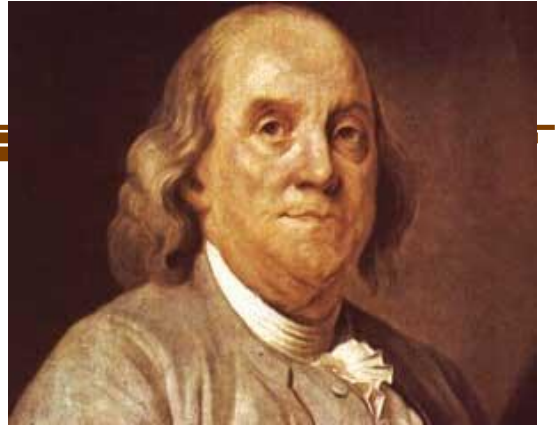
Information Science and Technology College of Northeast Normal University

Franklin's 13 virtues and Puritan ethics



(1) Temperance: Eat not to dullness; drink not to elevation.

节制：食不过饱，饮不过量

This virtue is consistent with Puritan code of drinking in moderation so as to keep a sober mind.

(2) Silence: Speak not but what may benefit others or yourself; avoid trifling conversation.

缄默：言则于人于己有益，不做鸡毛蒜皮的闲扯。

Great minds discuss idea. Average minds discuss events. Small minds discuss people.

智者论道、能者谈事、庸者诽人

Information Science and Technology College of
Northeast Normal University

Compiling and Running of Program

Dr. Zheng Xiaojuan
Professor

October. 2019

Information Science and Technology College of
Northeast Normal University

What were introduced in Last Lecture

- **Formal Definition of NFA**
- **Differences between NFA & DFA**
- **From NFA to DFA**
- **Minimizing DFA**

Information Science and Technology College of
Northeast Normal University

Summary of Homework

- **(1) Define data structure for Token**
  - **Be familiar with Data Structure**
  - **Be familiar with what a token is composed of;**

```
enum TkType {ID, Num,
                    if, else, while, int, real, ……
                     colon, comma, semi, ……}
```

```
Struct Token {TkType  type;
                    char sema[40];}
```

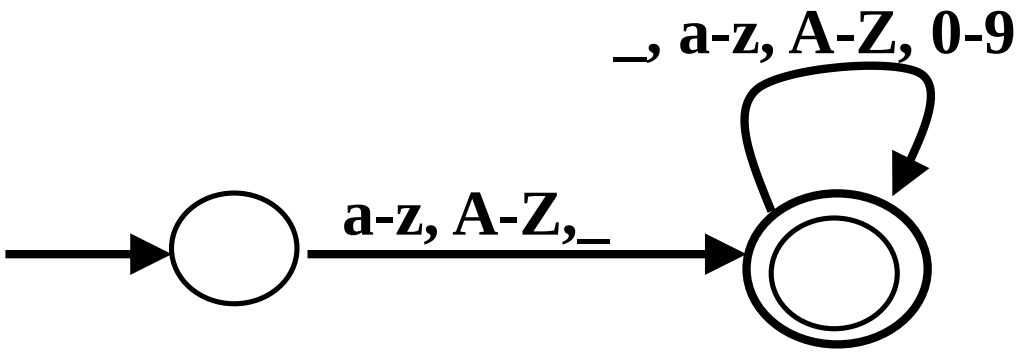
Information Science and Technology College of
Northeast Normal University

Summary of Homework

- **(2) Find out token types of C programming language, and give their DFA definition.**
  - **Token Type: identifier, keywords, constant, special symbols;**
  - **How to deal with keyword?**
    - **Keywords are part of identifier --- do not need to define a separate DFA for keywords;**
    - **Establish keyword table;**
    - **Whenever an identifier is recognized, search for the keyword table at first to decide whether the identifier is a keyword;**
  - **DFA definition for each token type**
    - **Know how is a token of the token type is structured;**
    - **Lexical rules**

Information Science and Technology College of
Northeast Normal University

DFA definition for identifier



{ _, a, b, …, z, A, …, Z, _ _,
_a, …, _z, …, _A, …, _Z,_0, …, _9,
…….}

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

2.3 Regular Expressions

- **Definition of Regular Expressions**
- **Regular Definition**
- **From Regular Expression to DFA**

Definition of Regular Expressions (RE)

- Some Concepts
- Formal Definition of RE
- Example
- Properties of RE
- Extensions to RE
- Limitations of RE
- Using RE to define Lexical Structure

Some Concepts

- alphabet(字母表)：a non-empty finite set of symbols，which is denoted as ∑，one of its elements is called symbol.
- string(符号串)：finite sequence of symbols, we use λ or ε to represent empty string(空串);
- 空串集{λ} is different from empty set ∅ { } 。
- length of a string(符号串长度)：the number of symbols in a string, we use |β| to represent the length of the string β;
- concatenate operator for strings(符号串连接操作)：

  if α and β are strings，we use αβ as the concatenation of two strings, especially we have λβ = βλ = β ;

测试：ε是什么？

A. 字符

B. 符号串 √

C. 正规式 √

测试：∅是什么？

A. 集合 √

B. 字

C. 正规式 √

Information Science and Technology College of Northeast Normal University

Some Operators on Set of Strings

- product of set of strings（符号串集的乘积）:
  if A and B are two sets of strings, AB is called the product of two sets of strings, $AB=\{\alpha\beta \mid \alpha \in A, \beta \in B\}$
  especially $\varnothing A=A\varnothing=A$, where $\varnothing$ represents empty set。
- power of set of strings(符号串集合的方幂):
  if A is a set of strings, A^i is called ith power of A, where i is a non-negative integer(非负整数)。
  $A^0=\{\lambda\}$
  $A^1 = A$, $A^2 = AA$
  $A^K = AA.....A$ (k)
- positive closure(符号串集合的正闭包): $A^+ = A^1 \cup A^2 \cup A^3 \ldots\ldots$
- star closure(符号串集合的星闭包): $A^* = A^0 \cup A^1 \cup A^2 \cup A^3 \ldots\ldots$

{a,ab} {c,d,cd} ={ac,ad,acd,abc,abd,abcd}

{a,ab}+ = {a,ab}∪{a,ab}{a,ab}∪.....

= {a,ab,aa,aab,aba,abab,……}

{a,ab}* ={λ}∪{a,ab}∪{a,ab}{a,ab}∪.....

= {λ,a,ab,aa,aab,aba,abab,……}

Formal Definition

- For a given alphabet $\sum$, a regular expression for $\sum$ defines a set of strings of $\sum$,
- If we use R_Σ to represent a regular expression for $\sum$, and $L(R_\Sigma)$ to represent the set of strings that R_Σ defines.

