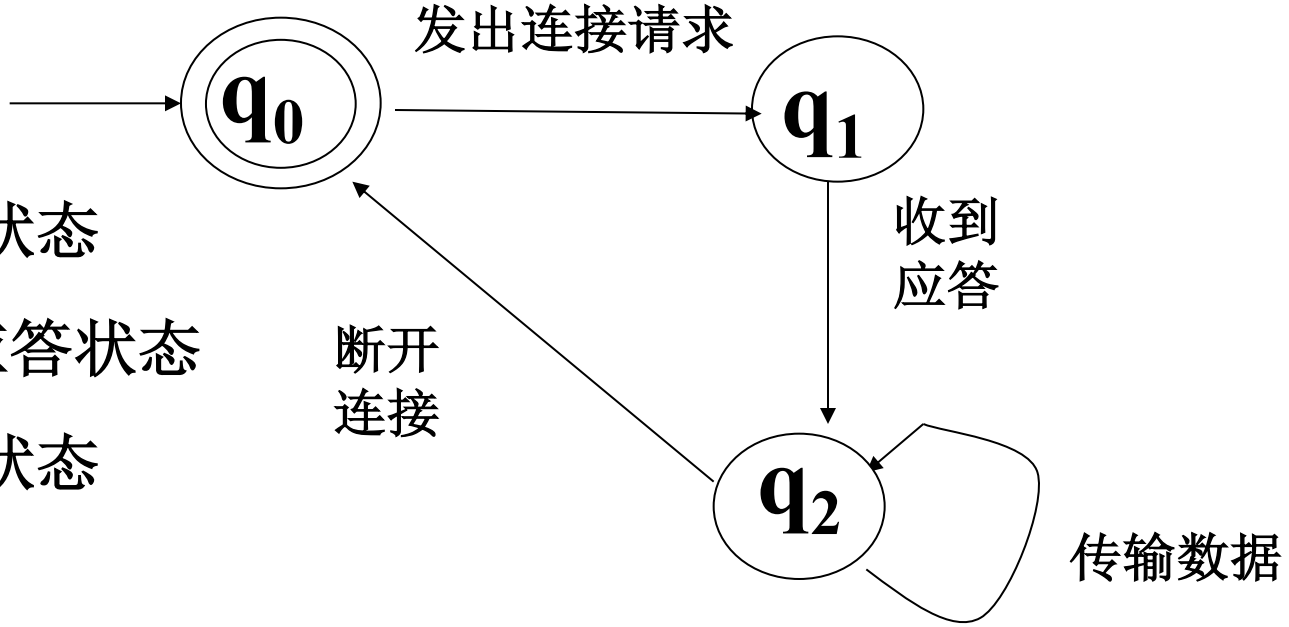
q4:通话状态



6.1 基本概念

例2：串口通信

两台微机通过串口通信，需在两台机器间建立好连接后，才可以传递数据，可以使用有限状态自动机，描述串口通信的状态。



q0：空闲状态

q1: 等待应答状态

q2：通信状态

6.1 基本概念

- 电话和串口都可抽象为有限自动机

1. 对象处于某一相对稳定的状态下；
2. 某个事件（输入）发生；
3. 这一事件引起一串处理发生，包括执行特定的功能，产生相应的输出等；
4. 处理结束，对象迁移到一个新的相对稳定状态。

6.1 基本概念

- 什么是有限状态自动机？

  是一种具有离散输入/输出系统的数学模型，简称 有限自动机。这一系统具有任意有限数量的内部“状态”。

  - 状态：一个标识，能区分自动机在不同时刻的状况。有限状态系统具有任意有限数目的内部“状态”

