



## 第二章 词法分析

DFA——确定有限自动机

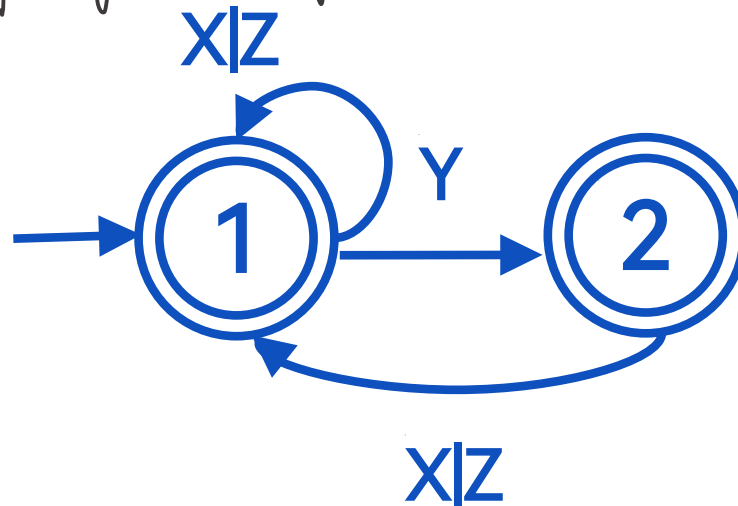
# 上次课留的 作业



设字母表  $\Sigma = \{x, y, z\}$ ,

- 1、包含偶数个  $x$  的所有符号串;
- 2、不包含连续两个  $y$  的所有符号串集合。

$((x|z)^* y (x|z))^* ((x|z)^* y | (x|z)^*)$



## 确定有限自动机 (DFA) 定义



DFA为一个五元组 $(\Sigma, S, S_0, f, Z)$ ，其中：

$\Sigma$ 是一个有穷字母表，它的每个元素称为一个输入字符；

$S$ 是状态的集合，它的每个元素称为一个状态；

$S_0 \in S$ ，是确定有限自动机唯一的一个初始状态；

$f$ 是在  $S \times \Sigma \rightarrow S$  上的转换函数；

$Z \subseteq S$ ，是一个终止状态集，又称为接受状态集；


# 确定有限自动机 (DFA) 定义

## DFA的确定性:

1.初始状态**唯一**

2.状态转换函数  $f: S \times \Sigma \rightarrow S$  是一个**单值函数** 也就是说, 对任何状态  $s \in S$ , 和输入符号  $a \in \Sigma$ ,  $f(s, a)$  **唯一地确定了下一个状态**, 即至多确定一个状态

## 确定有限自动机 (DFA) 定义

例子 

$S=\{0,1,2\}$ ,  $\Sigma=\{a,b\}$ ,

$f(0,a)=1, f(0,b)=2, f(1,a)=2$

$S_0=0$ ,  $Z=\{2\}$

## DFA的表示

### 状态转换矩阵

- ➔ 列标用 自动机的输入字符 表示
- ▶ 行标用 状态 来表示
- ⚙️ 矩阵元素 表示自动机的状态转换函数
- 👉 标识初始状态和终止状态：一般约定，第一行表示开始状态 $S_0$ ，或在右上角标注“+”；右上角标有“\*”或“-”的状态为终止状态；

