第2讲
词法分析-有限自动机

李 诚

国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

计算机科学与技术学院

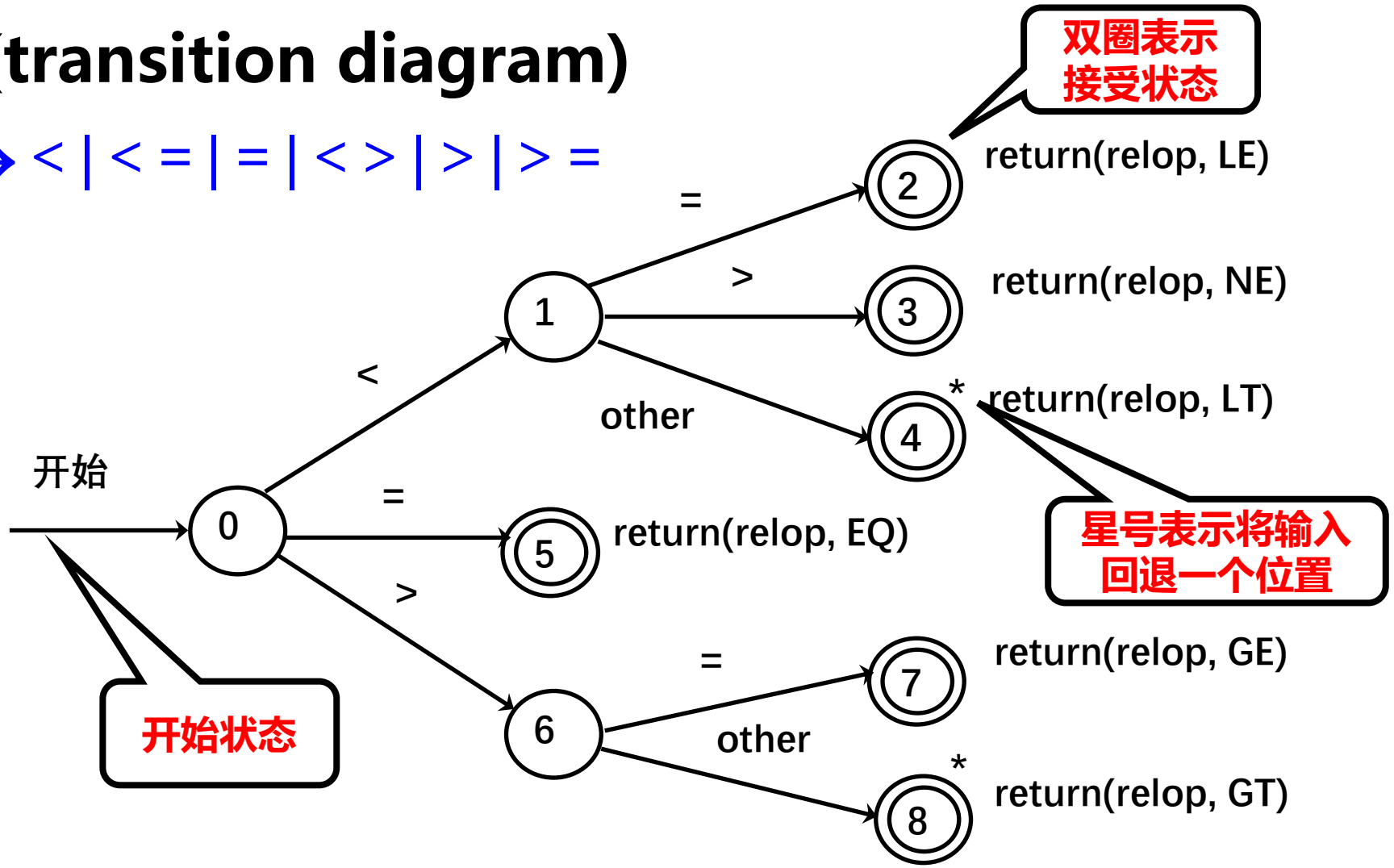
2023年09月06日

两个问题！

- 如何判定一个串匹配某个正则表达式?
- 如何形式化地描述这个匹配过程?