- 自动机接受一定的输入，执行一定的动作，产生一定的结果。

6.1 基本概念

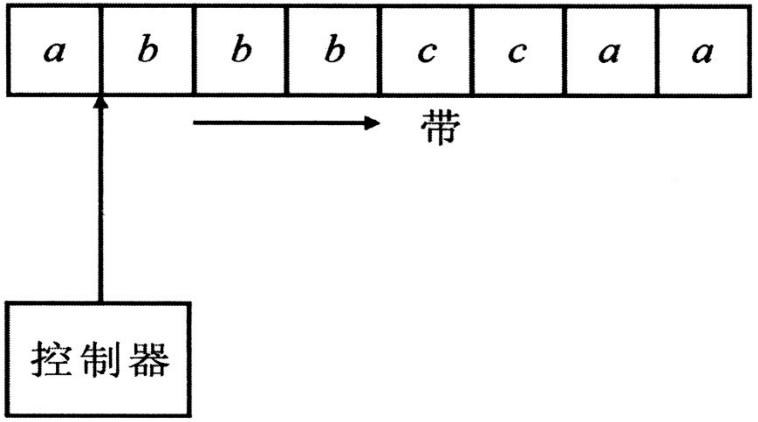
- 自动机的本质：根据状态、输入和规则决定下一个状态

  状态 + 输入（激励）+ 规则 —> 状态迁移

- 可能的状态、运行的规则都是事先确定的。一旦开始运行，就按照事先确定的规则工作，因此叫“自动机”。使用状态迁移描述整个工作过程。
- 大量通信软件的基本工作机制都是有限状态自动机。自动机理论在通信领域中的应用极为广泛

6.1 基本概念

有限自动机示意图



组成

一个有限控制器

一个读头

一条写有字符的输入带

工作原理：读头在输入带上从左向右移动，每当读头从带上读到一个字符时，便引起控制器状态的改变，同时读头右移一个符号的位置。

6.1 基本概念

- 控制器包括有限个状态，状态与状态之间存在着某种转换关系。每当在某一状态下读入一个字符时，便使状态发生改变（称为状态转换）。
- 状态转换包括以下几种情况：1) 转换到其自身，即保持当前状态不变；2) 转换的后继状态只有一个；3) 转换的后继状态有若干个。
- 如果一个有限自动机每次转换的后继状态都是唯一的，称为确定的有限自动机（DFA）；如果转换的后继状态不是唯一的，则称为不确定的有限自动机（NFA）。
- 通常把有限自动机开始工作的状态称为“初始状态”，把结束工作的状态称为“终止状态”或“接受状态”。