## DFA的表示

### DFA的表示

- DFA  $M = (\{S, U, V, Q\}, \{a, b\}, f, S, \{Q\})$ , 其中

$f$  定义为:  $S \times \Sigma \rightarrow S$

$f(S, a) = U$

$f(V, a) = U$

$f(S, b) = V$

$f(V, b) = Q$

$f(U, a) = Q$

$f(Q, a) = Q$

$f(U, b) = V$

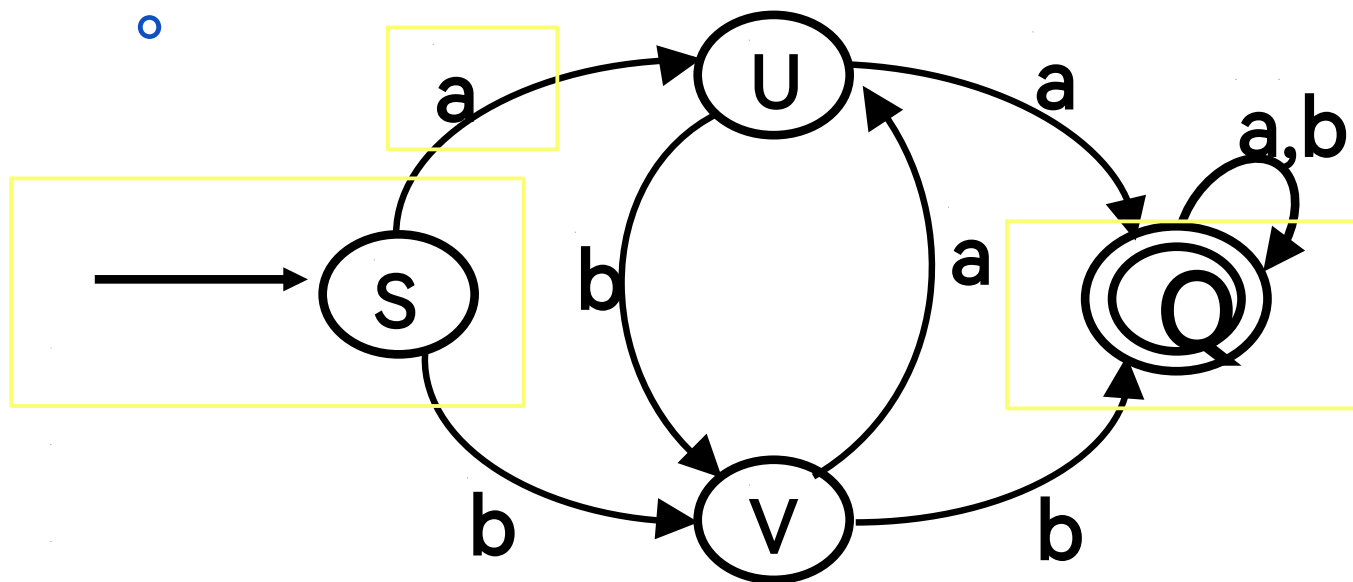
$f(Q, b) = Q$

输入字符 $\Sigma$	a	b
状态 $S$	U	V
U	Q	V
V	U	Q
$Q^*$	Q	Q

## DFA的表示

### 状态转换图

结点表示状态，转换边表示转换函数，  
边的箭头方向指向转换函数中定义的转换方向。标识出初始状态和终止状态。





## DFA接受的集合

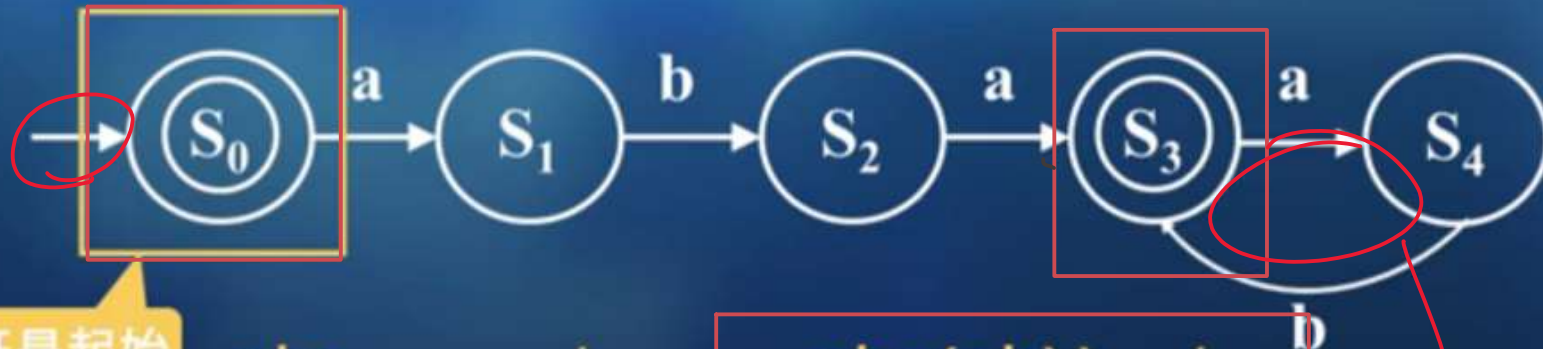


- ✓ 对于 $\Sigma^*$ 中的任何字符串 $t$ , 若存在一条从初始结点到某一终止结点的路径, 且这条路上所有弧的标记符连接成的字符串等于 $t$ , 则称 $t$ 可为DFA  $M$ 所接受 (识别)