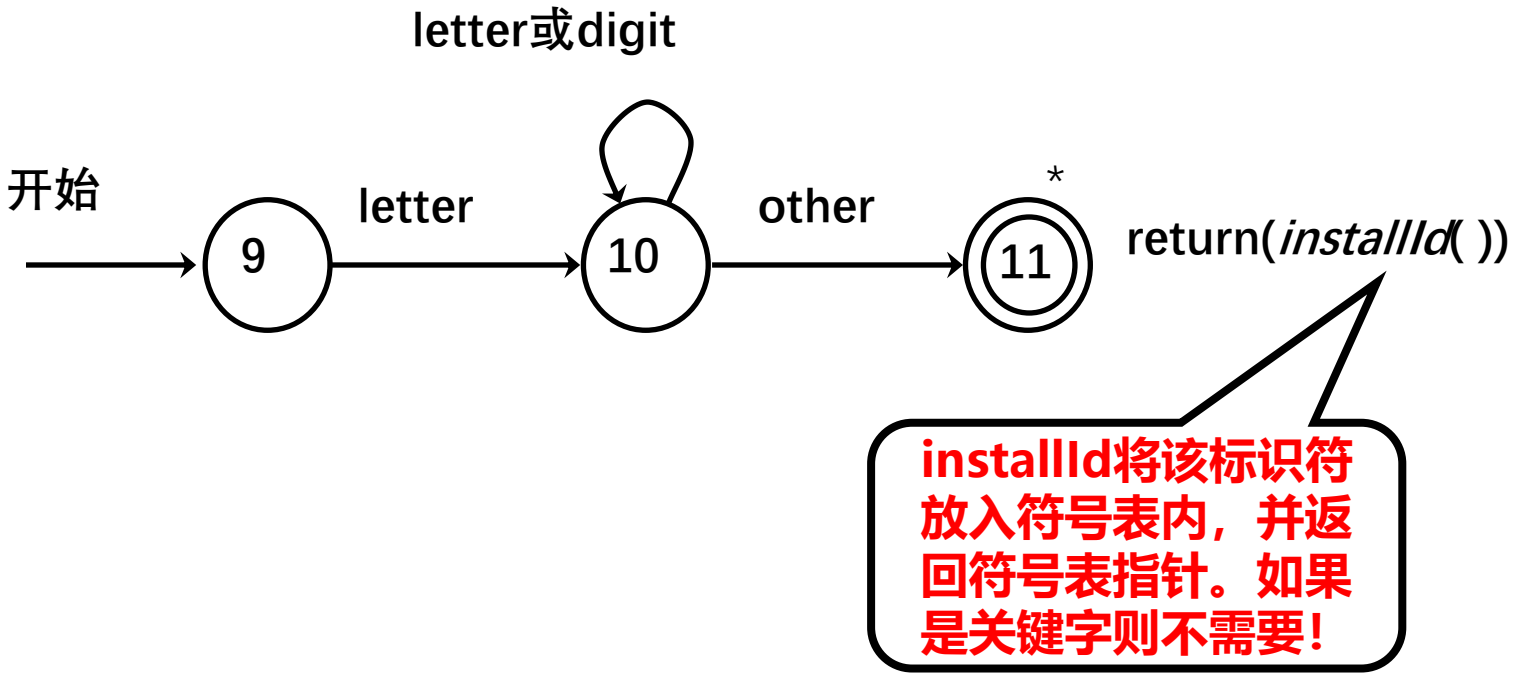
词法记号的识别：转换图

- **转换图(transition diagram)**
  - ❖ **relop → < | < = | = | < > | > | > =**



状态转换图

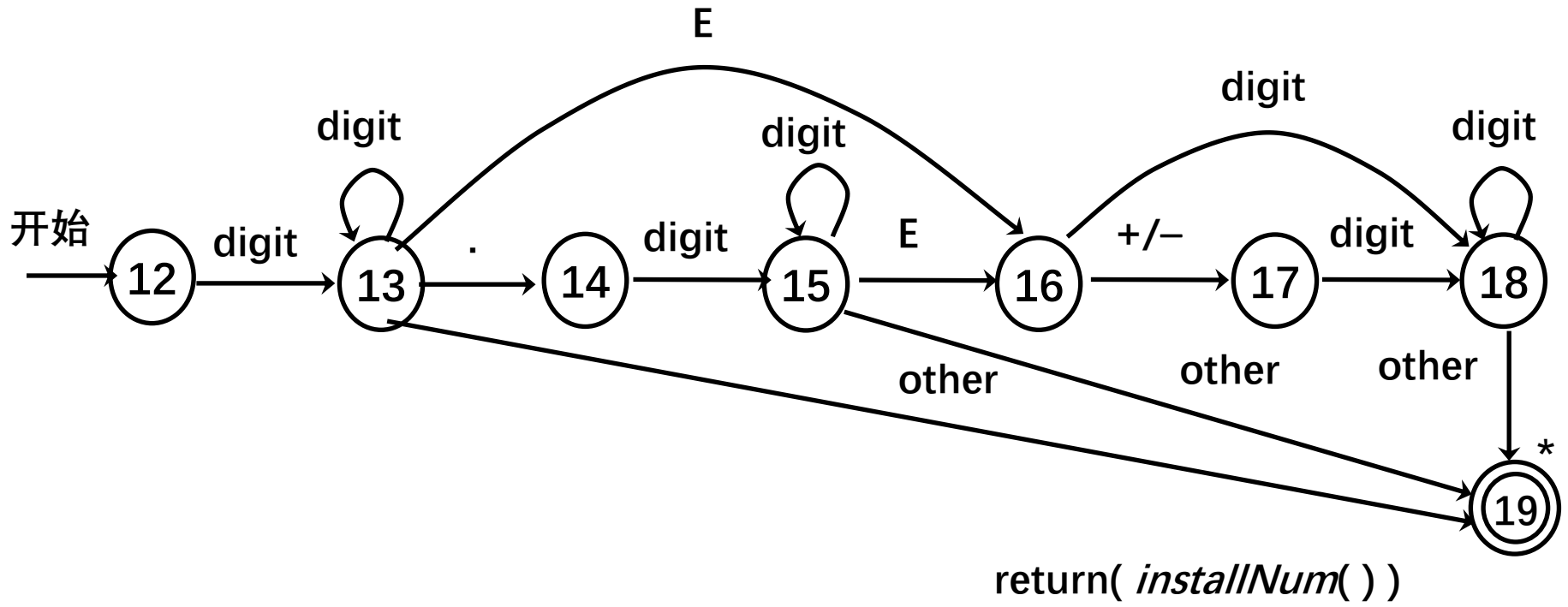
- 标识符和关键字的转换图
  - id → letter (letter | digit)*