Formal Definition

- ∅ is a regular expression，L(∅)={ }
- λ is a regular expression，L(λ)={ λ }
- for any c ∈Σ, c is a regular expression, L(c)={c}
- if A and B are regular expressions, following operators can be used
  - (A) , L((A))= L(A)
  - choice among alternatives A | B，L(A | B)=L(A)∪L(B)
  - concatenation A B , L(A B)= L(A)L(B)
  - repetation A* , L(A*)= L(A)*

Example

- ∑={ a,b }

RE	Set of strings
1. ab*	1. {a, ab, abb, abbb, ……}
2. a(a|b)*	2. {a, aa, ab, aaa, aab, aba, abb, ……}

Comparing with DFA

- **Equivalent in describing the set of strings;**
- **Can be conversed into each other;**
- **DFA is convenient for implementation;**
- **RE is convenient for defining and understanding;**
- **Both of them can be used to define the lexical structure of programming languages;**

Properties

- A | B = B | A　　| 的可交换性
- A | (B | C) =(A | B) C　　| 的可结合性
- A (B C) =(A B)C　　连接的可结合性
- A (B | C) =A B | A C　　连接的可分配性
- (A | B) C =A C | B C　　连接的可分配性
- A** =A*　　幂的等价性
- A=λA=Aλ　　λ是连接的恒等元素

Extensions

- **Some extensions can be made to facilitate definition**
  - **A^+**
  - **any symbol: “.”**
  - **range: [0-9] [a-z] [A-Z]**
  - **not in the range: ~(a|b|c)**
  - **optional: r?=(λ|r)**

Limitations

- **RE can not define such structure like**
  - **Pairing 配对，（）**
  - **Nesting嵌套，**
- **RE can not describe those structures that include finite number of repetitions**

  for example：w c w , w is a string containing a and b;

(a|b)* c (a|b)* can not be used, because it cannot guarantee that the strings on both sides of c are the same all the time;

Regular Definition

Definition

- **It is inconvenient to define set of long strings with RE, so another formal notation is introduced, which is called "*Formal Definition*";**
- **The main idea is that naming some sub-expressions in RE;**
- **Example:**

(1|2|...|9) (0|1|2|...9)*

NZ_digit= 1|2|...|9

digit = *NZ_digit* | 0

*NZ_digit digit**

Information Science and Technology College of Northeast Normal University

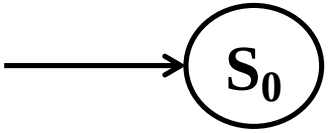
Defining Lexical Structure of *C0*

- letter = a|…|z|A|…|Z
- digit = 0|…|9
- NZ-digit = 1|…|9
- Reserved words:
  Reserved = {| }| read| write
- Identifiers: =letter(letter|digit)*
- Constant:
  integer: int = NZ-digit digit* | 0
- Other symbols: syms = +|*| := | ;
- Lexical structure:
  lex = Reserved | identifier |int | syms