只要绕了一圈最后回到了终止状态 (不论中间经过几次终止状态)。  
o o o o

- ✓ DFA  $M$  所能接受的字符串的全体记为 $L(M)$

# 编译原理



既是起始  
又是终止

有→和回

aba ✓

abaa ✗

abaab ✓

$aba(ab)^*$  ✓

$\epsilon$  ✓

回路

即只要以aba开头，  
后面可接 $(ab)^*$

↑ 开头  
↑ 必须到结尾回才行

可以接受一个特殊的字符串

确定有限自动机 (DFA) 的定义

DFA的表示

DFA接受的集合

应用实例

## 应用实例



一道考研题，给定一个字符串，判断其是否合法

0, 1, ②,

1) 这个字符串由0,1组成，由2作为结尾

2) 合法的要求是不能出现两个1串。

如01012就是非法的，0112是合法的

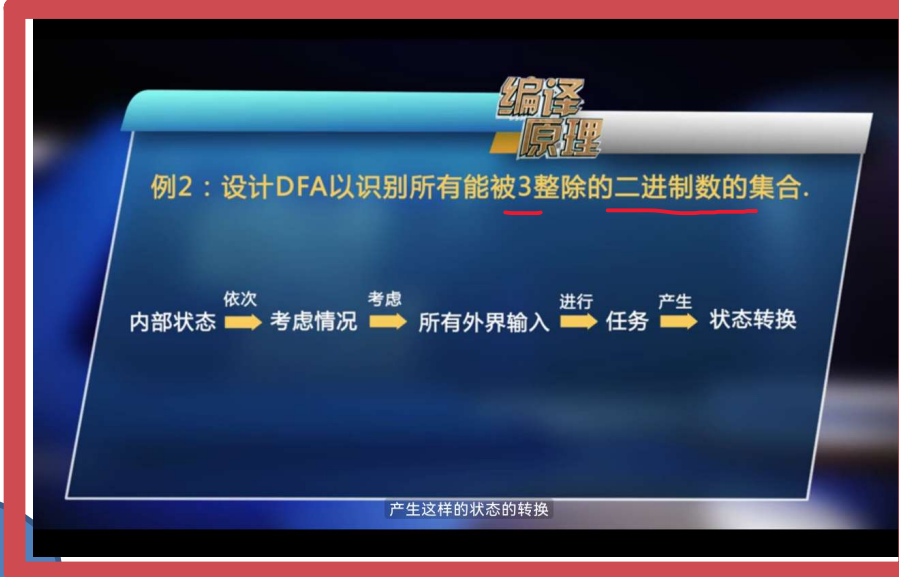
确定有限自动机 (DFA) 的定义

DFA 的表示

DFA 接受的集合

## 应用实例

## 应用实例



确定有限自动机 (DFA) 的定义

DFA 的表示

## 应用实例

被几整除

就几个状态.