状态转换图

- **无符号数的转换图**

number → digit$^+$ (.digit$^+$)? (E[+–]? digit$^+$)?

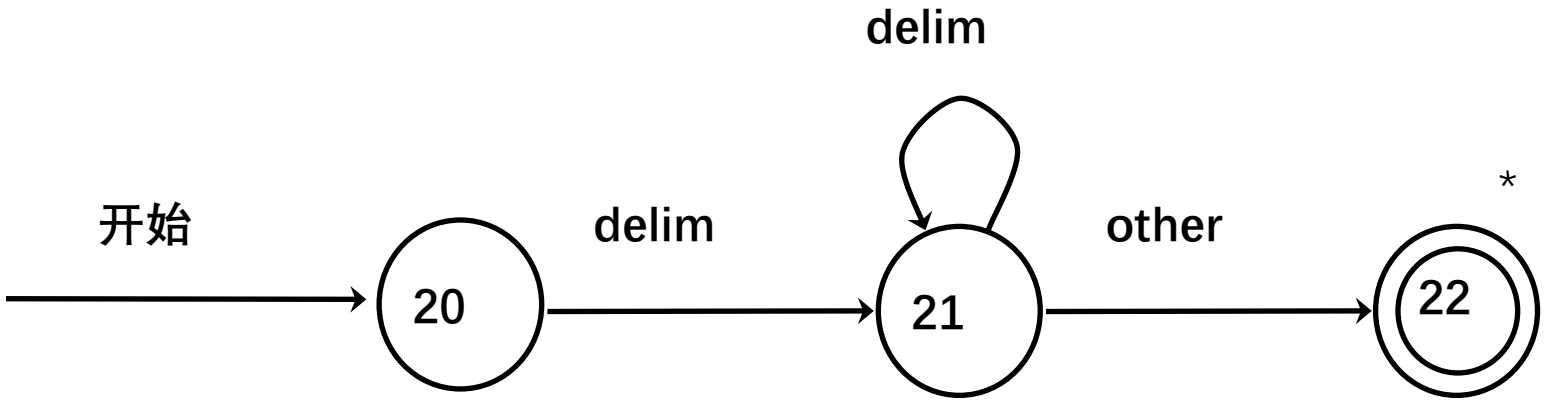


状态转换图

- **空白的转换图**

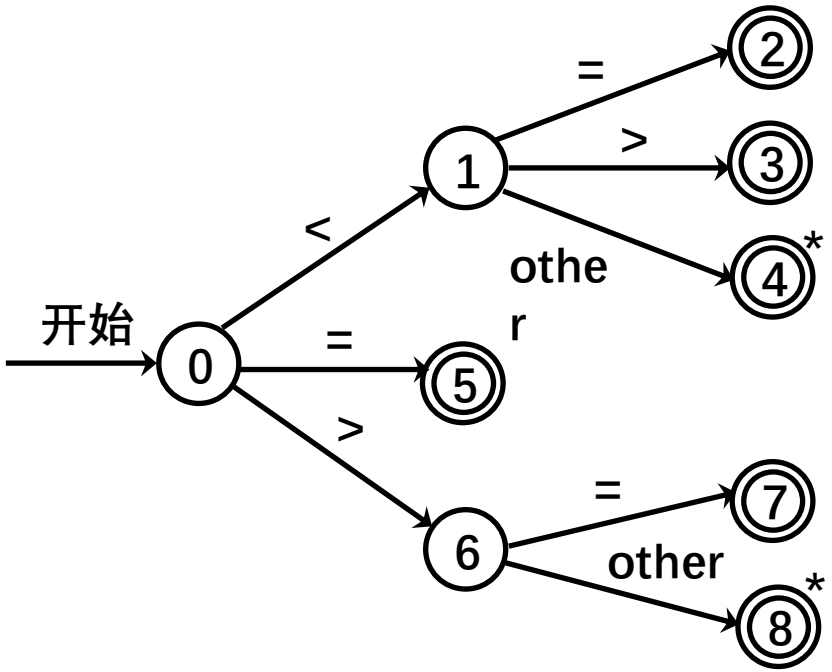
delim → blank | tab | newline

ws → delim+



基于转换图的词法分析

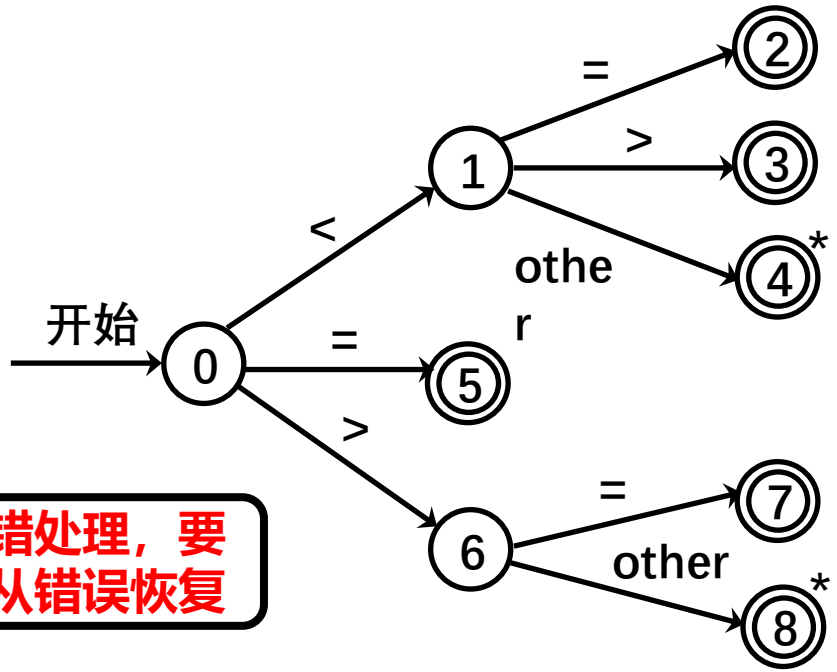
- 例：relop的转换图的概要实现



基于转换图的词法分析

- 例：relop的转换图的概要实现

```
TOKEN getRelop() {
    TOKEN retToken = new(RELOP);
    while (1) {
        switch (state) {
        case 0: c = nextChar();
                if (c == '<') state = 1;
                else if (c == '=') state = 5;
                else if (c == '>') state = 6;
                else fail();
                break;
        case 1: ...
        ...
        case 8: retract();
                retToken.attribute = GT;
        return(retToken);
       }
    }
}