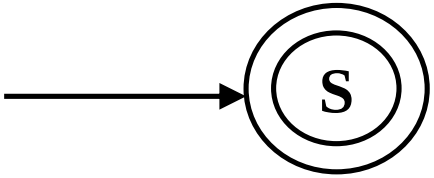
From RE to NFA

Rules

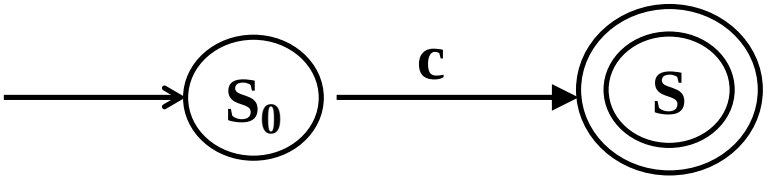
- ∅ is a regular expression，L(∅)={ }



- λ is a regular expression，L(λ)={ λ }

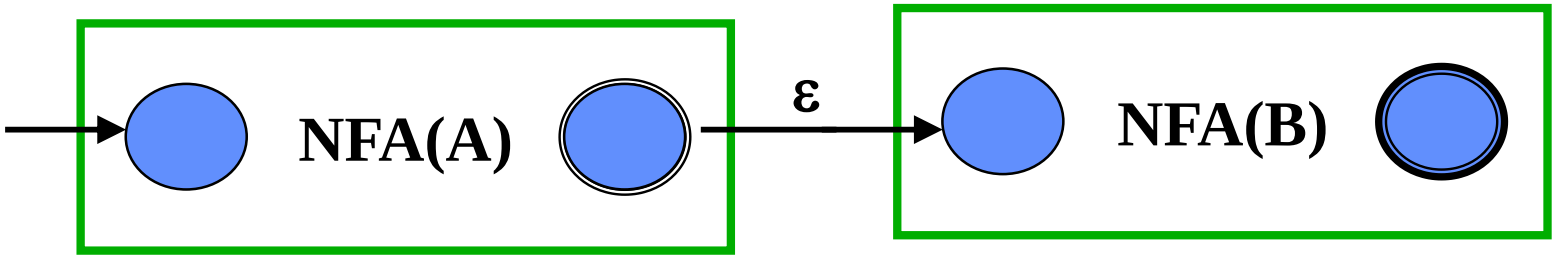


- for any c ∈Σ, c is a regular expression, L(c)={c}



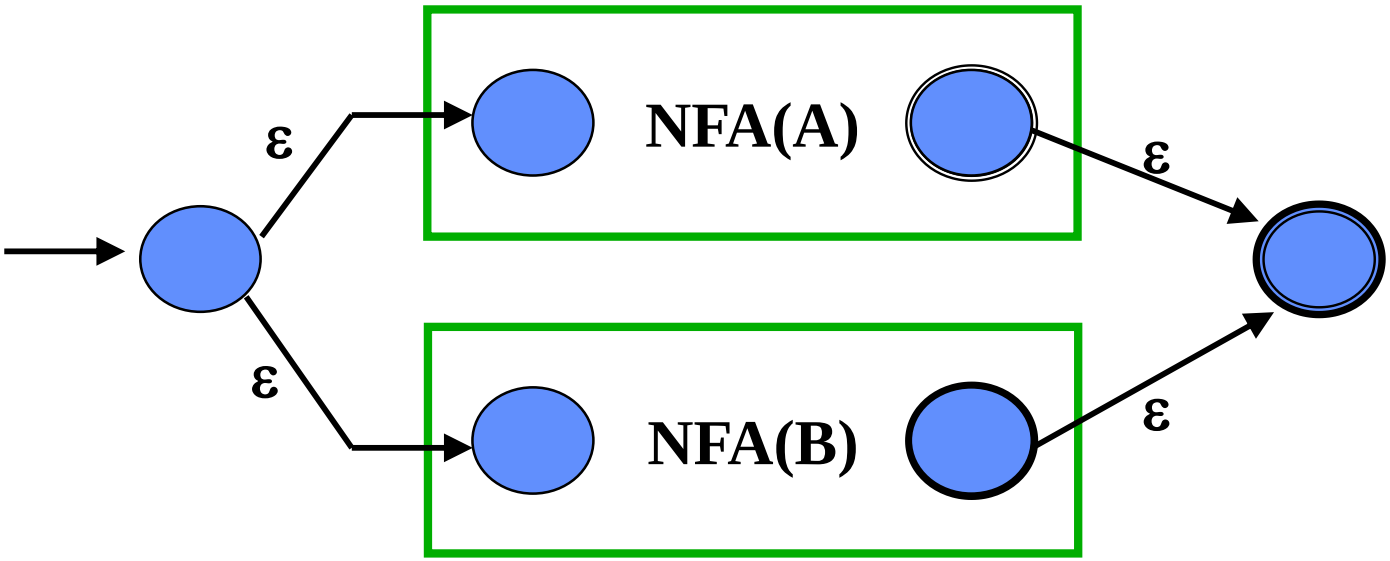
Rules

- (A), L((A)) = L(A), no change;
- A B, L(A B)= L(A)L(B)



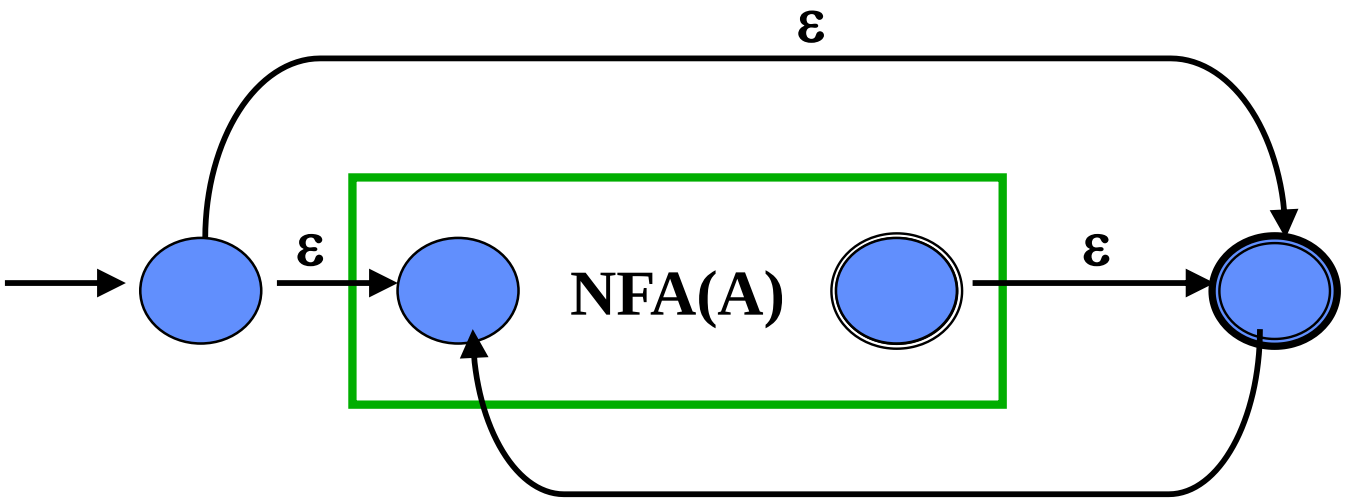
Rules

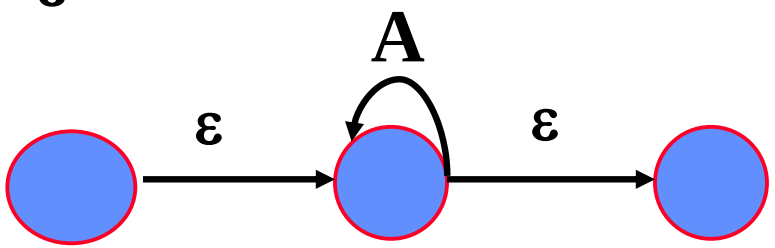
■ A | B，L(A | B)=L(A)∪L(B)