一个稍微需要技巧性的例子

用自动机描述  
被3整除的数。

编译原理

例2：设计DFA以识别所有能被3整除的二进制数的集合。

S0：余0状态（能被3整除状态）

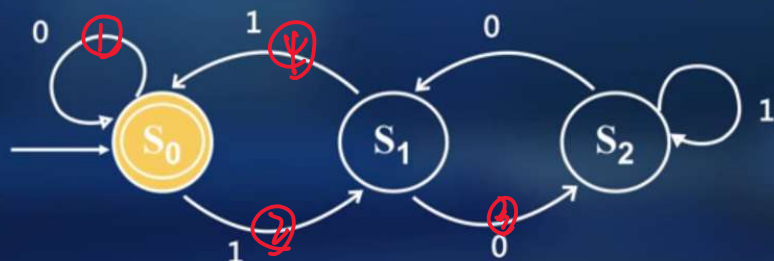
S1：余1状态

S2：余2状态

S0: 00 01

S1: 10 11

S2: 100 101



直到读取完这个二进制数后

- ① 初始状态为余0. 若输入00, 则 ②  
② 若输入01, 则 ③ 余1.  
③ 若输入10, 则 ④ 余2  
④ 若输入11, 则 ① 余0.

确定有限自动机 (DFA) 的定义

DFA的表示

DFA接受的集合

应用实例

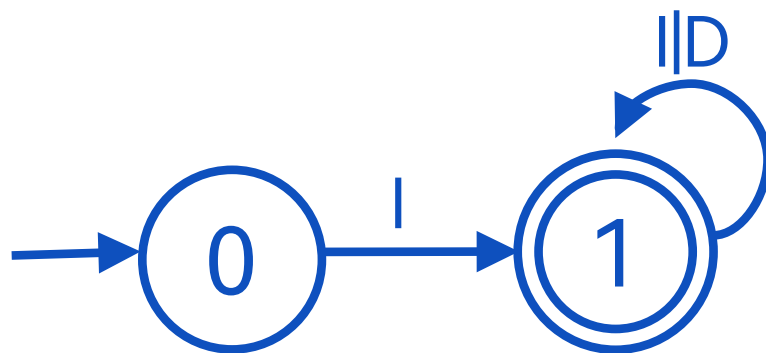
用DFA描述单词

用DFA描述单词

## 用DFA描述单词

### 标识符的描述

我们用 I 表示所有字母，D 表示数字，  