```



出错处理，要能从错误恢复

回退

词法分析中的冲突及解决

R = Whitespace | Integer | Identifier| ‘+’

识别 “foo+3”

- “f” 匹配 R, 更精确地说是 Identifier
- 但是 “fo”也匹配 R, “foo” 也匹配, 但 “foo+”不匹配

如何处理输入？如果

- $x_1 \ldots x_i \in L(R)$ 并且 $x_1 \ldots x_k \in L(R)$

Maximal match 规则:

- 选择匹配 R 的最长前缀

词法分析中的冲突及解决

R = Whitespace| ‘new’ | Integer| Identifier识别 “new foo”

- “new” 匹配 R, 更精确地说是 ‘new’
- 但是 “new”也匹配 Identifier

如何处理输入？ 如果

- $x_1 \ldots x_i \in L(R_j)$ 并且 $x_1 \ldots x_i \in L(R_k)$

优先 match 规则:

- 选择先列出的模式 (j 如果 j < k)
- 必须将 ‘new’列在 Identifier的前面

词法错误

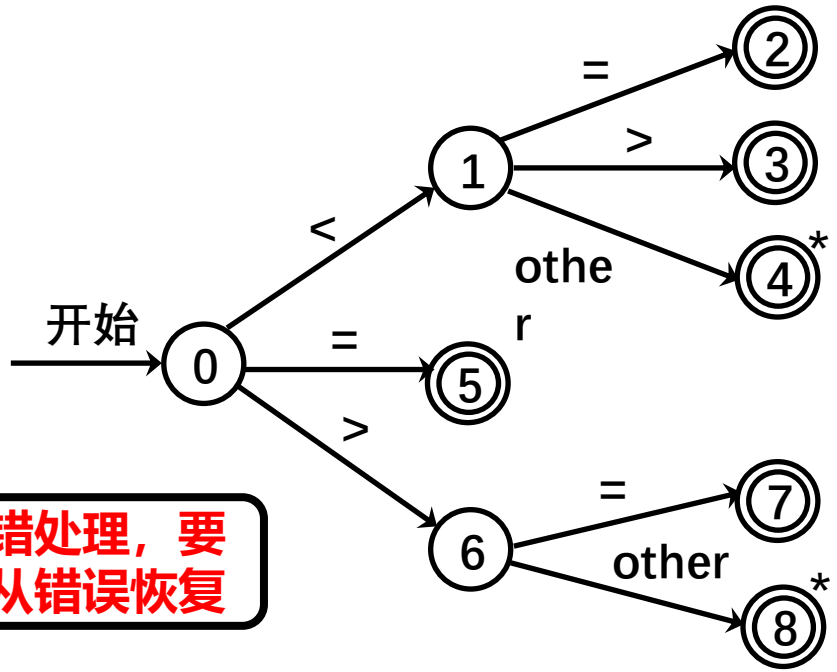
- **词法分析器对源程序采取非常局部的观点**
  - 例：难以发现下面的错误

    fi (a == f (x)) …
- **在实数是“数字串.数字串”格式下**
  - 可以发现 **123.x** 中的错误
- **紧急方式的错误恢复**
  - 删掉当前若干个字符，直至能读出正确的记号
  - 会给语法分析器带来混乱
- **错误修补**
  - 进行增、删、替换和交换字符的尝试
  - 变换代价太高，不值得