Rules

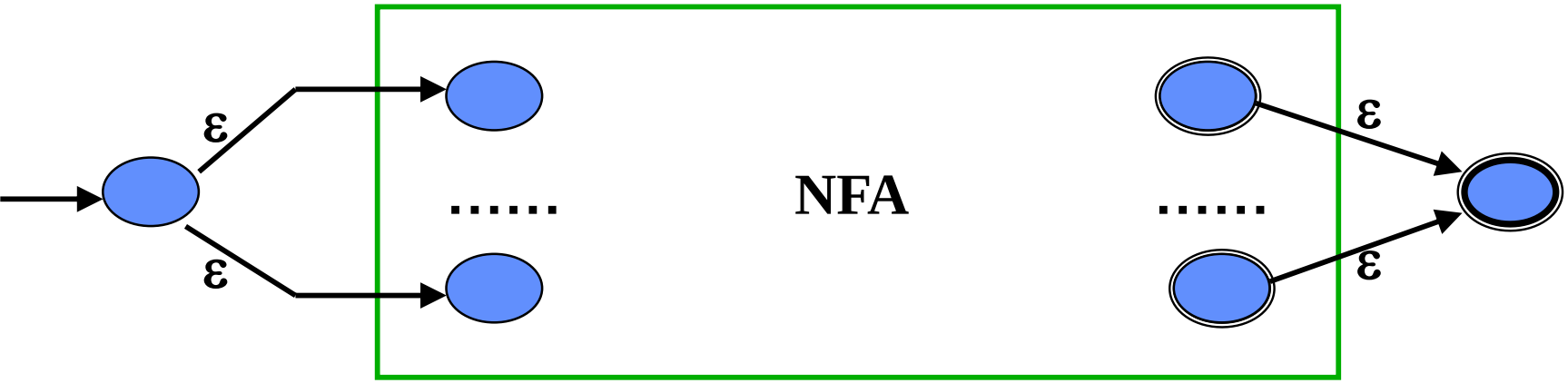
■ A* , L(A*) = L(A)*





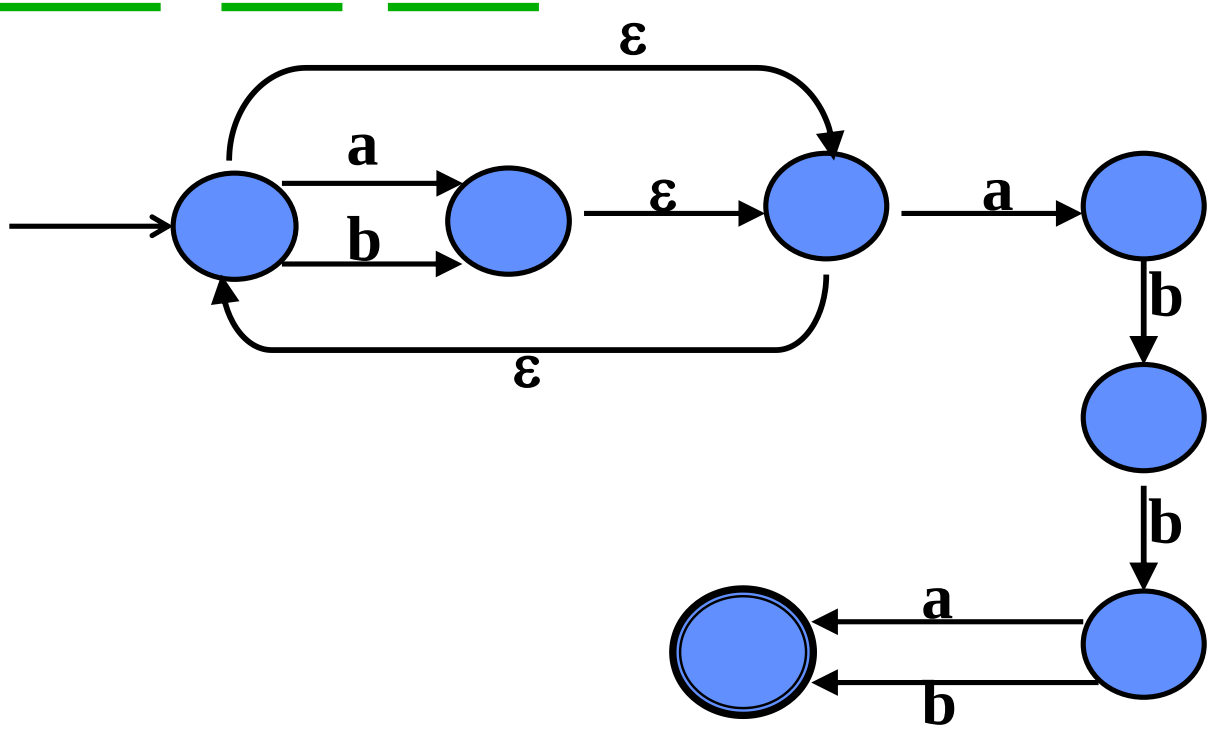
Attention

- **The rules introduced above are effective for those NFAs that have <u>one start state</u> and <u>one terminal state</u>;**
- **Any NFA can be extended to meet this requirement;**