则有：



确定有限自动机 (DFA) 的定义

DFA 的表示

DFA 接受的集合

应用实例

用 DFA 描述单词

用 DFA 描述单词

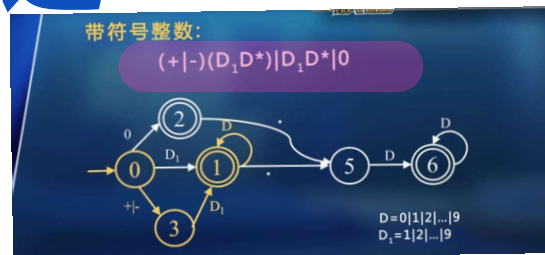
## 用 DFA 描述单词

## 常数的描述

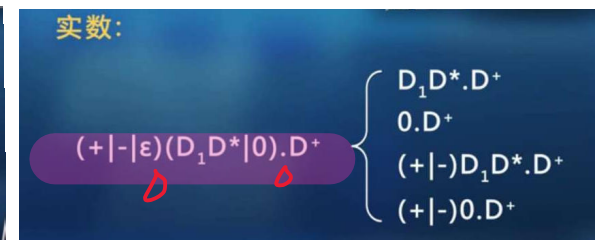
$D1 = 1|2|3|\dots|9$

无符号整数

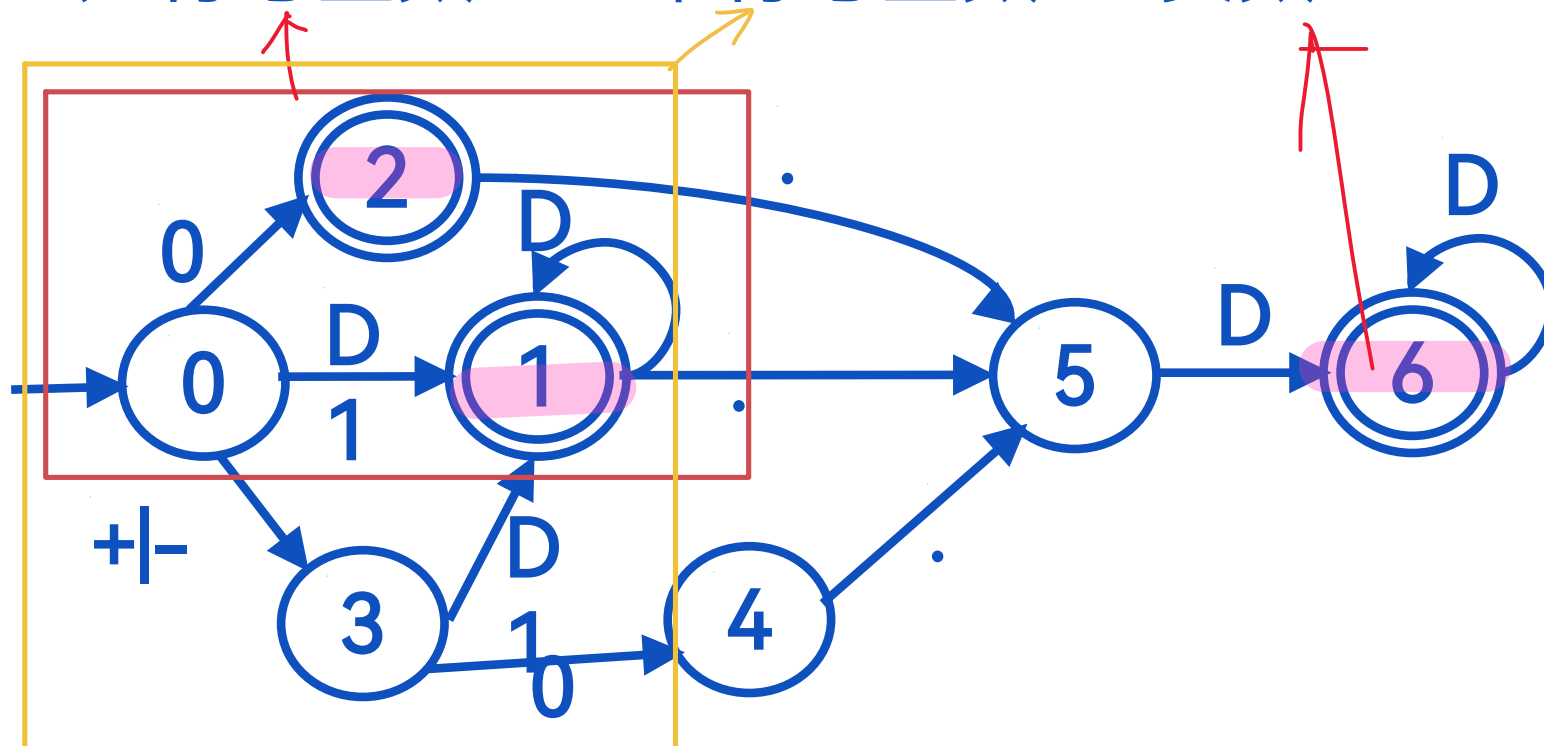
正、负、无符号  
0 或一个数  
 $\underbrace{0, D^+}_{\text{首位非0数}} | 0 \cdot \underbrace{D^+}_{\text{小数}} \cdot D^+$   
D 出现一次或多次



带符号整数



实数 有小数!