基于转换图的词法分析

- 例：relop的转换图的概要实现

```
TOKEN getRelop() {
    TOKEN retToken = new(RELOP);
    while (1) {
        switch (state) {
        case 0: c = nextChar();
                if (c == '<') state = 1;
                else if (c == '=') state = 5;
                else if (c == '>') state = 6;
                else fail();
                break;
        case 1: ...
        ...
        }
    }
}
```



出错处理，要能从错误恢复

问题：怎么为每一个正则定义自动找到一个状态转换图?

两个问题！

- 如何判定一个串匹配某个正则表达式?
- 如何形式化地描述这个匹配过程?

有限自动机的定义

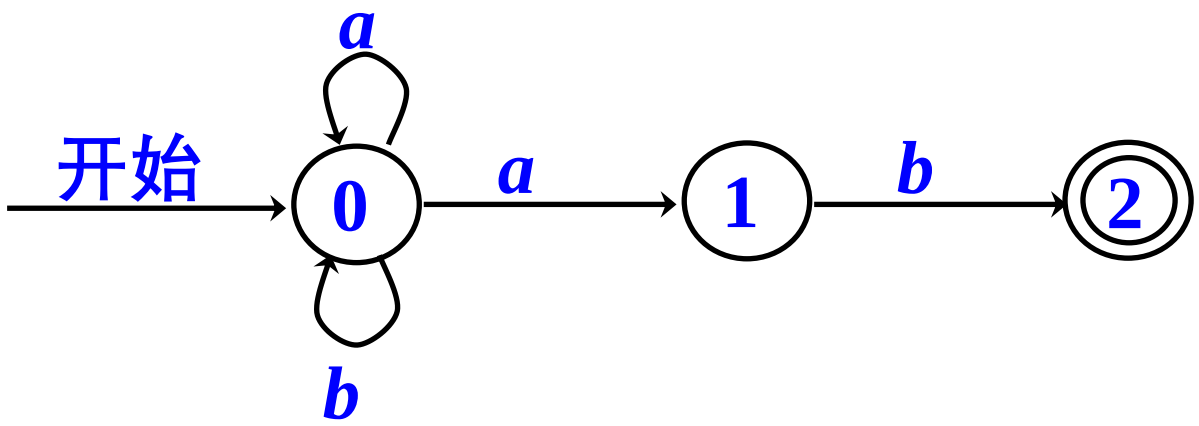
- （不确定的）有限自动机NFA是一个数学模型，它包括：
  - 有限的状态集合S
  - 输入符号集合$\sum$
  - 转换函数$move: S \times (\sum \cup \{\varepsilon\}) \rightarrow P(S)$
  - 状态s_0是唯一的开始状态
  - $F \subseteq S$是接受状态集合

幂集

有限自动机的定义

- （不确定的）有限自动机NFA是一个数学模型，它包括：
  - 有限的状态集合S
  - 输入符号集合$\sum$
  - 转换函数$move: S \times (\sum \cup \{\varepsilon\}) \rightarrow P(S)$
  - 状态s_0是唯一的开始状态
  - $F \subseteq S$是接受状态集合

