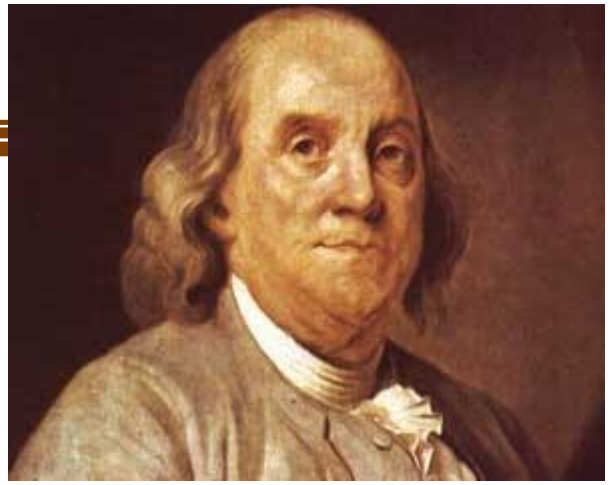
Information Science and Technology College of Northeast Normal University

Benjamin Franklin (1706-1790)



Without continual growth and progress，such words as improvement, achievement , and success have no meaning.

没有持续的成长和进步，诸如成就和成功等词就没有意义。

Information Science and Technology College of
Northeast Normal University

Compiling and Running of Program

Dr. Zheng Xiaojuan
Professor

September. 2019

Information Science and Technology College of Northeast Normal University

Outline

- 2.1 Overview
  - 2.1.1 General Function of a Scanner
  - 2.1.2 Some Issues about Scanning
- 2.2 Finite Automata
  - 2.2.1 Definition and Implementation of DFA
  - 2.2.2 Non-Determinate Finite Automata
  - 2.2.3 Transforming NFA into DFA
  - 2.2.4 Minimizing DFA
- 2.3 Regular Expressions
  - 2.3.1 Definition of Regular Expressions
  - 2.3.2 Regular Definition
  - 2.3.4 From Regular Expression to DFA
- 2.4 Design and Implementation of a Scanner
  - 2.4.1 Developing a Scanner from DFA
  - 2.4.2 A Scanner Generator – Lex

Information Science and Technology College of
Northeast Normal University

2.2 Finite Automata

- **Definition of DFA**
- **Implementation of DFA**
- **Non-Determinate Finite Automata**
- **Transforming NFA into DFA**
- **Minimizing DFA**

Definition of DFA

- formal definition
- two ways of representations
- examples
- some concepts

Formal Definition of DFA

One DFA defines a set of strings;

each string is a sequence of characters in Σ;

Start state gives the start point of generating strings;

Terminal states give the end point;

Transforming function give the rules how to generate strings;

and returns either one unique state or Σ(no definition);

- **Features of a DFA**
  - **One start state;**
  - **For a state and a symbol, it has at most one edge;**

- **Functions of DFA**
  - **It defines a set of strings;**
  - **It can be used for defining lexical structure of a programming language**

Two ways of Representations

- **Table**
  - **Convenient for implementation**
- **Graph**
  - **easy to read and understand**

Two ways of Representations (Table)

- **Transforming Table**
  - **start state : S^0**
  - **terminal state: S***
  - **Row(行): characters**
  - **Column(列)： states**
  - **Cell(单元): states or ⊥**

Information Science and Technology College of
Northeast Normal University

Example of Transforming Table

	a	b	c	d
S^0	S1	⊥	S2	S^*
S1	⊥	S1	⊥	S2
S2	S^*	⊥	⊥	⊥
S^*	⊥	⊥	S^*	⊥

$\sum$: {a, b, c, d}

SS: {S^0, S1, S2, S^*}

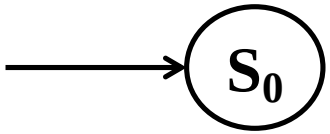
Start state: S^0

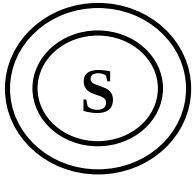
Set of terminal states: {S^*}

Φ: {(S^0,a)→ S1, (S^0,c)→S2,
(S^0,d)→S^*, (S1,b)→S1,
(S1,d)→S2, (S2,a)→S^*,
(S^*, c)→S^*}

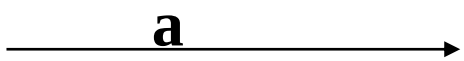
Information Science and Technology College of
Northeast Normal University

Two ways of Representations (graph)