6.1 基本概念

北京郵電大學 Beijing University of Posts and Telecommunications

- 确定的有限自动机的形式化定义

确定的有限自动机是一个五元组 $M=(Q,T,\delta,q_0,F)$

其中：

Q ：有限的状态集合；

T ：有限的输入字母表；

δ ：转换函数，是$Q\times T$ 到Q 的映射；

q_0 ：初始状态，$q_0\in Q$ ；（初始状态只有一个）

F ：终止状态集，$F\subseteq Q$ ；

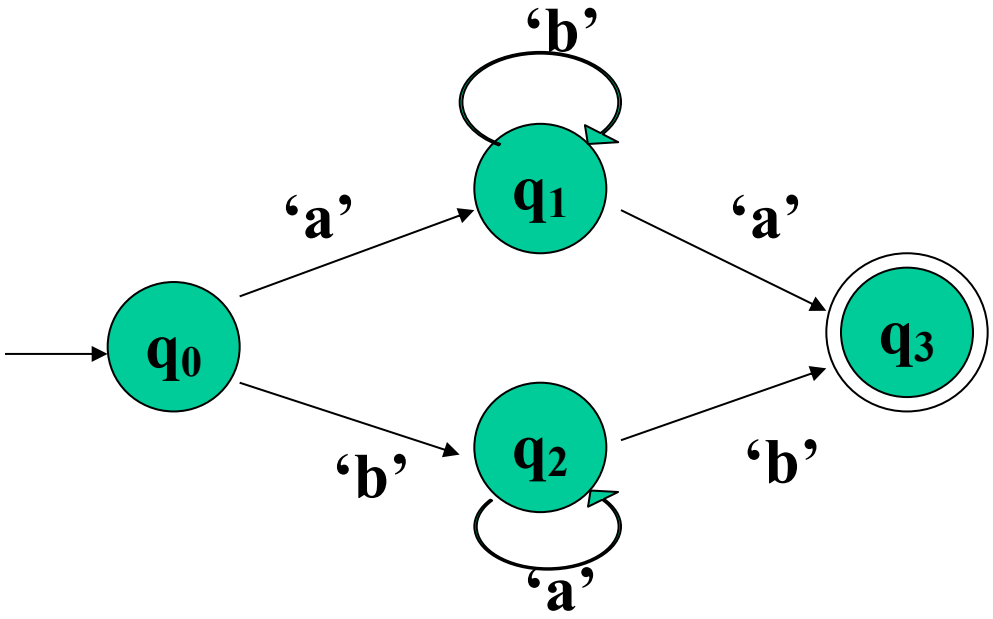
6.1 基本概念

- 为了描述一个有限自动机的工作状况，可采用状态转换图。状态转换图是一个有向图，图中的每个节点表示一种状态，一条边（或弧）表示一个转换关系。

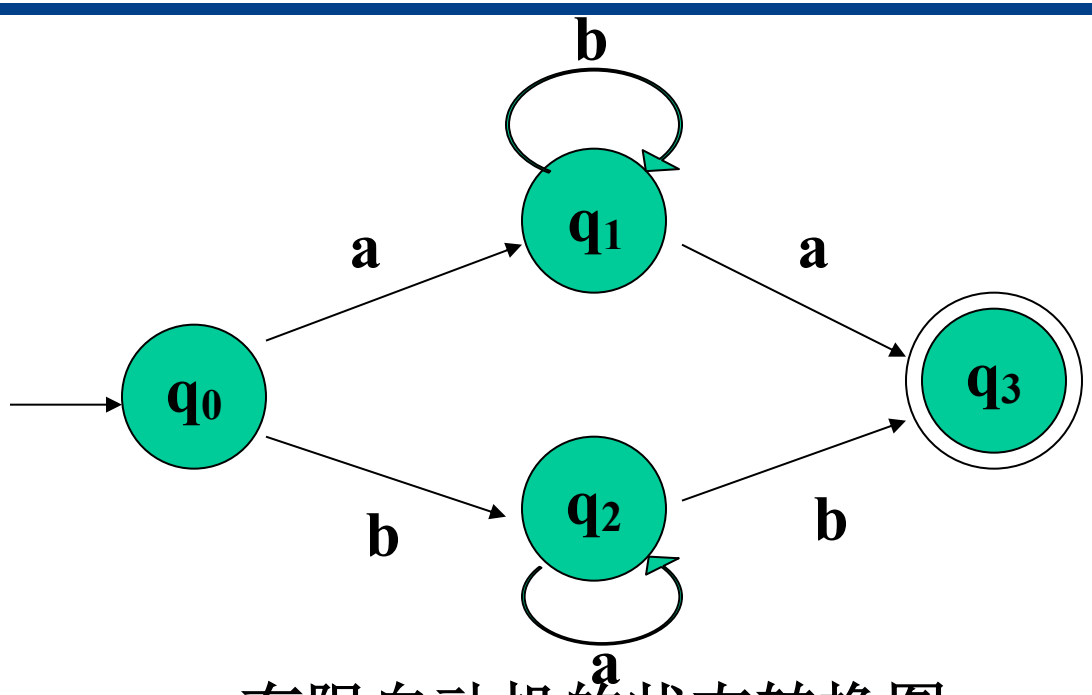
初始状态：q0

终止状态：q3

控制器的状态集合：{q0,q1,q2,q3}



用于识别输入的字符串是否是 ab^na 或者 ba^nb 形式的有限自动机。



有限自动机的状态转换图

状态集合 $Q=\{q_0,q_1,q_2,q_3\}$

字母表 $T=\{a,b\}$

初始状态 q_0

终止状态集 $F=\{q_3\}$

转换函数

$$\delta(q_0,a)=q_1 \quad \delta(q_0,b)=q_2 \quad \delta(q_1,a)=q_3$$

$$\delta(q_1,b)=q_1 \quad \delta(q_2,a)=q_2 \quad \delta(q_2,b)=q_3$$

$$\delta(q_3,a)=\varnothing \quad \delta(q_3,b)=\varnothing$$

6.1 基本概念

- 存储程序的计算机本身也可以认为是一个有限状态机。输入输出都是离散量，某时刻的状态由当时进行的操作、寄存器、主存储器和辅助存储器中存储内容确定。
- 一个运行中的程序在不同时刻也具有不同的状态，程序执行中的状态就是各种“变量”当时所存放的值。比如我们设计的求**5！**的程序，初始进入循环前的状态是：**(p=1 and i=2)**, 执行完循环后的状态是：（**p=40 and i=6**）。

6.2 程序设计实例研究

应用有限自动机模型求解问题的核心问题就是抽象出状态，描述出状态转移图和状态转移函数

应用有限自动机解题步骤

1、确定输入集

2、绘制状态迁移图（确定状态，在每一个状态下对输入进行分类，针对每一类输入，确定下一个状态）

3、确定状态转移函数（在某状态下，接收到某一字符后，自动机要执行的操作，以及迁移到的下一状态）

6.2 程序设计实例研究

问题分析：探测器向计算机发出的信号可以认为是一个任意长的字符序列（以EOF结束），比如：“011011000111101”，这样设计程序实际上演变为读取该字符序列，然后进行相关的操作。