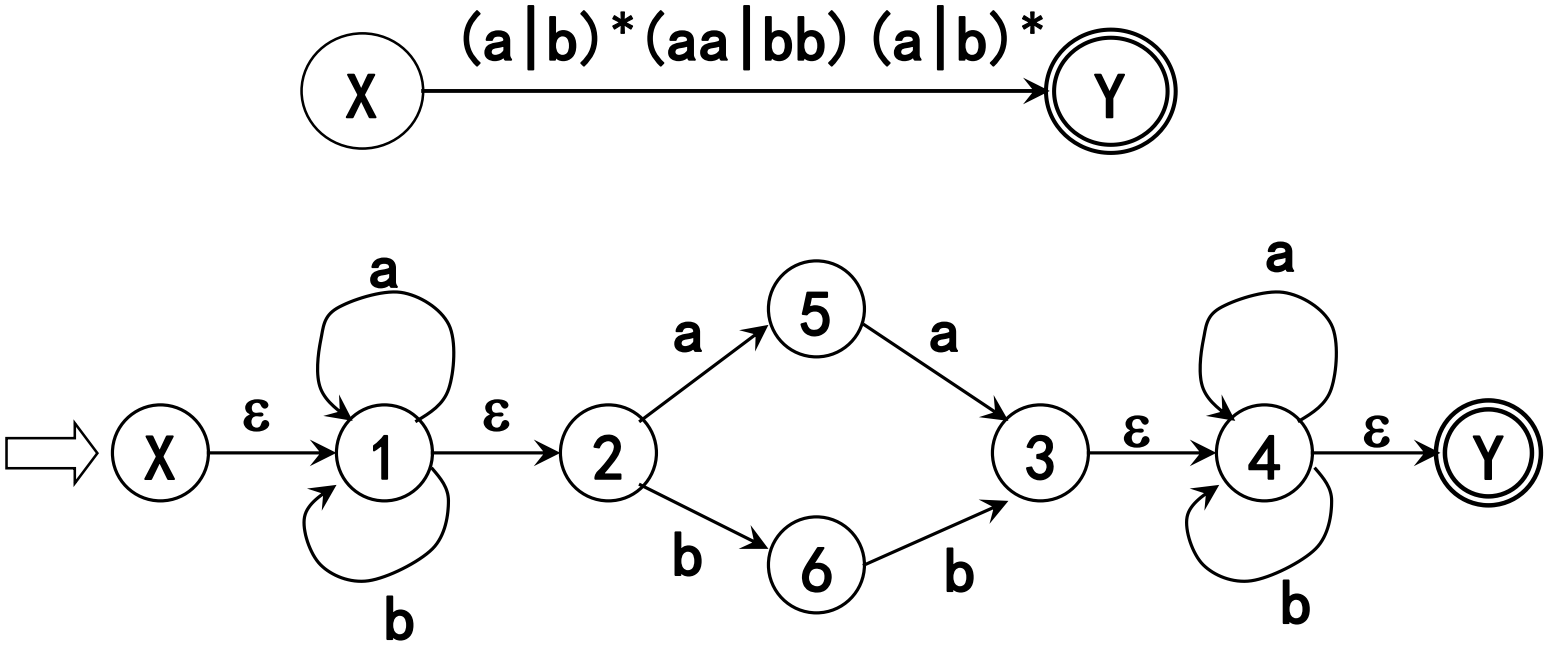


Example

- **(a | b)* a b b (a | b)**



- (a|b)*(aa|bb) (a|b)*



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

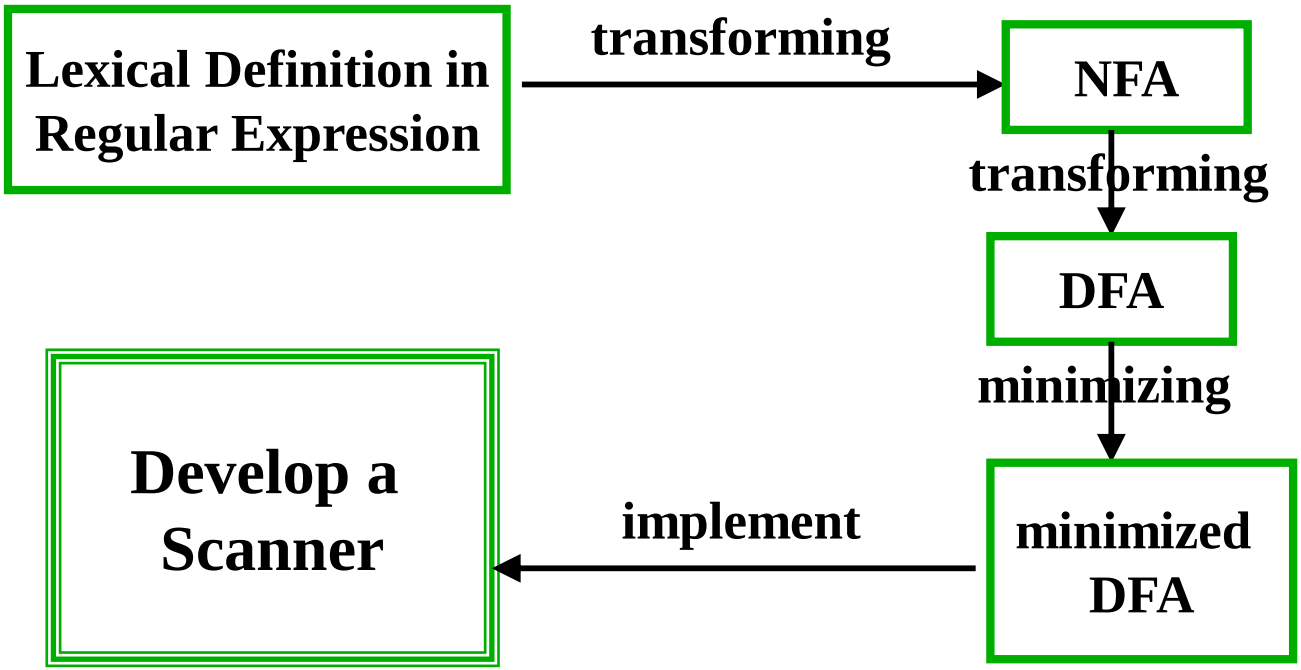
Information Science and Technology College of
Northeast Normal University

2.4 Design and Implementation of a Scanner

- **Developing a Scanner Manually**
- **A Scanner Generator – Lex**

Information Science and Technology College of
Northeast Normal University

Developing a Scanner Manually



Implement Scanner with DFA

- **Implementation of DFA**
  - **Just checking whether a string is acceptable by the DFA;**
- **Implementation of a Scanner**
  - **not checking;**
  - **but recognizing an acceptable string(word) and establish its internal representation**
    - **<token-type, semantic information>**

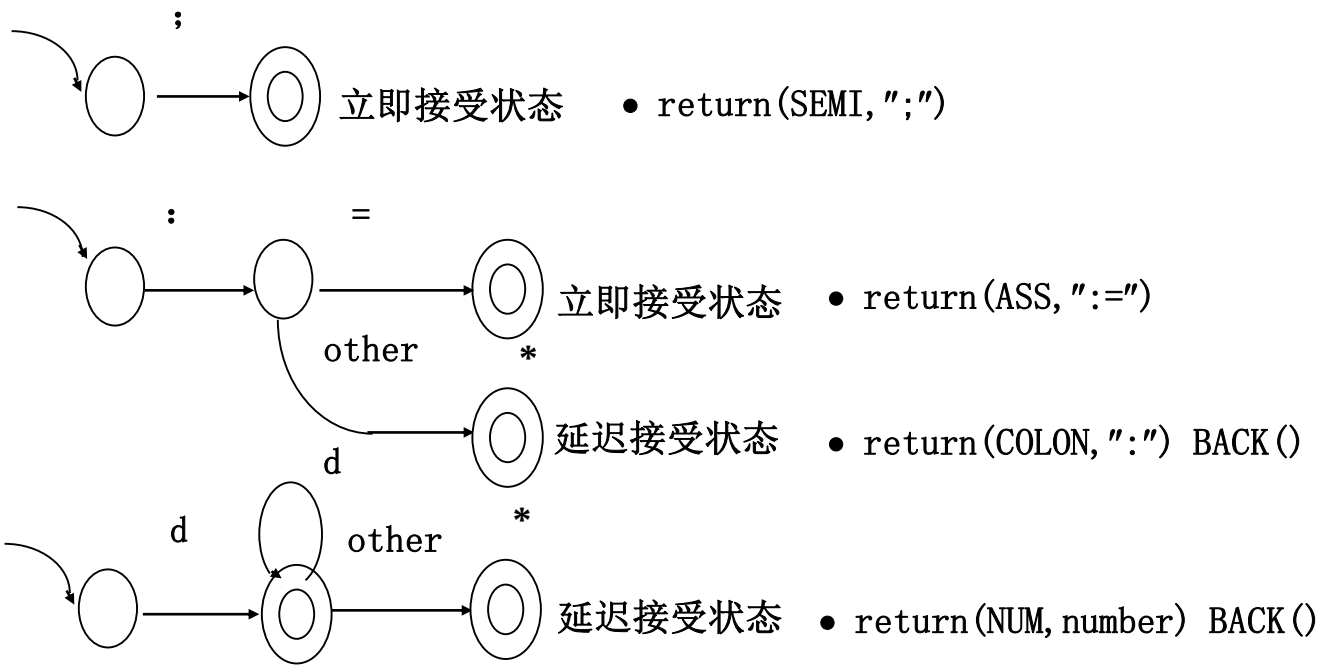
Information Science and Technology College of
Northeast Normal University

Implement Scanner with DFA

- **Some Issues**
  - **Independent or Attached**
  - **Skip these special characters**
    - **Blank, Tab, comments, return (line number)**
  - **When to stop scanning**
    - **At the end of the source file**
  - **Keywords & identifier**
  - **How to know the end of recognizing one token?**