- **Graph**
  - **start state:** 

  - **terminal state:** 

  - **State** 

  - **Edge:** 

Information Science and Technology College of
Northeast Normal University

Example of Graphical DFA

∑: {a, b, c, d}

SS: {S^0, S1, S2, S^*}

Start state: S^0

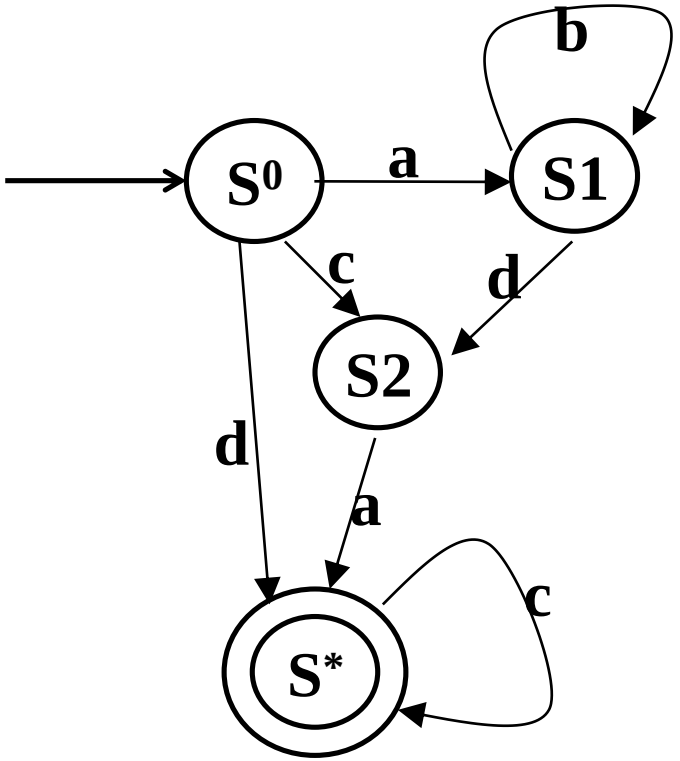
Set of terminal states: {S^*}

Φ: {(S^0,a)→ S1, (S^0,c)→S2,

(S^0,d)→S^*, (S1,b)→S1,

(S1,d)→S2, (S2,a)→S^*,

(S^*, c)→S^*}



Information Science and Technology College of
Northeast Normal University

Some Concepts

- **String acceptable by a DFA**
  - **If *A* is a DFA, $a_1 a_2 \ldots a_n$ is a string, if there exists a sequence of states ($S_0, S_1, \ldots, S_n$), which satisfies**

$$S_0 \xrightarrow{a1} S_1 \ , S_1 \xrightarrow{a2} S_2 \ , \ldots\ldots , \ S_{n-1} \xrightarrow{an} S_n$$

where S_0 is the start symbol, S_n is one of the accept states, the string $a_1 a_2 \ldots a_n$ is acceptable by the DFA *A*.

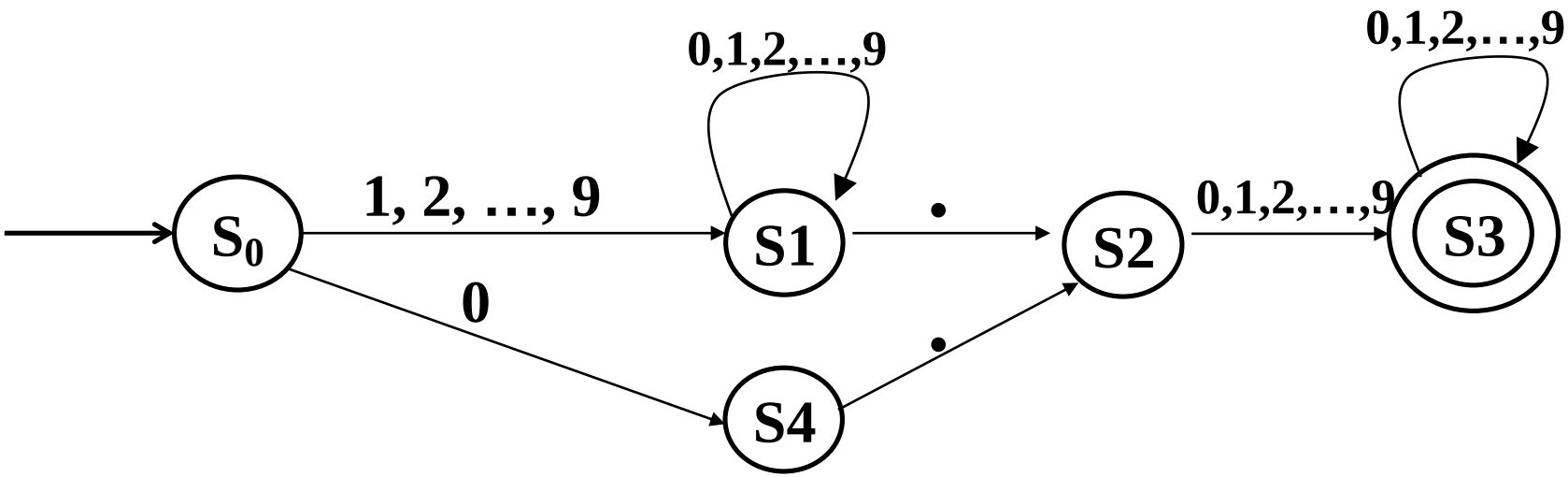
Some Concepts

- **Set of strings defined by DFA**
  - **The set of all the strings that are acceptable by a DFA *A* is called the set of strings defined by *A*, which is denoted as *L(A)***

Information Science and Technology College of Northeast Normal University

Relating DFA to Lexical Structure of a Programming language

- **Use a DFA to define the lexical structure of one word-type in a programming language**
  - **usigned real number (无符号实数)**



Relating DFA to Lexical Structure of a Programming language

- A DFA can be defined for the lexical structure of all the words in a programming language;

- The set of strings defined by the DFA is the set of allowed words in the programming language;

- The implementation of the DFA can be used as a scanner for the programming language;