观测时长：字符序列中0的个数(6秒)；
车辆总数：字符序列中1的个数(9辆)；
两车间最大时间间隔：两个1之间的最大连续0的个数(3秒)；

号，统计出观测的时长、在观测时长内通过的车辆总数、以及两辆车之间最大的时间间隔。

6.2 程序设计实例研究

- 重新审题：

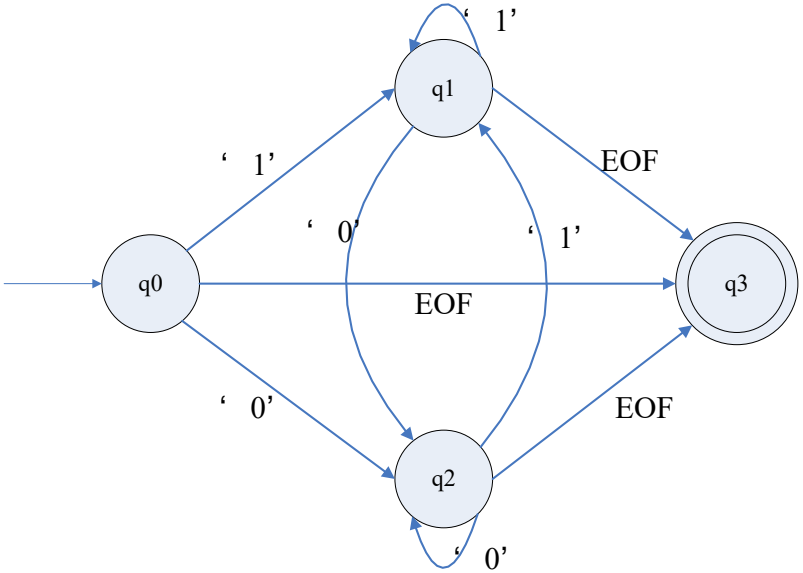
– 程序读入以**EOF**结束的由‘**0**’或者‘**1**’组成的字符串，这个字符串可以映射为有限自动机模型中的字符输入带

– 我们的任务就是设计控制器程序逐字符地读取输入带，进行处理并引起控制器的状态改变，最终产生输出，因此这个问题的求解就抽象为一个有限自动机。

交通灯观测实例研究-解法1

1、确定输入集T＝{‘1’，‘0’，EOF}

2、绘制状态迁移图（每一个状态下对输入集进行分类，确定状态）



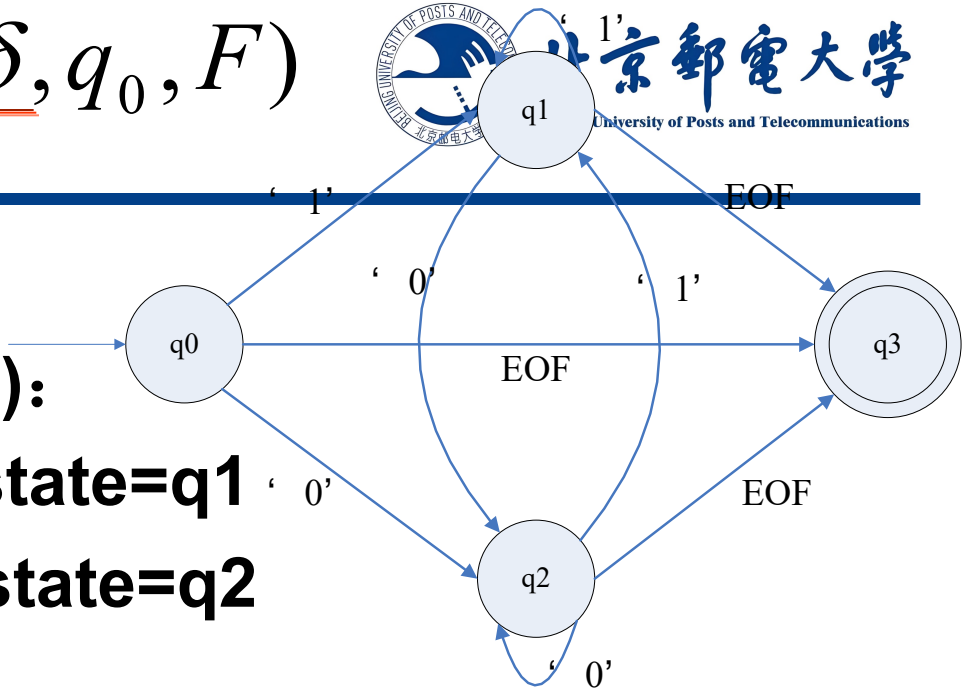
q0:初始状态

q1:读入字符'1'即进入q1状态，说明前一个字符是‘1’