识别语言
$(a|b)^*ab$
的NFA

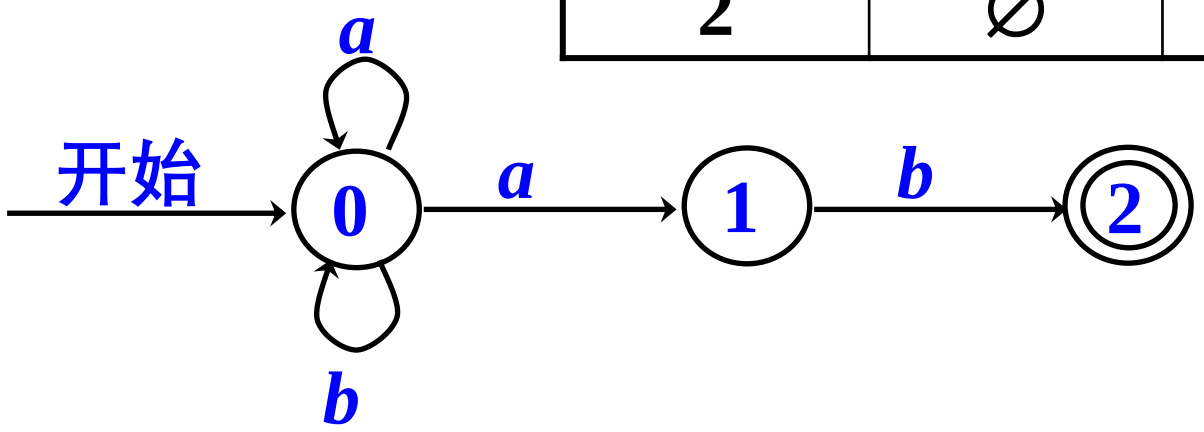


有限自动机的实现

- 构造状态之间的转换表，在读入字符串的过程中，不停查表，直至到达接受状态
- 或者，报告非法输入

	输 入 符 号	
	a	*b*
0	{0, 1}	{0}
1	∅	{2}
2	∅	∅

识别语言
$(a|b)^*ab$
的NFA

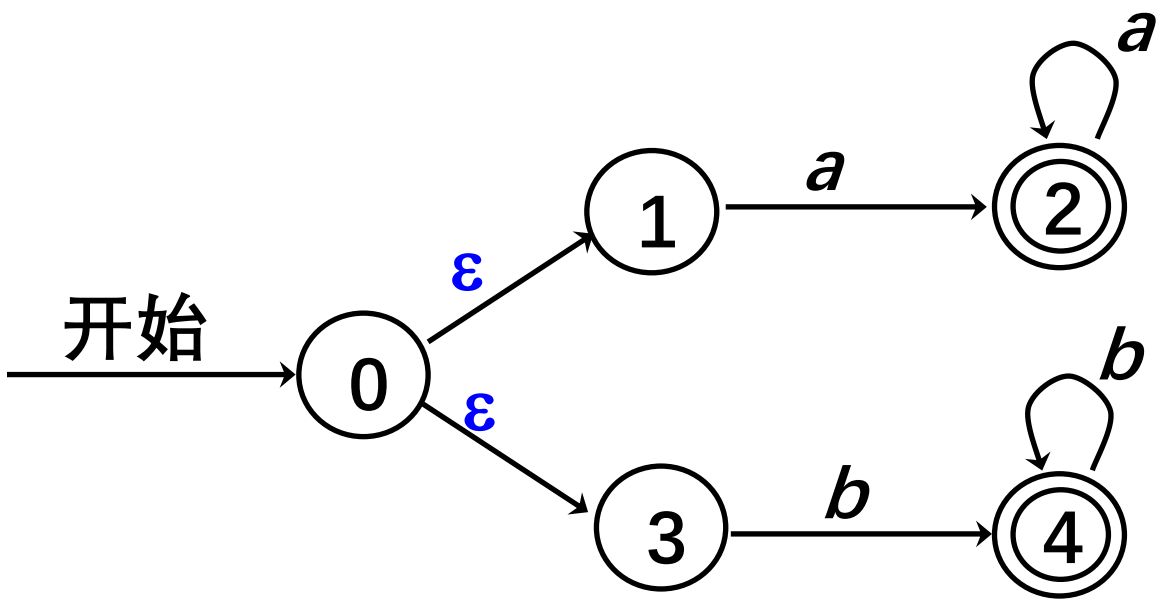


有限自动机

- 例 识别aa*|bb*的NFA

有限自动机

- 例 识别aa*|bb*的NFA



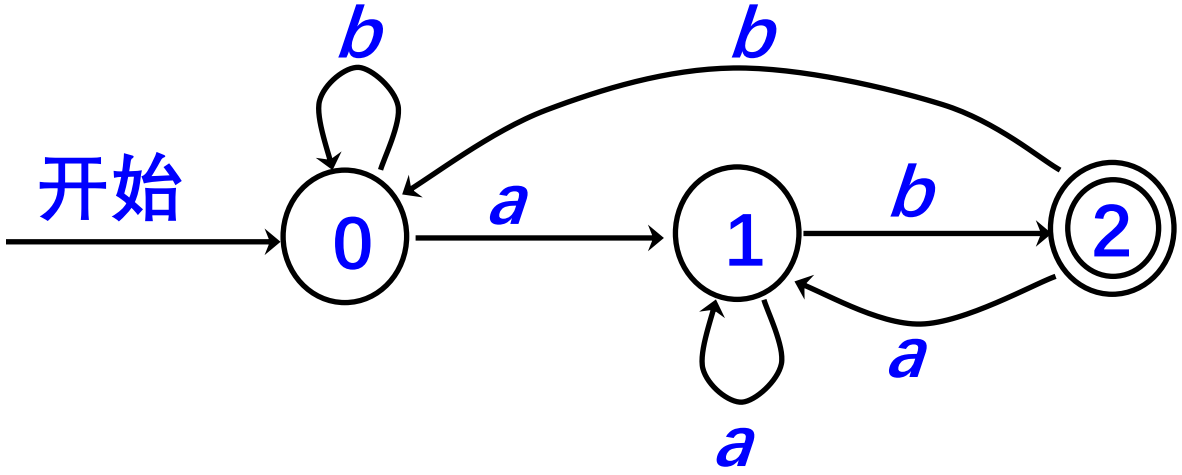
利用NFA识别token的问题

- 转换函数move：$S \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(S)$
- 对于一个token，
  - 有可能要尝试很多不同的路径，
  - 大部分路径都是白费功夫
  - 尝试+回退的方式 => 效率很低
  - 考虑很多project，百万行代码+
- 思考：**有没有一种确定的形式化描述，对于输入的一个符号，只有唯一的跳转?**