Information Science and Technology College of
Northeast Normal University

立即接受状态和延迟接受状态



Information Science and Technology College of
Northeast Normal University

Defining Lexical Structure of *C0*

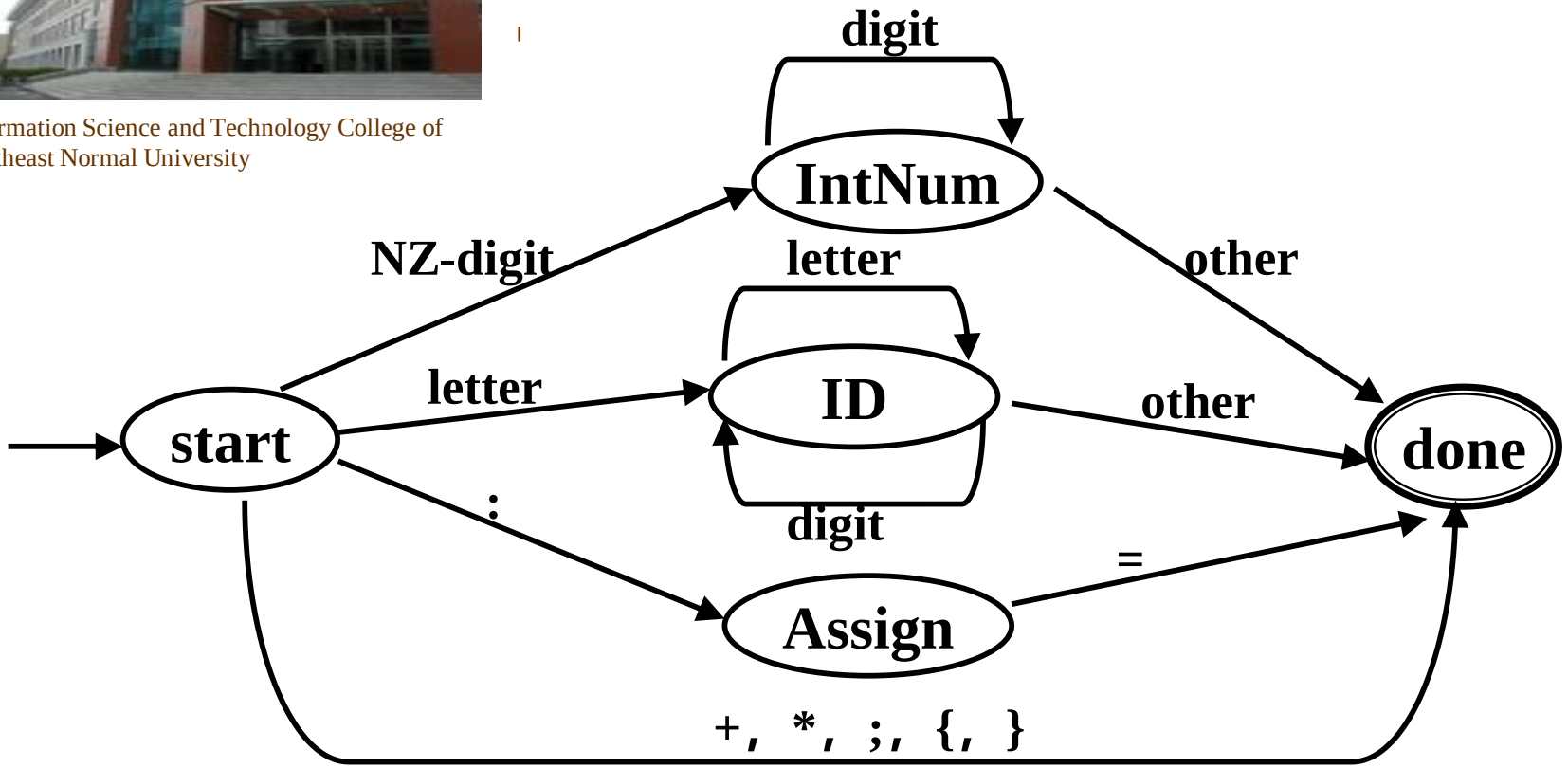
- **letter = a|…|z|A|…|Z**
- **digit = 0|…|9**
- **NZ-digit = 1|…|9**
- **Reserved = {| }| read| write**
- **Identifier =letter(letter|digit)***
- **Constant:**

  int = NZ-digit digit* | 0
- **Other symbols: syms = +|*| := | ;**
- **Lexical structure:**

  lex = Reserved | identifier |int | syms

Information Science and Technology College of Northeast Normal University

DFA for *C0*



Reserved(key)-words will be decided by checking identifier in reserved(key)-words table;

Information Science and Technology College of Northeast Normal University

Developing a Scanner from DFA

- **Input: a sequence of symbols, with a special symbol EOF as the end of the sequence;**
- **Output: a sequence of tokens;**

Information Science and Technology College of
Northeast Normal University

Developing a Scanner from DFA

Token Type:

```
typedef  enum { IDE, NUM, ASS,    //标识符，整数，赋值号
                PLUS, MINUS, SEMI,    // +, *, ;
                ，{ ，READ, WRITE ，} //keywords
               } TkType
```

Information Science and Technology College of
Northeast Normal University

Developing a Scanner from DFA

Data Structure for TOKEN:

```
struct Token {TkType  type;
              char val[50];}
```

Information Science and Technology College of Northeast Normal University

Developing a Scanner from DFA

Global Variables:

- char str[50]; ----- store the string has been read already;
- int len = 0; ----- the length of the str
- Token tk; ----- current token
- Token TokenList[100]; ---- the sequence of tokens
- int total = 0; ----- the number of tokens generated

Information Science and Technology College of
Northeast Normal University

Developing a Scanner from DFA

Predefined Functions:

- **bool ReadNextchar() --- read current symbol to CurrentChar,**
  if current symbol is EOF returns false;
  else returns true;
- **int IsKeyword(str) --- checking whether str is one of keywords,**
  if str is a keyword, it returns the number
  of the keywords;
  else it returns -1;
- **void SkipBlank() -- skip blank characters & return until read one other character to CurrentChar;**

Developing a Scanner from DFA

```
SkipBlank();
start:    case CurrentChar of
            "1..9": str[len] = CurrentChar; len++;  goto IntNum ;
            "a..z", "A..Z": str[len] = CurrentChar; len++; goto ID;
            ":": goto Assign;
            "+" : tk.type =PLUS; SkipBlank() ;goto Done;
            "*": tk.type = MINUS; SkipBlank() ;goto Done;
            ";" :tk.type = SEMI ; SkipBlank() ;goto Done;
          EOF: exit;
          other: error();
```

Information Science and Technology College of
Northeast Normal University

Developing a Scanner from DFA

```
IntNum:
    if  (not ReadNextchar())
     {if len !=0 {tk.type = NUM, strcpy(tk.val, str);
                  goto Done}}
    case CurrentChar of
        "0..9": str[len] = CurrentChar; len++; goto IntNum ;
        other:  tk.type = NUM, strcpy(tk.val, str);
```

Developing a Scanner from DFA

```
ID:
   if  (not ReadNextchar())
       {if len !=0 {tk.type = IDE, strcpy(tk.val, str);
                    goto Done}}

   case CurrentChar of

       "0..9": str[len] = CurrentChar; len++; goto ID;

       "a..z", "A..Z": str[len] = CurrentChar; len++; goto ID;

       other: if  IsKeyword(str)

                        {tk.type = IsKeyword(str) }

                  else  {tk.type = IDE, strcpy(tk.val, str) };

                  goto done;
```

Information Science and Technology College of
Northeast Normal University

Developing a Scanner from DFA

Assign:

```
if  (not ReadNextchar())  {if len !=0 error; exit;};
case CurrentChar of
     "=": Tk.type = ASS;
          goto Done ;
     other:error();
```