q2:读入字符'0'即进入q2状态，说明前一个字符是‘0’

q3:终止状态

$$M=(Q,T,\delta,q_0,F)$$

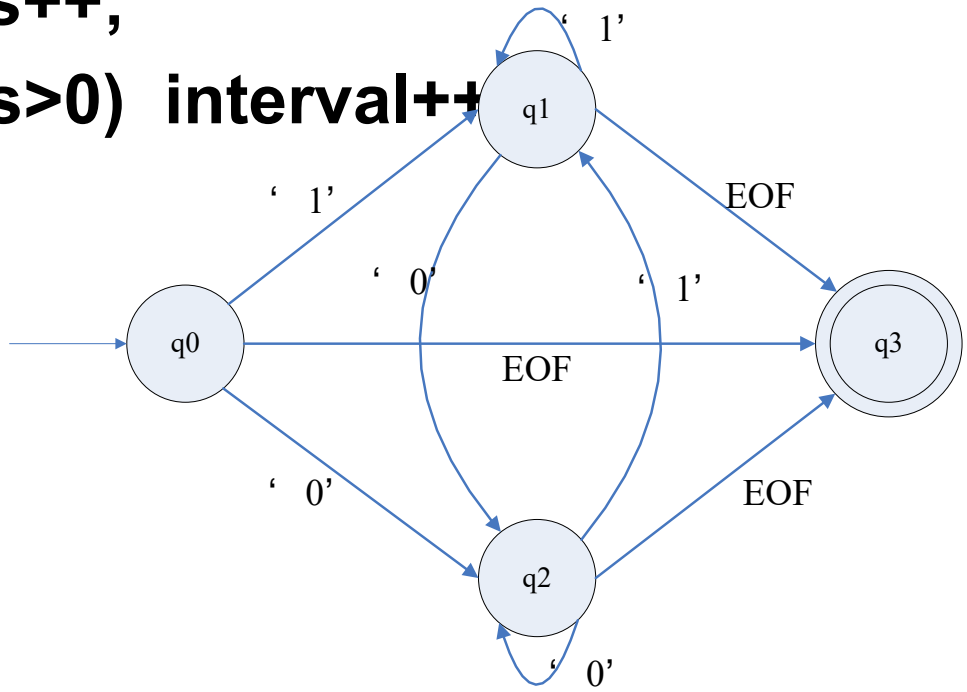


3、确定转换函数

- **当前状态是q0 (state==q0)：**
  - **读入'1'：vehicles++; state=q1**
  - **读入'0'：seconds++; state=q2**
  - **读入EOF： state=q3**
- **当前状态是q1 (state==q1，说明前一个字符是'1')：**
  - **读入'1'(11)：vehicles++;**
  - **读入'0'(10)：interval=1;seconds++; state=q2**
  - **读入EOF： state=q3**

- 当前状态是**q2 (state==q2，说明前一个字符是'0')：**
  - 读入**'1'**(01)：**if(vehicles>0) 处理最大时长**
    vehicles++; state=q1
  - 读入**'0'**(00)：**seconds++;**
    if(vehicles>0) interval++
  - 读入**EOF： state=q3**

终止状态集**F={q3}**



解法2对应代码：

解法1对应代码：

此处自动机会有不同处理

有限自动机解题通用处理模式

```
#include<stdio.h>
#define START 0
#define GET1 1
#define GET0 2
#define END 3

main()
{
   char signal;
   int vehicles,seconds,interval,longest;
   int state;

   state=START;
   vehicles=0; seconds=0;longest=0;
   printf("input signals,'#'to end:\n");
```