Summary

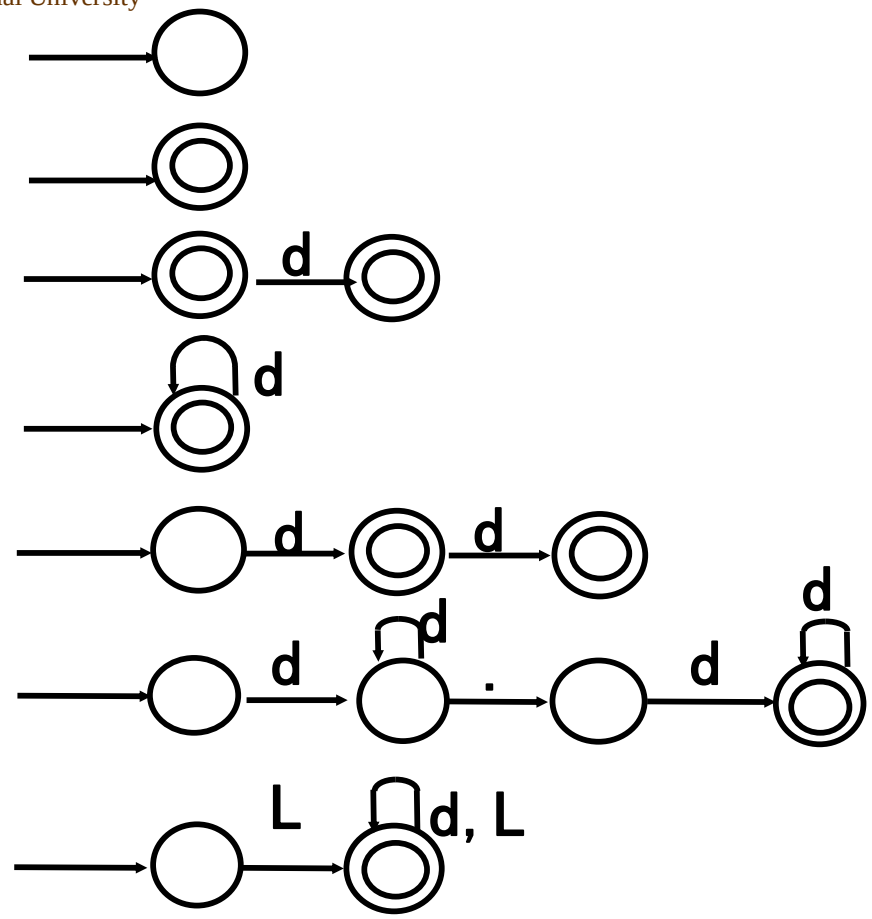
- **Some concepts**
  - **DFA (五元组)**
  - **String acceptable by a DFA**
  - **Set of strings (language) defined by a DFA**
- **Two forms of DFA (table, graphical)**
- **How to define**
  - **Structure of a set of strings;**
  - **Lexical structure of a programming language**

Assignment

- **For standard C programming language**
  - **Find out the token types and their lexical structures;**
  - **Try to write down DFA for each token type;**

Information Science and Technology College of Northeast Normal University

Some special case



{ }

{ε}　ε：什么符号也没有

{ε, d}

{ε, d, dd, ddd, ……}

{d, dd}

real number

identifier

Implementation of DFA

Information Science and Technology College of Northeast Normal University

Implementation of DFA

- **Objective (meaning of implementing a DFA)**
  - **Given a DFA which defines rules for a set of strings**
  - **Develop a program, which**
    - **Read a string**
    - **Check whether this string is accepted by the DFA**
- **If a string is accepted by a DFA,**
  - **Next state;**
  - **Stop in the final state;**
- **If a string is not accepted by a DAF,**
  - **No next state (⊥);**
  - **Not stop in the final state;**

Implementation of DFA

- **Two ways**
  - **Basing on transforming table of DFA**
  - **Basing on graphical representation of DFA**

Information Science and Technology College of Northeast Normal University

Transforming Table based Implementation

- **Main idea**
  - **Input : a string**
  - **Output: *true* if acceptable, otherwise *false***
  - **Data structure**
    - **Transforming table (two dimensional array *T*)**
  - **Two variables**
    - ***CurrentState*: record current state;**
    - ***CurrentChar*: record current character that is read in the string;**

Information Science and Technology College of Northeast Normal University

Transforming Table based Implementation

- **Main idea**
  - **General Algorithm**

1.*CurrentState* = S_0

2. read the first character as *CurrentChar*

3. if *CurrentChar* is not the end of the string,

if *T*(*CurrentState*,*CurrentChar*)≠ error,

***CurrentState* = *T*(*CurrentState*,*CurrentChar*),**

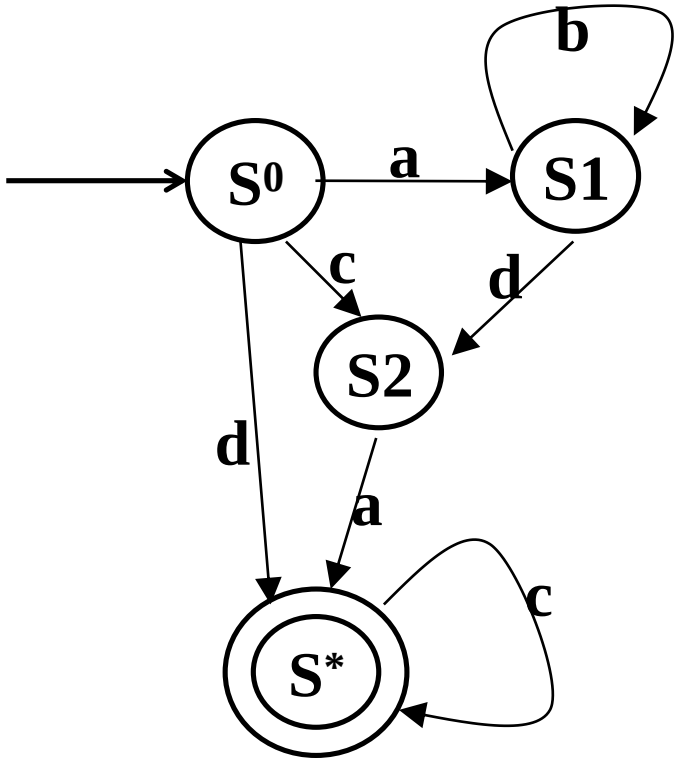
read next character of the string as *CurrentChar*,

goto 3;

4. if *CurrentChar* is the end of the string and *CurrentState* is one of the terminal states, return *true*; otherwise, return *false*.