有限自动机

- **确定的有限自动机（简称DFA)也是一个数学模型，包括：**
  - 有限的状态集合S
  - 输入符号集合$\sum$
  - 转换函数$move: S \times \sum \to S$，且可以是部分函数
  - 状态s_0是唯一的开始状态
  - $F \subseteq S$是接受状态集合

识别语言
$(a|b)^*ab$
的DFA

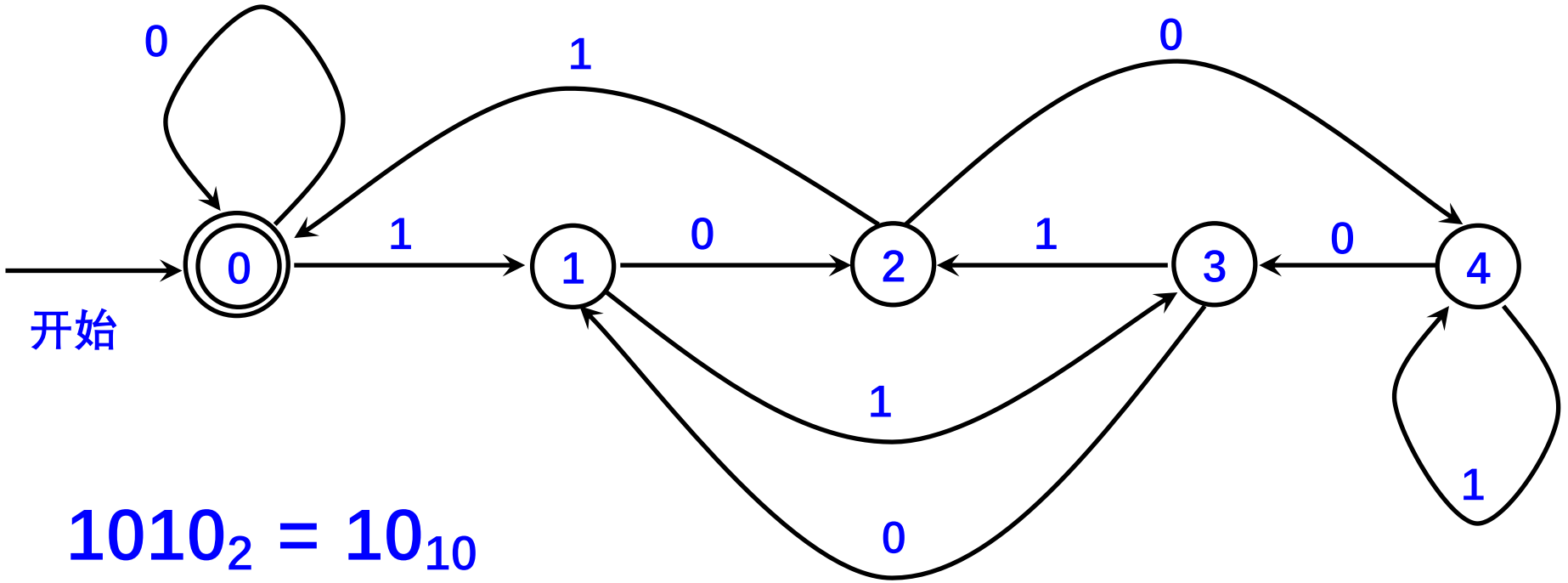


有限自动机

- 例 DFA，识别{0,1}上能被5整除的二进制数

有限自动机

- 例 DFA，识别{0,1}上能被5整除的二进制数

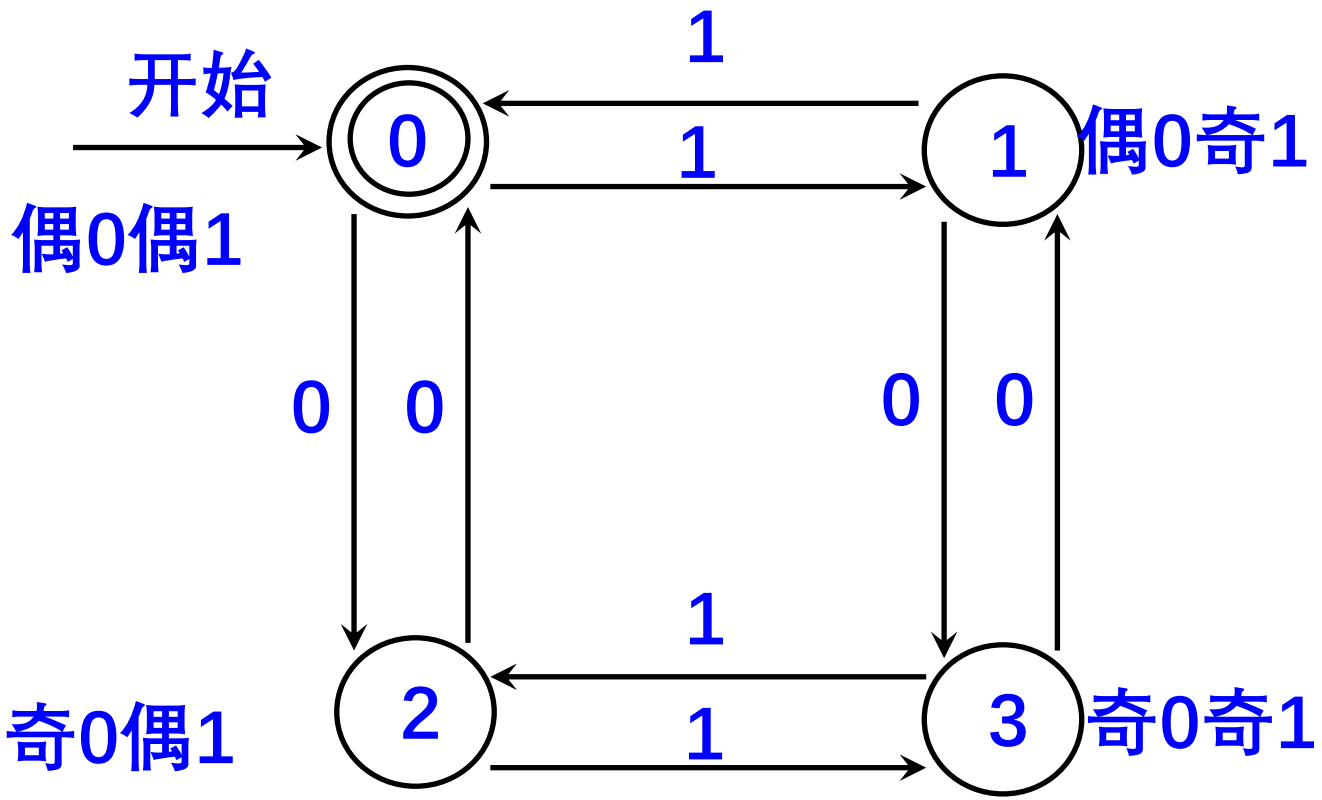


$1010_2 = 10_{10}$

$111_2 = 7_{10}$

有限自动机

- 例　DFA,接受 0和1的个数都是偶数的字符串

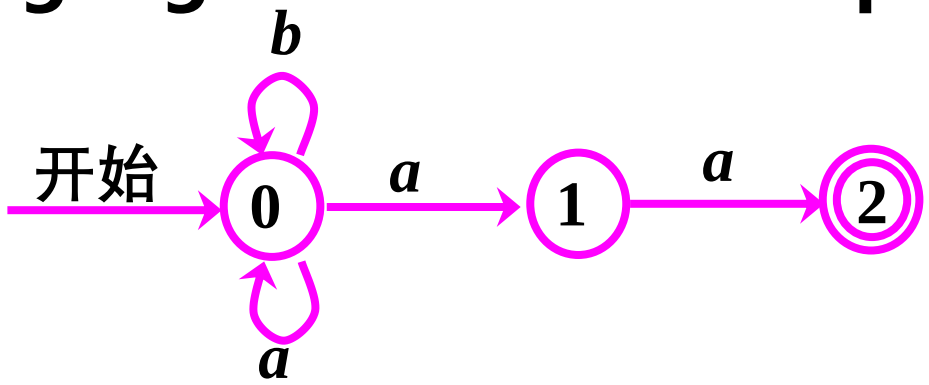


NFA vs. DFA

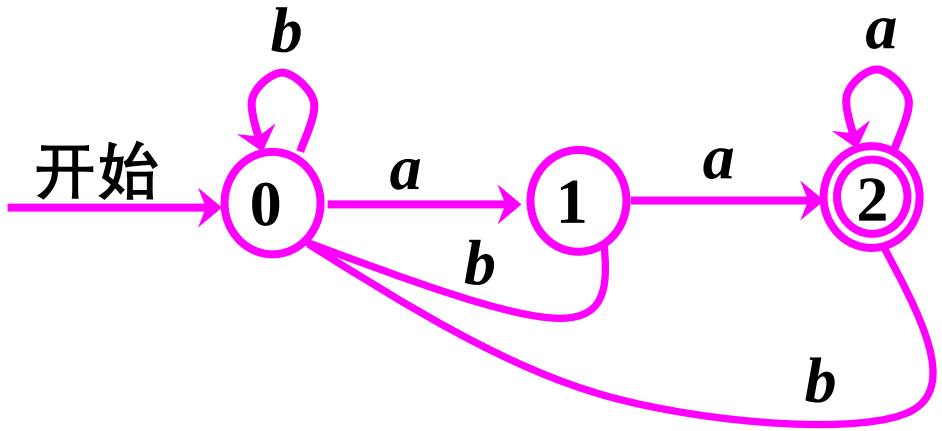
- NFAs and DFAs recognize the same set of languages (regular languages)
- Major differences:
  - Move function
    - $S \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(S)$ *NFA*
    - $S \times \Sigma \rightarrow S$ *DFA*
  - DFA does not accept ε as input
- DFAs are faster to execute
  - There are no choices to consider

NFA vs. DFA

- For a given language NFA can be simpler than DFA



- DFA can be exponentially larger than NFA



一起努力
打造国产基础软硬件体系！

李 诚

国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

计算机科学与技术学院

2023年09月06日