交通观测灯(自动机)-解法1.c

```
while(state!=END){
   signal=getchar();
   switch(state){
     case START:/*若当前状态是初始状态，则只是简单的对时钟数或者车辆数加1*/
         switch(signal){
            case '1': vehicles++;
                      state=GET1; /*状态迁移到1*/
                       break;
           case '0':
                      seconds++;
                      state=GET0;
                      break;
           case '#':
                      state=END;
                       break;
          }
           break;
```

```
case GET1:/*若当前状态是q1，即上一个信号是'1'*/
    switch(signal){
       case '1': /*当前读入信号是'1'，则只是简单将车辆加1*/
            vehicles++;
            break;
       case '0': /*前信号是'0'，则要开始计数两个1之间的时钟间隔interval*/
            seconds++;
            interval=1;/*开始计数interval*/
            state=GET0;
            break;
       case '#':
           state=END;
           break;
       }
   break;
```

```
case GET0:        /*若当前状态是q2，即上一个信号是'0'*/
    switch(signal){
      case '0': /*当前读入信号是'0',则需要判断是否要将interval加1*/
          seconds++;
          if(interval>0)  /*interval>0,表示状态是从状态1转移到状态2的，
                            所以interval需要继续加1*/
              interval++;
              break;
      case '1': /*当前读入信号是'1',需要判断是否要处理最长时间间隔*/
          vehicles++;
         if (interval>0)/*interval>0,表示状态是从状态q1转移到状态q2的，所
以                  在状态迁移回q1之前需要处理最大时间间隔*/
            if(interval>longest)
                longest=interval;
           interval=0;
           state=GET1;
          break;
     case '#':state=END;     break;
}
```

```
                }/*switch*/
    }/*while*/

    printf("%d vehicles passed in %d seconds\n",vehicles,seconds);
    printf("the longest gap was %d  seconds\n",longest);
    system("PAUSE");
    return 0;

}
```

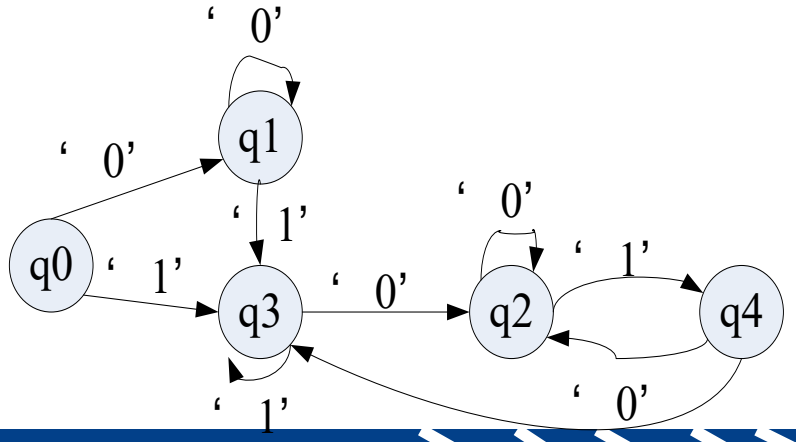
交通灯观测实例研究-解法2

北京郵電大學 Beijing University of Posts and Telecommunications

1、确定输入集T＝{‘1’，‘0’，‘#’}

2、对输入集中的‘1’和‘0’进行进一步分类，确定状态:

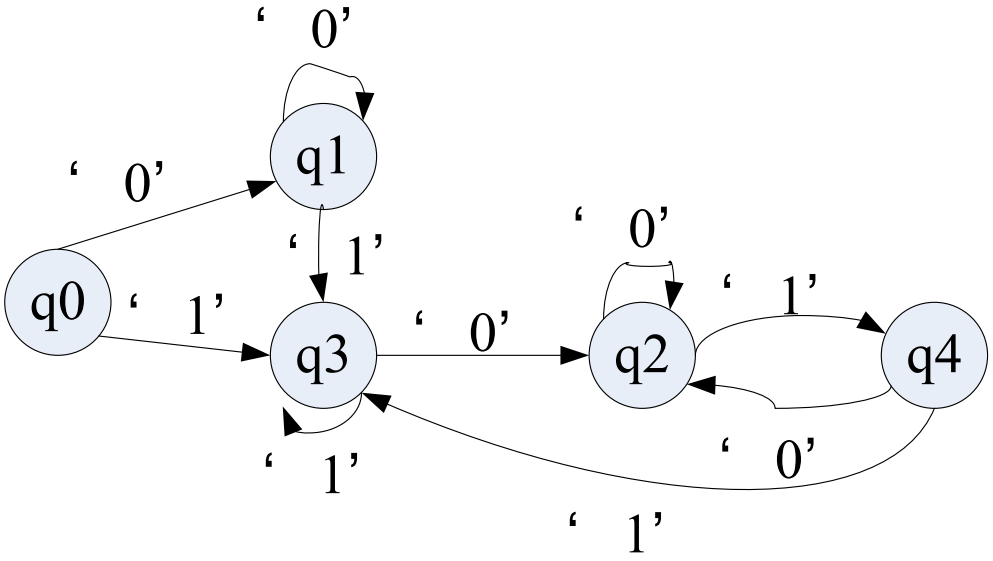
- ‘0’
  - 这不是两个1之间的‘0’：进入状态 q1
  - 这是两个1之间的‘0’ ：进入状态 q2
- ‘1’
  - 这不是需要处理时间间隔的‘1’：进入状态 q3
  - 这是需要处理时间间隔的‘1’：进入状态 q4
- ‘#’
  - 进入结束状态 q5