Example

	a	b	c	d
S^0	S1	⊥	S2	S*
S1	⊥	S1	⊥	S2
S2	S*	⊥	⊥	⊥
S*	⊥	⊥	S*	⊥



1) abdacc ⟶ *true*
2) cab ⟶ *false*

Information Science and Technology College of
Northeast Normal University

Program structure for Table-based Implementation

Transforming table for the DFA

***Variables*: CurrentChar, CurrentState**

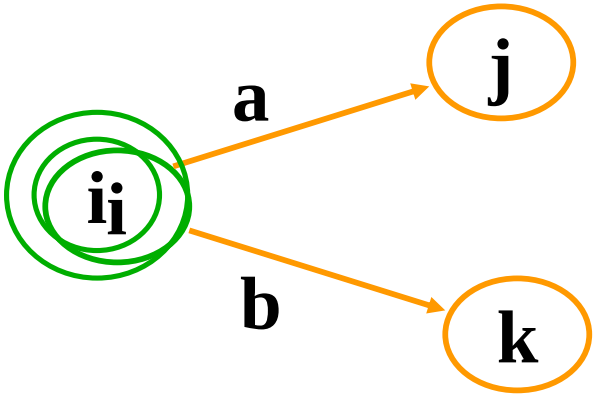
Read the string that want to be checked

***Checking process*:**

Information Science and Technology College of
Northeast Normal University

Graph based Implementation of DFA

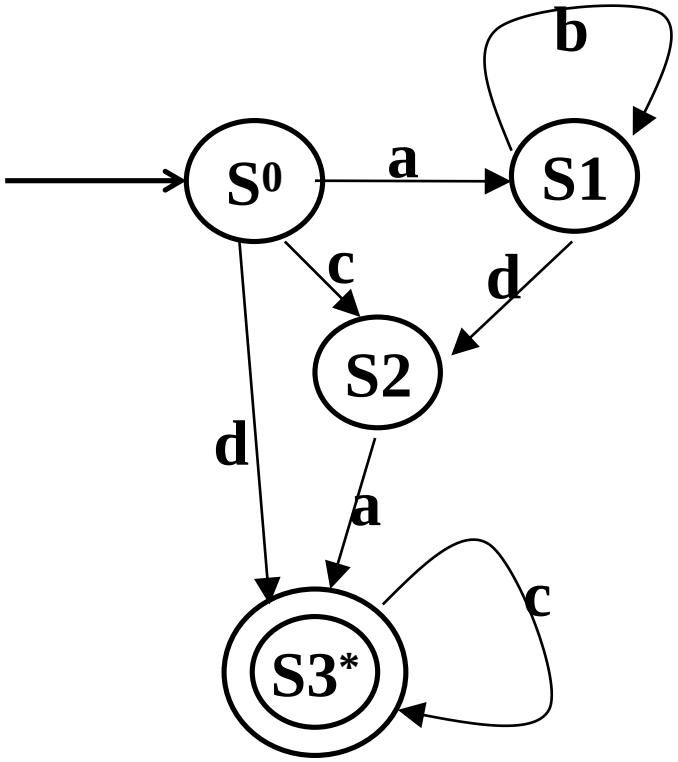
- each state corresponds to a *case* statement
- each edge corresponds to a *goto* statement
- for accept state, add one more branch, if current char is the end of the string then accept;




```
Li: case CurrentChar of
    a     : goto Lj
    b     : goto Lk
    other : Error( )
```

```
Li: case Current
    a     :
    b     :
    #     : r
    other :
```

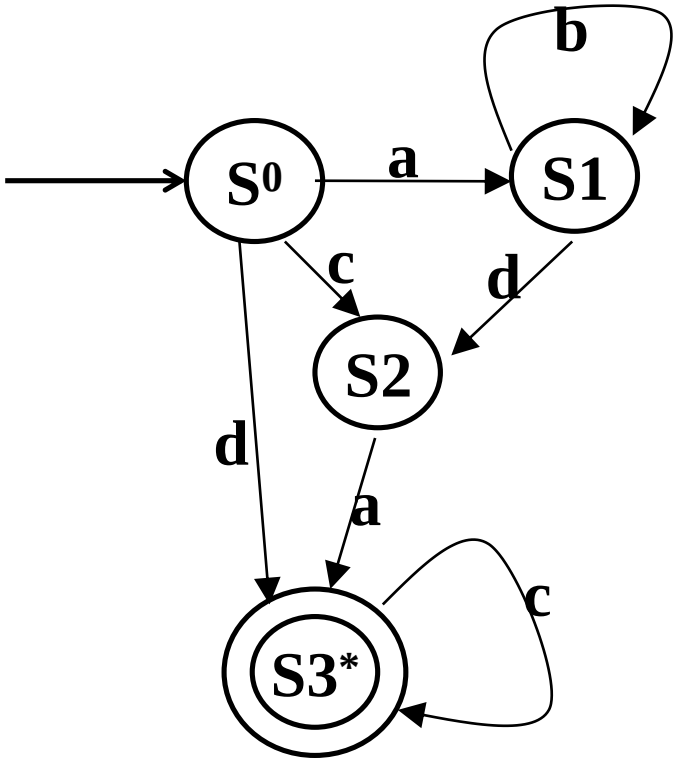
Information Science and Technology College of Northeast Normal University