Information Science and Technology College of
Northeast Normal University

Developing a Scanner from DFA

```
Done:
  TokenList[total] = tk;          // add new token to the token list
  total ++;                       //
  len = 0;                        //start storing new token string
  strcpy(str, “”);                // reset the token string
  SkipBlank();                    // skip blank characters
  goto start;                     //start scanning new token
```

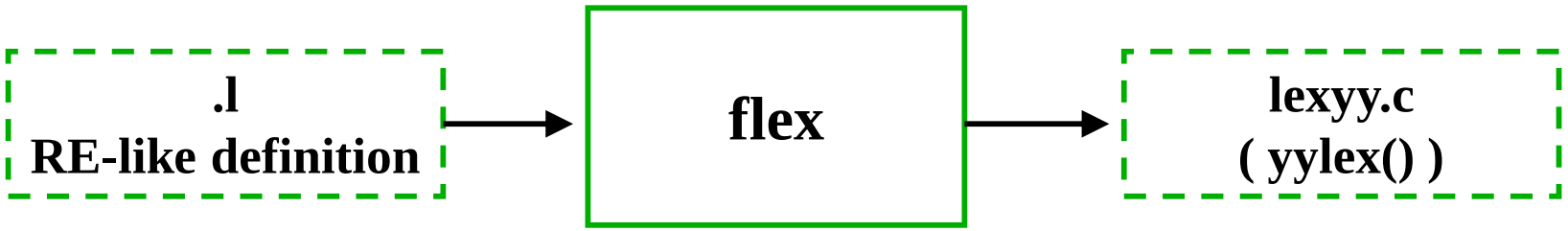
What are problems with this Scanner

- **str & TokenList use array, which is not practical;**
- **Not deal with errors;**
- **Not deal with line number;**

Information Science and Technology College of
Northeast Normal University

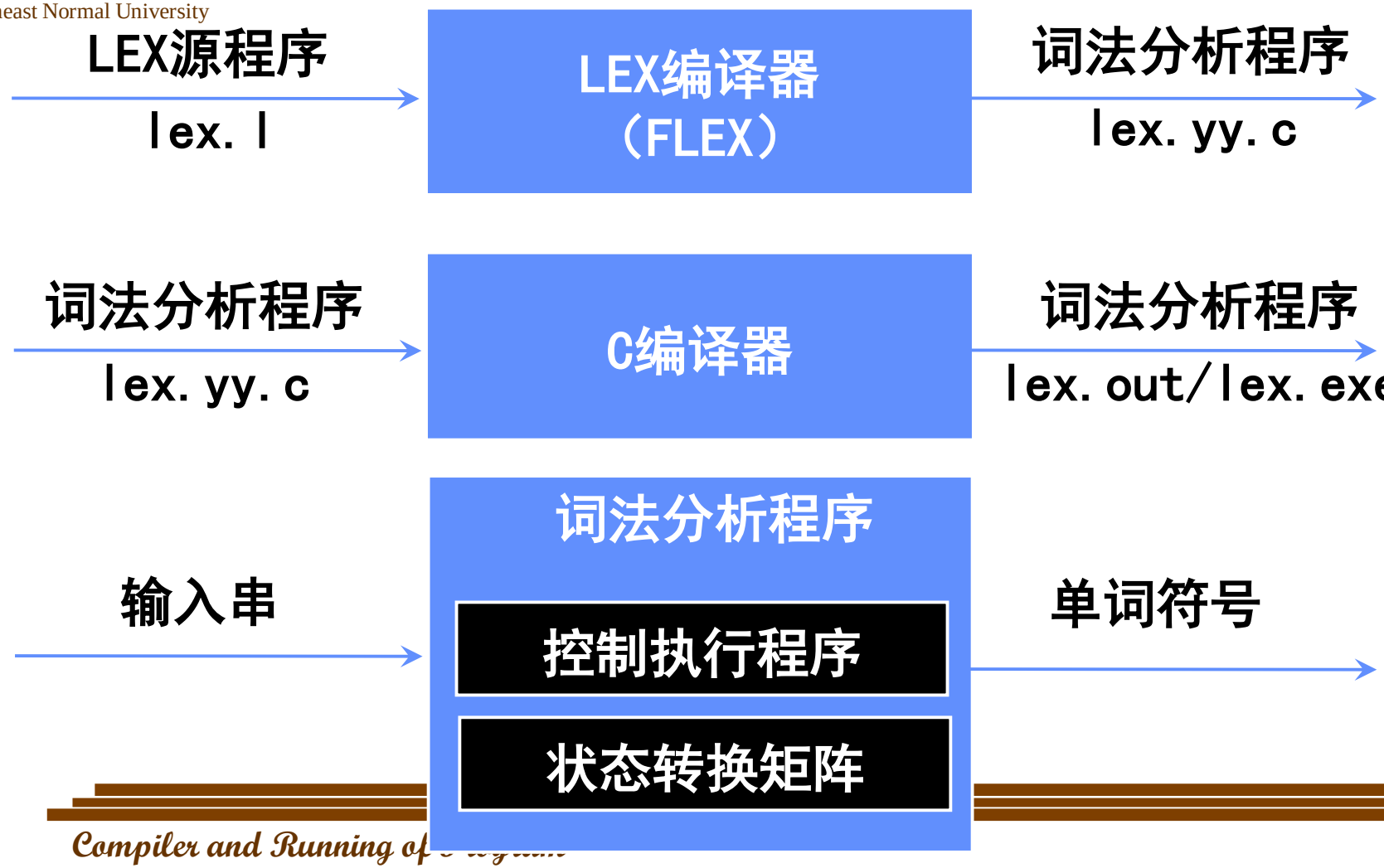
A Scanner Generator – Lex

- **Different versions of Lex;**
- **flex is distributed by GNU compiler package produced by the Free Software Foundation, which is freely available from Internet;**