交通灯观测实例研究-解法2

北京郵電大學 Beijing University of Posts and Telecommunications

3、绘制状态迁移图



说明：在q0~q4的任何一个状态下，如果接收到字符’#’,则进入终止状态q5。此处为了保持图的简洁，没有画出到q5的迁移。

q1：接收到0，且这不是两个1之间的0
q2：接收到0，且这是两个1之间的0
q3：接收到1，且这不是需要处理最长时间间隔的1
q4：接收到1，且这是需要处理时间间隔的1

交通灯观测实例研究–解法2

4、确定状态转移函数

- 当前状态是q0
  - 读入‘1’：vehicles++; state=q3
  - 读入‘0’：seconds++; state=q1
  - 读入‘#’： state=q5
- 当前状态是q1
  - 读入‘1’：vehicles++; state=q3
  - 读入‘0’：seconds++;
  - 读入‘#’： state=q5

交通灯观测实例研究-解法2

4、确定状态转移函数（续）

- 当前状态是q3
  - 读入‘1’：vehicles++;
  - 读入‘0’：seconds++; interval++;state=q2
  - 读入‘#’： state=q5
- 当前状态是q2
  - 读入‘1’：vehicles++;
    if(interval>longest)
    longest=interval;
    interval=0; state=q4 ;
  - 读入‘0’：seconds++; interval++;
  - 读入‘#’： state=q5

交通灯观测实例研究–解法2

北京郵電大學 Beijing University of Posts and Telecommunications

4、确定状态转移函数（续）

– 当前状态是**q4**

- 读入‘1’：**vehicles++; state=q3**
- 读入‘0’：**seconds++; interval++; state=q2**
- 读入‘#’：**state=q5**