LS0: read character to ***CurrentChar;***
case CurrentChar ***of***
a: goto LS1;
c: goto LS2:
d: goto LS3;
other: return false;

LS1: read character to ***CurrentChar;***
case CurrentChar ***of***
b: goto LS1;
d: goto LS2:
other: return false;

Information Science and Technology College of Northeast Normal University



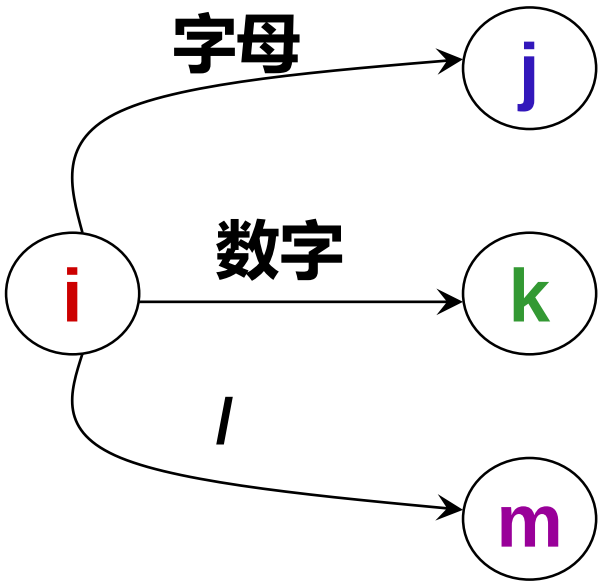
LS2: read character to ***CurrentChar;***
case CurrentChar ***of***
a: goto LS3;
other: return false;

LS3: read character to ***CurrentChar;***
case CurrentChar ***of***
c: goto LS2:
#: return true;
other: return false;

Information Science and Technology College of
Northeast Normal University

Graph based Implementation of DFA

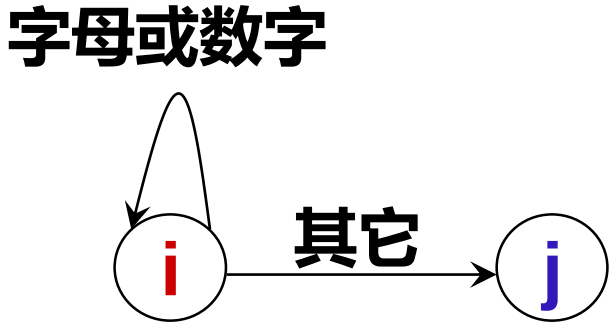
- 不含回路的分叉结点
  - 可用一个CASE语句或一组IF-THEN-ELSE语句实现




```
GetChar( );
if (IsLetter( ))
    {...状态j的对应程序段...;}
else if (IsDigit( ))
    {...状态k的对应程序段...;}
else if (ch='/')
    {...状态m的对应程序段...;}
else
    {...错误处理...;}
```

Graph based Implementation of DFA

- 含回路的状态结点
  - 对应一段由WHILE结构和IF语句构成的程序

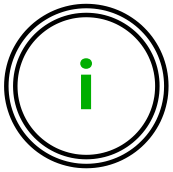



```
GetChar( );
while (IsLetter( )or IsDigit( ))
      GetChar( );
...状态j的对应程序段...
```

Information Science and Technology College of
Northeast Normal University

Graph based Implementation of DFA

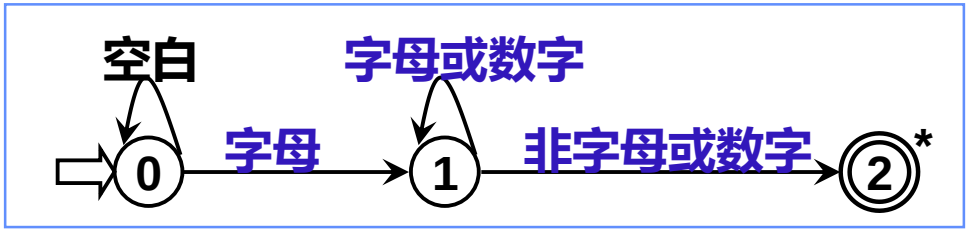
- 终态结点
  - 表示识别出某种单词符号，对应返回语句



RETURN (C，VAL)