确定有限自动机 (DFA) 的定义

DFA 的表示

DFA 接受的集合

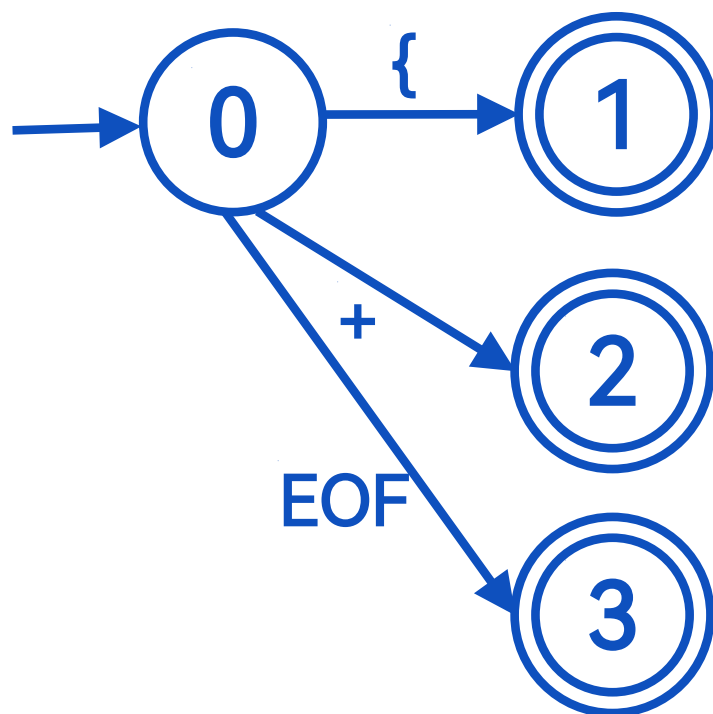
应用实例

用 DFA 描述单词

用 DFA 描述单词

## 用 DFA 描述单词

### 特殊符号





确定有限自动机 (DFA) 的定义

DFA 的表示

DFA 接受的集合

应用实例

用 DFA 描述单词

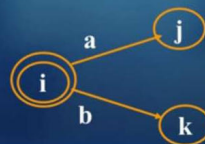
自动机的实现

## 自动机的实现

### 直接转换法

针对状态数不太多的情况，用双重 switch 语句来实现。

方法1：直接转向法



```
Li: switch ( CurChar )
{ case 'a' : goto Lj;
  case 'b' : goto Lk;
  case '#' : Accept;
  default : Error( );
}
```

### 状态转换矩阵

自动机存储在矩阵中，状态比较多，每次都去查表，跳转。

方法2：基于状态转换矩阵

```
state=S0;
curChar=readNextChar();
while(curChar!= '#'
      && T[state,curChar]!=error)
{
    //则当前状态转为新的状态
    state = T[state,curChar];
    curChar=readNextChar() ;
}
if(curChar== '#' && state∈FinalState)
    return(true);
else return(false);
```

$\Sigma$	a	b
S		
0 <sup>+</sup>	1	
1		2
2	3	
3 <sup>*</sup>	3	3

特点：程序内容不变 | 只需改变状态表内容，但状态多时占用存储空间大。

确定有限自动机 (DFA) 的定义

DFA的表示

DFA接受的集合

应用实例

用DFA描述单词

自动机的实现

注意的问题

## 注意的问题



特殊的状态  
“死状态”



缺省  
状态



缺省  
转换函数

确定有限自动机 (DFA) 的定义

DFA的表示

DFA接受的集合

应用实例

用DFA描述单词

自动机的实现

注意的问题

自动机等价

## 自动机等价 $\bar{0}=\bar{0}$

对于两个DFA  $M1$ 和 $M2$

若有 $L(M1)=L(M2)$ ,则称 $M1$ 和 $M2$ 等价

正则集.

## 习题

设计一个DFA

使其能接受被4整除的二进制数