源代码：

交通观测灯(自动机)-解法2.c

6.2 程序设计实例研究

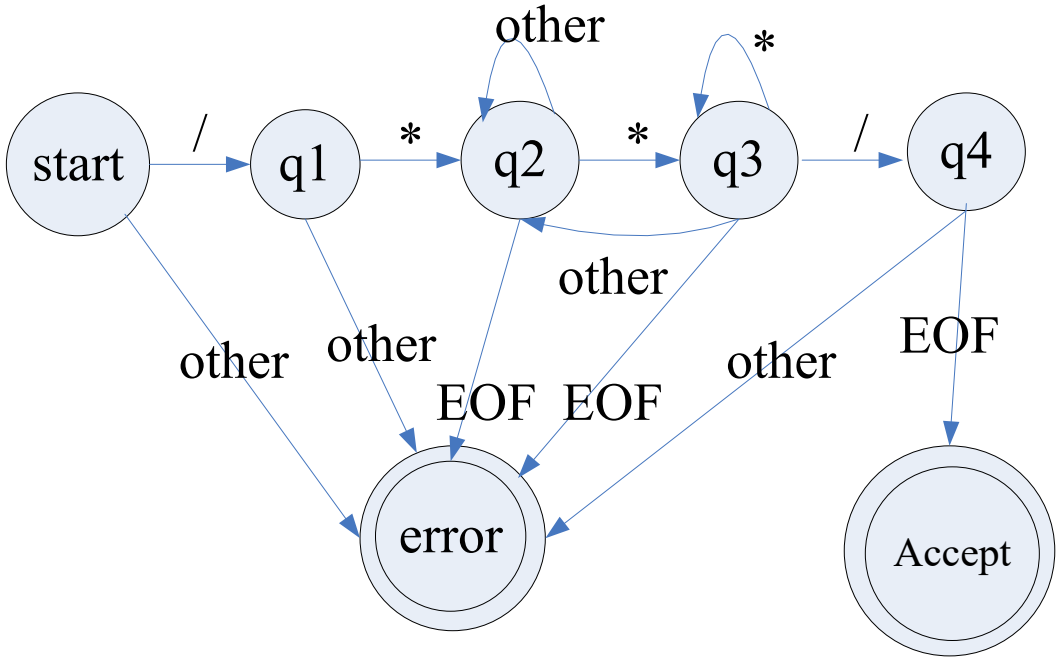
例2 检验输入字符串是否是合法的C语言注释 /*…*/

q1:等待注释开始的*

q2:等待注释结束的*

q3: 等待注释结束的/

q4： 等待EOF状态



绘制状态转换图时，关键是要对每一个状态下的输入字符进行分类，然后针对每一类，看需要迁移到哪个状态，同时做什么操作。

6.2 程序设计实例研究

北京郵電大學 Beijing University of Posts and Telecommunications

- 转换函数分析
  - start状态下：
    - 输入‘/’:state=q1
    - 输出非‘/’:state=ERROR
  - q1状态下：
    - 输入‘*’:state=q2
    - 输出非‘*’:state=ERROR
  - q2状态下：
    - 输入‘*’:state=q3
    - 输入EOF：state=ERROR
    - 输出其他：state=q2

6.2 程序设计实例研究

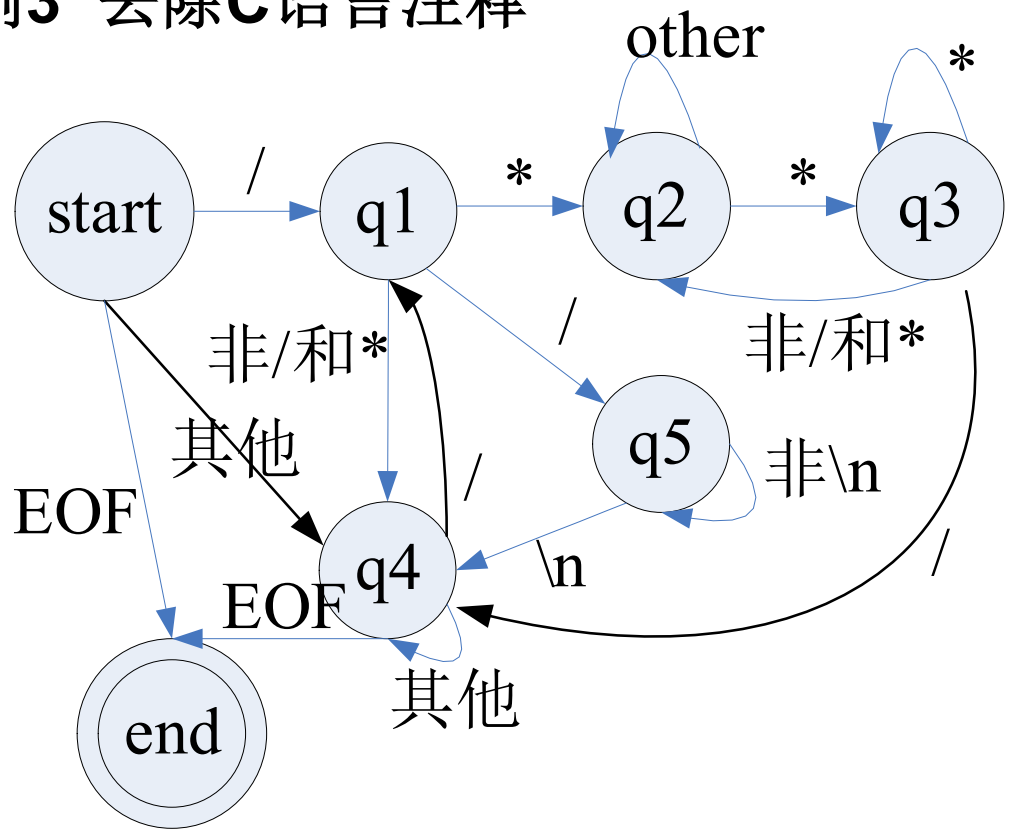
- 转换函数分析(续）
  - q3状态下：
    - 输入‘*’:状态不变
    - 输入‘/’:state=q4
    - 输入EOF：state=ERROR
    - 输出其他：state=q2
  - q4状态下：
    - 输入EOF：state=ACCEPT
    - 输出其他：state=ERROR

源代码

检验C注释风格(自动机).c

6.2 程序设计实例研究

- 例3 去除C语言注释



在q1状态下，如果读取的字符不是*和/,则先要往文件中写入/,再写入读取的字符