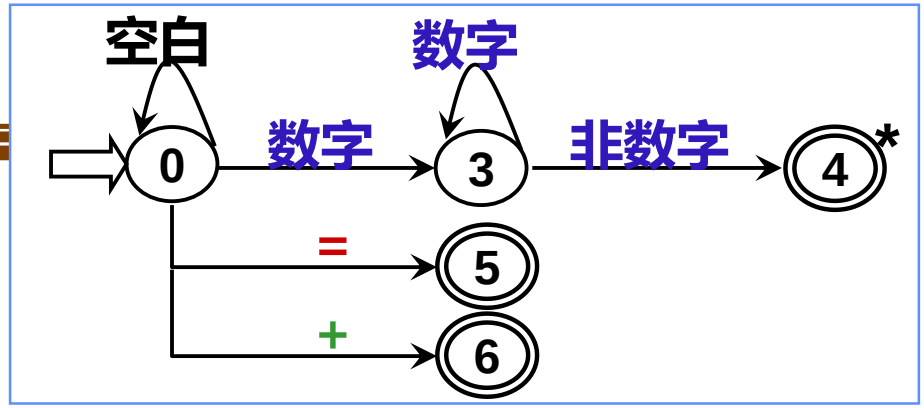
其中，C为单词种别，VAL为单词自身值

Information Science and Technology College of
Northeast Normal University



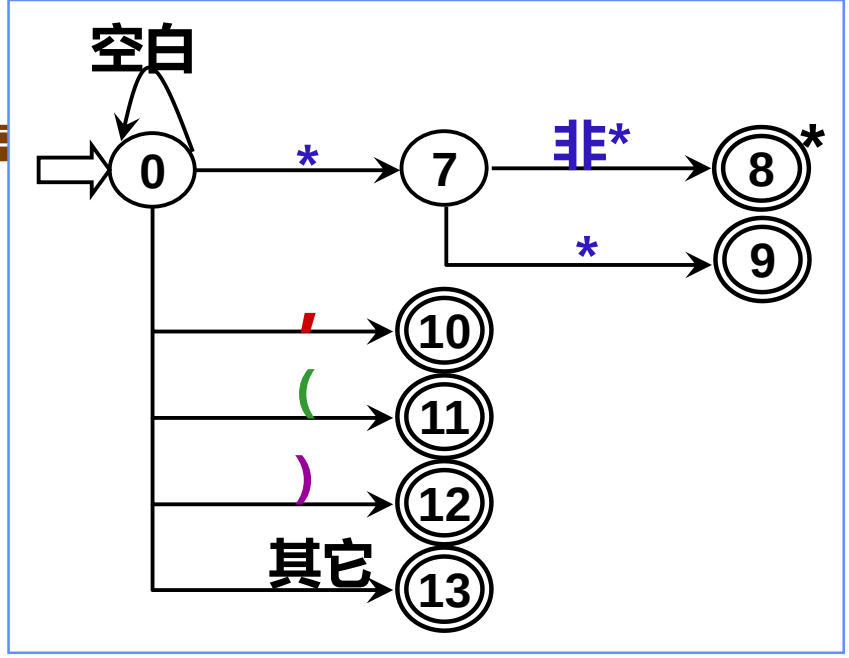

```
int code, value;
strToken := " ";      /*置strToken为空串*/
GetChar(); GetBC();
if (IsLetter())
begin
      while (IsLetter() or IsDigit())
      begin
            Concat(); GetChar();
      end
      Retract();
      code := Reserve();
      if (code = 0)
      begin
            value := InsertId(strToken);
            return ($ID, value);
      end
      else
            return (code, -);
end
```

Information Science and Technology College of Northeast Normal University




```
else if (IsDigit())
begin
      while (IsDigit())
      begin
            Concat( ); GetChar( );
      end
      Retract();
      value := InsertConst(strToken);
      return($INT, value);
end
else if (ch ='=') return ($ASSIGN, -
);
else if (ch ='+') return ($PLUS, -);
```

Information Science and Technology College of
Northeast Normal University

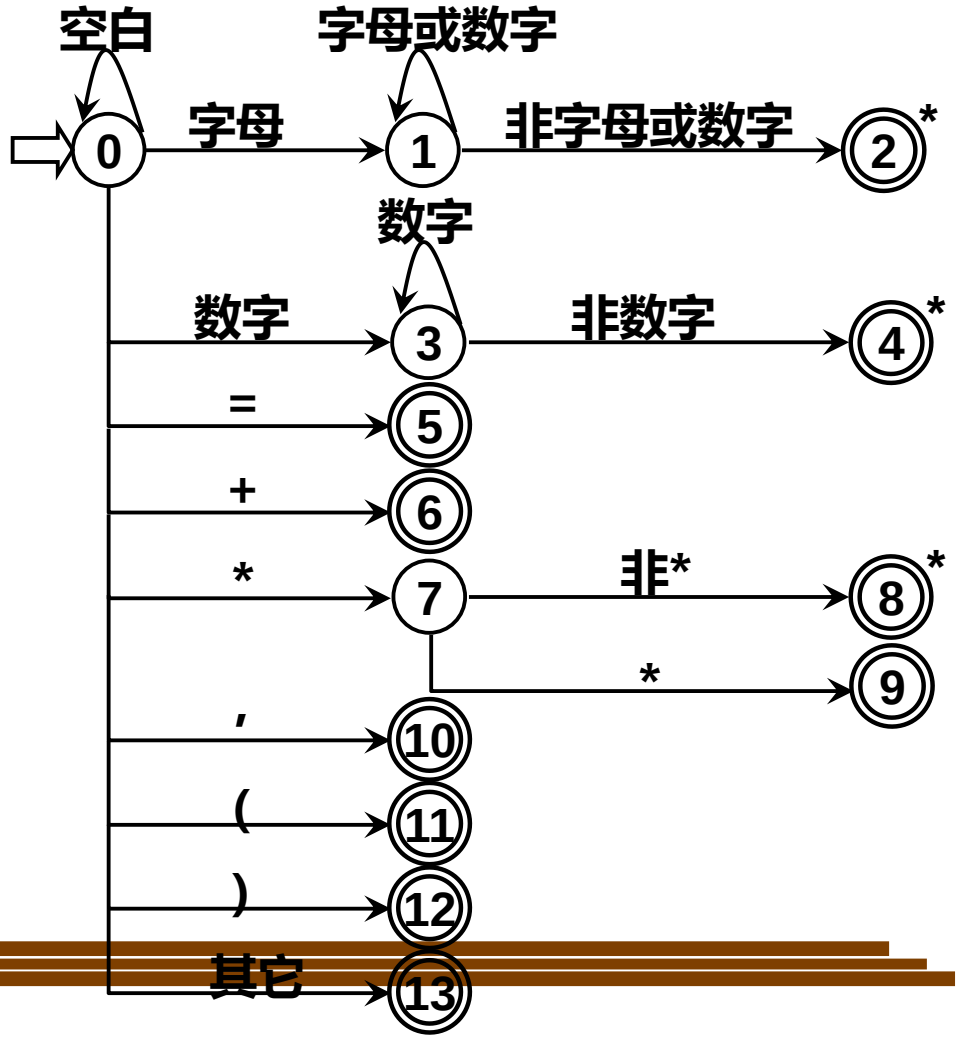



```
else if (ch ='*')
begin
      GetChar();
      if (ch ='*') return ($POWER, -);
      Retract(); return ($STAR, -);
end
else if (ch =',') return ($COMMA, -);
else if (ch ='(') return ($LPAR, -);
else if (ch =')') return ($RPAR, -);
else ProcError( );          /* 错误处理*/
```