Information Science and Technology College of Northeast Normal University

A Scanner Generator – Lex



Information Science and Technology College of

A Scanner Generator – Lex

AUXILIARY DEFINITION

letter→A|B|...|Z

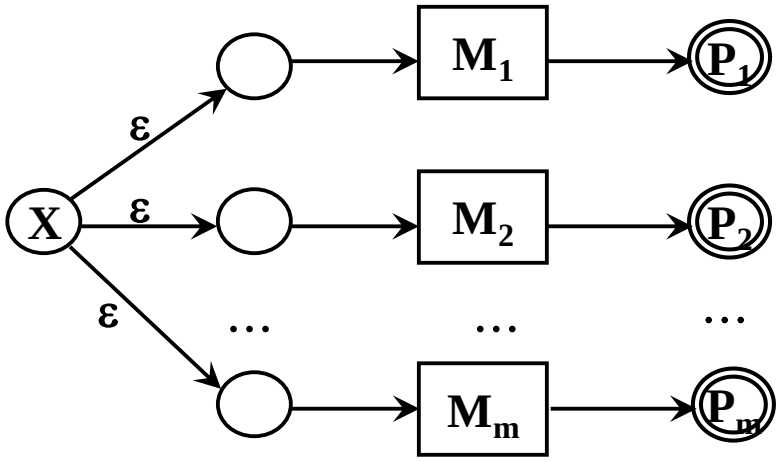
digit →0|1|...|9

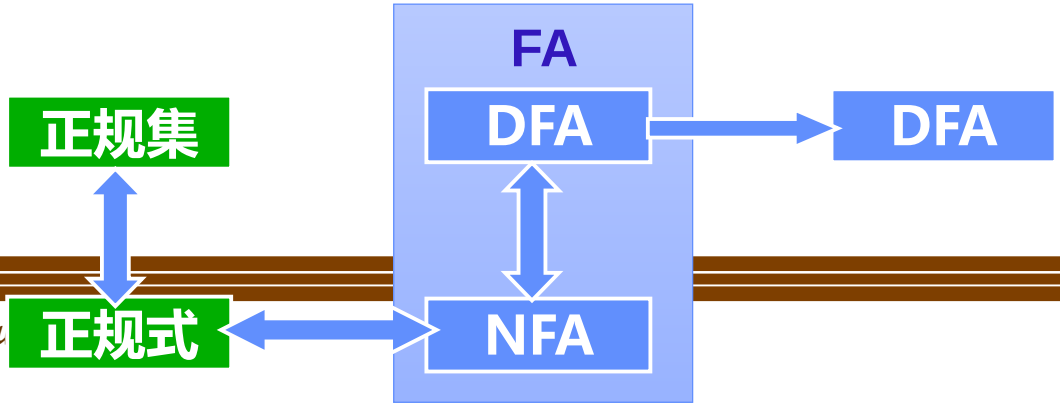
RECOGNITION RULES		
1	DIM	{ RETURN (1,-) }
2	IF	{ RETURN (2,-) }
3	DO	{ RETURN (3,-) }
4	STOP	{ RETURN (4,-) }
5	END	{ RETURN (5,-) }
6	letter(letter|digit)*	{ RETURN (6, TOKEN) }
7	digit(digit)*	{ RETURN (7, DTB) }
8	=	{ RETURN (8, -) }
9	+	{ RETURN (9,-) }
10	*	{ RETURN (10,-) }
11	**	{ RETURN (11,-) }
12	,	{ RETURN (12,-) }
13	(	{ RETURN (13,-) }
14	)	{ RETURN (14,-) }

Information Science and Technology College of
Northeast Normal University

A Scanner Generator – Lex

- LEX的工作过程
  - 对每条识别规则P_i构造一个相应的非确定有限自动机M_i；
  - 引进一个新初态X，通过ε弧，将这些自动机连接成一个新的NFA；
  - 把M确定化、最小化，生成该DFA的状态转换表和控制执行程序





Practice（I）

- **Developing a Scanner Manually for *C0* ;**
- **Test the Scanner;**

Practice（II）

- **Get Lex-similar free software;**
- **Get to know its specification language;**
- **Try to define the lexeme of *C0* with the specification language;**
- **Implement Scanner for *C0* with the software;**
- **Test the Scanner;**

Summary for § 2. Scanning

Information Science and Technology College of
Northeast Normal University

What have been introduced

- √ 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

Summary

- About ***finite automata***
  - Definition of DFA
    - (∑, start state, set of states, set of terminate states, f)
  - Definition of NFA
    - (∑, set of start states, set of states, set of terminate states, f)
  - Differences between NFA and DFA
    - Number of start states
    - ε
    - Allows more than one ledges for a state and one same symbol

Summary

- About ***finite automata***
  - From NFA to DFA
    - main idea
    - solve problem
  - Minimizing DFA
    - main idea
    - solve problem
  - Implementing DFA
    - Table based
    - Graph based

Summary

- **About *regular expressions***
  - **Definition of regular expression**
  - **Regular definition**
  - **From regular expression to NFA**
  - **Defining lexical structure with regular expression**

Summary

- **About *scanner***
  - **Defining lexical structure of the programming language with regular expression**
  - **Transforming regular expression into NFA**
  - **Transforming NFA into DFA**
  - **Minimizing DFA**
  - **Implementing DFA**

Summary

- **Original Problem**
  - **Develop a Scanner**
    - **Read source program in the form of stream of characters, and recognize tokens with respect to lexical rules of source language;**
- **General techniques**
  - **Use RE to define Lexical structure;**
  - **RE -> NFA -> DFA -> minimized DFA -> implement**
- **General Problem**
  - **Use RE/FA to define the structural rules**
  - **Check whether the input meets the structural rules**

Summary

- **Application in similar problems**
  - **Use RE(DFA) to formally describe the structures**
    - **Strings**
    - **Security policies**
    - **Interface specification (component contract)**
  - **Check**
    - **whether a string meets the structural rules;**
    - **Whether certain execution meets security policies;**
    - **Properties checking**

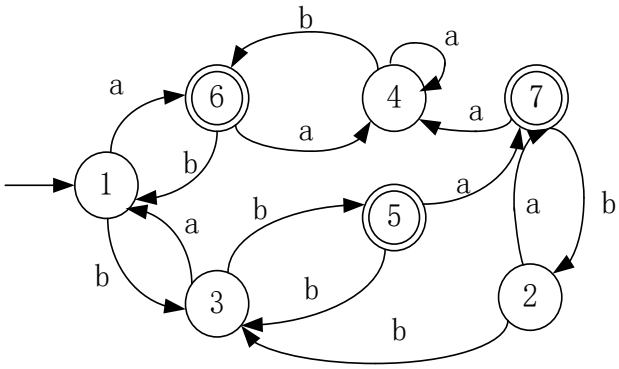
Information Science and Technology College of Northeast Normal University

Any Questions?

Reading Assignment

- **Topic: How to develop a parser(语法分析器)?**
- **Objectives**
  - **Get to know**
    - **What is a parser? (input, output, functions)**
    - **The Syntactical structure of a C programs?**
    - **Different Parsing techniques and their main idea?**
- **References**
  - **Optional textbooks**
- **Hand in a report either in English or in Chinese, and one group will be asked to give a presentation at the beginning of next class;**
- **Tips:**
  - **Collect more information from textbooks and internet;**
  - **Establish your own opinion;**

1. 画出与正则表达式a（ab|c）*等价的确定有限自动机。
2. 请用状态分离法将下面的DFA化简。



3. 给出自动机DFA的正则表达式。