Information Science and Technology College of
Northeast Normal University

词法分析器的自动机设计示例

单词	词码	词义
DIM	1	-
IF	2	-
DO	3	-
STOP	4	-
END	5	-
标识符	6	内部字符串
常数(数)	7	标准二进制形式
=	8	-
+	9	-
*	10	-
**	11	-
,	12	-
(	13	-
)	14	-



Application of DFA

Application of DFA for some Problems

- ***Problem 1***: **Develop a program for checking whether a string is a binary number can be divided by 3;**
- ***Problem 2***: **Document requirements for an embedding system;**
- ***Problem 3***: **Formal specification & checking of security policies;**
- ***Problem 4***: **Formal specification of component contract;**

Application of DFA for some Problems

- ***Problem 1***: **Develop a program for checking whether a string is a binary number can be divided by 3;**
- ***Problem Analysis***:
  - **The input string should be composed of either 0 or 1;**
  - **The binary number represented by the input string should be divided by 3;**

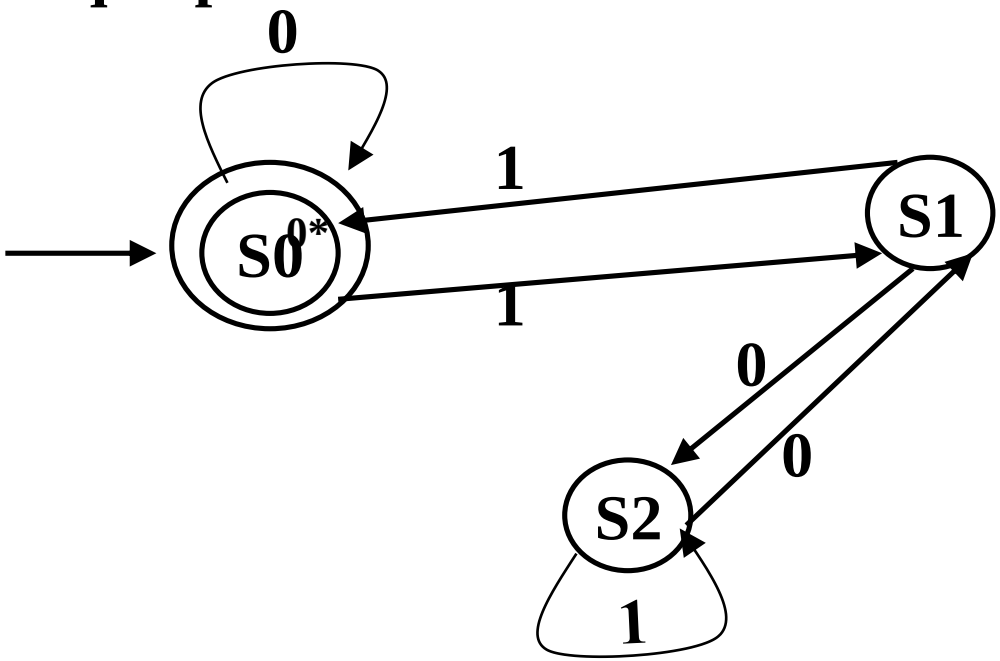
With DFA we can solve it!

Information Science and Technology College of Northeast Normal University

设二进制数i，后面跟一个0，产生符号串2i，后面跟一个1，产生符号串2i+1
余数： i/3=q， 2i/3=2q

	2i	2i+1
q=0	0	1
q=1	2	0
q=2	1	2

- ∑ = {0,1}
- SS = { S0, S1, S2},
- Sq represents the state that the remainder(余数) is q; (q=0,1,2)



Start State: S0
TS = {S0}

Implementation

Assignment

- **Define a DFA for accepting a set of binary numbers, each binary number can be divided by 5;**

  (定义一个DFA，它所接受的符号串为能被5整除的二进制数;)

- **Implementation of the DFA above;**

Application of DFA for some Problems

- ***Problem 2***: **Document requirements for an embedding system;**

- **The features of embedding system**
  - **Event driven**
  - **Changing in different states**
  - **Trigger corresponding devices to do the task;**

Information Science and Technology College of Northeast Normal University

Application of DFA for some Problems

- ***Problem 2***: **Document requirements for embedding systems;**

- **∑ = set of events**
- **SS = all states that the system has**
- **S0 = initial state**
- **TS = all the exit states**
- **Φ(Si, event) = Sj**

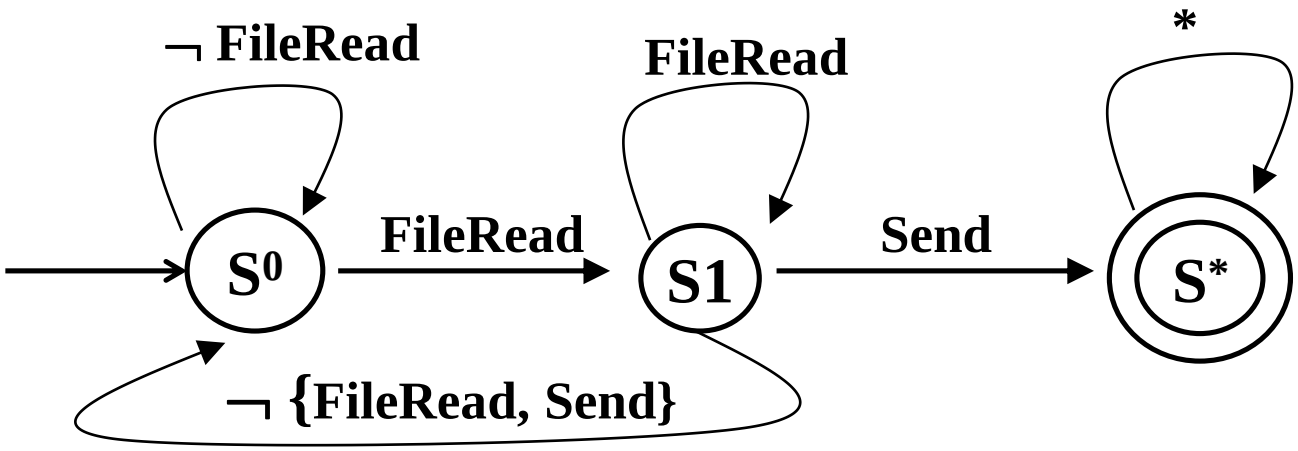
Application of DFA for some Problems

- ***Problem 3***: **formal specification & checking of security policies;**

- **Security policies**
  - **Formal specification with DFA;**
  - **Checking by implementing the DFA**

A simple security policy

- **Basing on system calls**



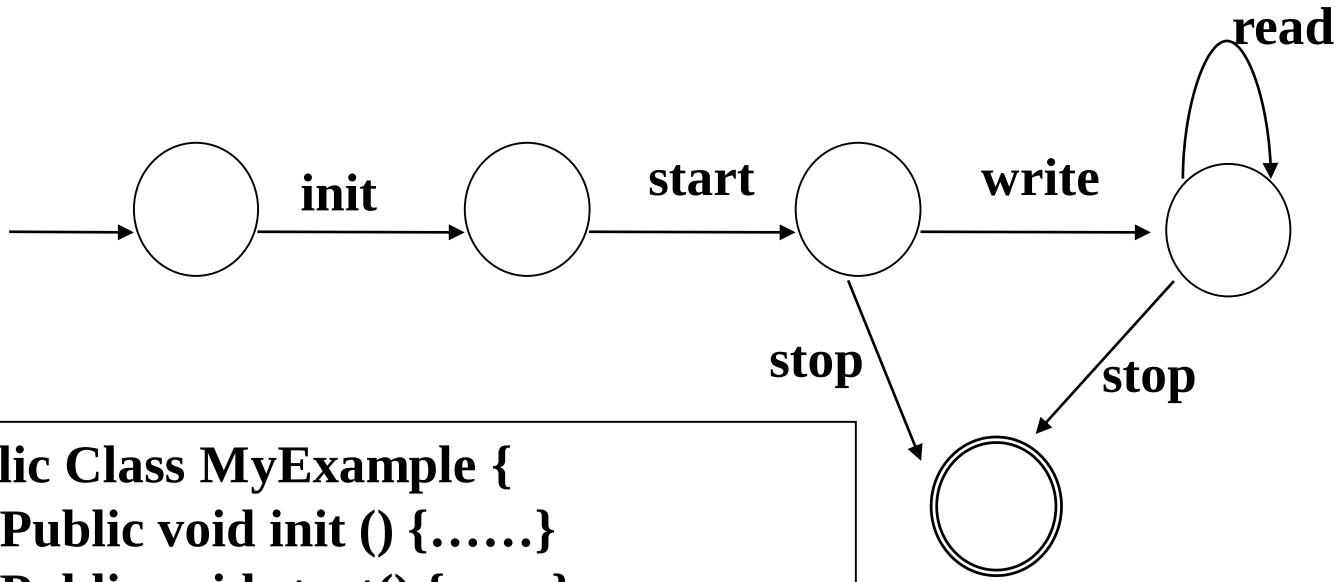
A security policy that prohibits execution of *send* after a *FileRead* has been executed.

Application of DFA for some Problems

- ***Problem 4***: **formal specification of component contract**
- **C. Szyperski, "Component Software – beyond Object-Oriented Programming", Addison-Wesley, 2002**
- **The definition of software components should come with the notion of contract in order to organize the guarantee of properties all along the software life cycle.**

Information Science and Technology College of
Northeast Normal University

A simple Component Contract




```
Public Class MyExample {
      Public void init () {……}
      Public void start() {……}
      Public int read() {……}
      Public void write(int  n) {……}
      Public void stop() {……}
}
```