2.3 记号的识别－有限自动机

模式的描述—正规式

记号的识别—有限自动机（确定、不确定）

2.3.1 不确定的有限自动机

（Nondeterministic Finite Automaton, NFA）

定义2.4 NFA是一个五元组（5-tuple）：

M =（S, ∑, move, s0, F），其中

（1） S是有限个状态（state）的集合；

（2） ∑是有限个输入字符（包括ε）的集合；

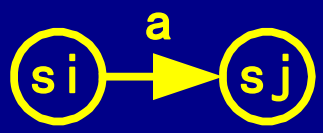
（3） move是一个状态转移关系，move(si, ch)=sj表示，当前状态si下若遇到输入字符ch，则转移到状态sj；

（4） s0是唯一的初态（也称开始状态）；

（5） F是终态集（也称接受状态集），它是S的子集，包含了所有的终态。 ■

〈1〉直观的表示方式

① 状态转换图：用一个有向图来直观表示NFA。



NFA中的每个状态，对应转换图中的一个节点；

NFA中的每个move(si, a)=sj，对应转换图中的一条有向边；

它表示从si节点出发，进入sj节点，字符a（或ε）是边上的标记。

② 状态转换矩阵：用一个二维数组来直观表示NFA。

矩阵中，一个状态对应一行，一个字符对应一列；

每个矩阵元素M[si,a]中的内容，是从状态si出发，经字符a（或ε）所到达的下一状态sj的集合；

在转换矩阵中，一般以矩阵第一行所对应的状态为初态，而终态需要特别指出。

	...	a
i		j

例2.7 识别正规式(a|b)*abb所描述正规集的NFA的三种表示形式分别如下。（其中，转换矩阵表示中0为初态，3为终态）

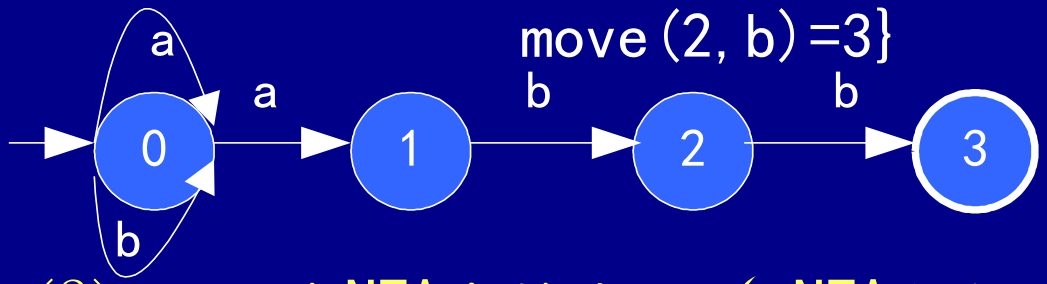
按定义：S={0, 1, 2, 3}, Σ={a, b} s0 = 0, F={3}

move={ move(0, a)=0, move(0, a)=1,
move(0, b)=0, move(1, b)=2,
move(2, b)=3}

按状态转换矩阵：

	a	b
0	{0, 1}	{0}
1	–	{2}
2	–	{3}
3	–	–

按状态转换图：



〈2〉记号在NFA中的表现（NFA如何识别记号）

对字符串，从初态开始，经一系列状态转移到达终态。

例如：对于字符串abb，有

定义：move(0, a)=1, move(1, b)=2, move(2, b)=3

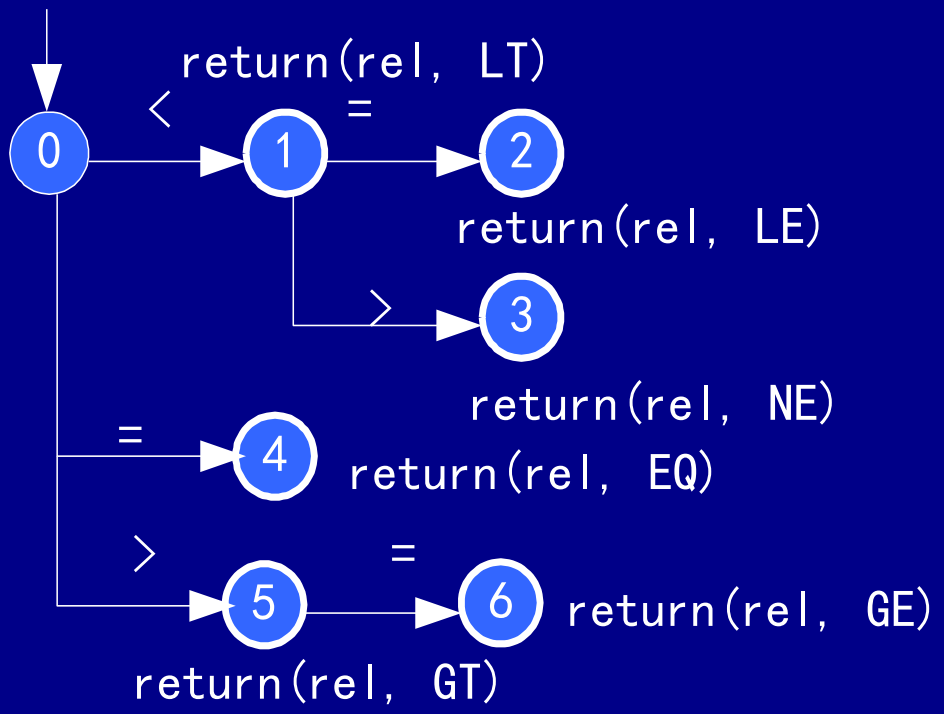
转换矩阵：m[0, a]={0, 1}, m[1, b]=2, m[2, b]=3

转换图：0a1b2b3

显然，转换图最直观，即每一个记号，实质上是从初态开始到某个终态的路径上的标记。

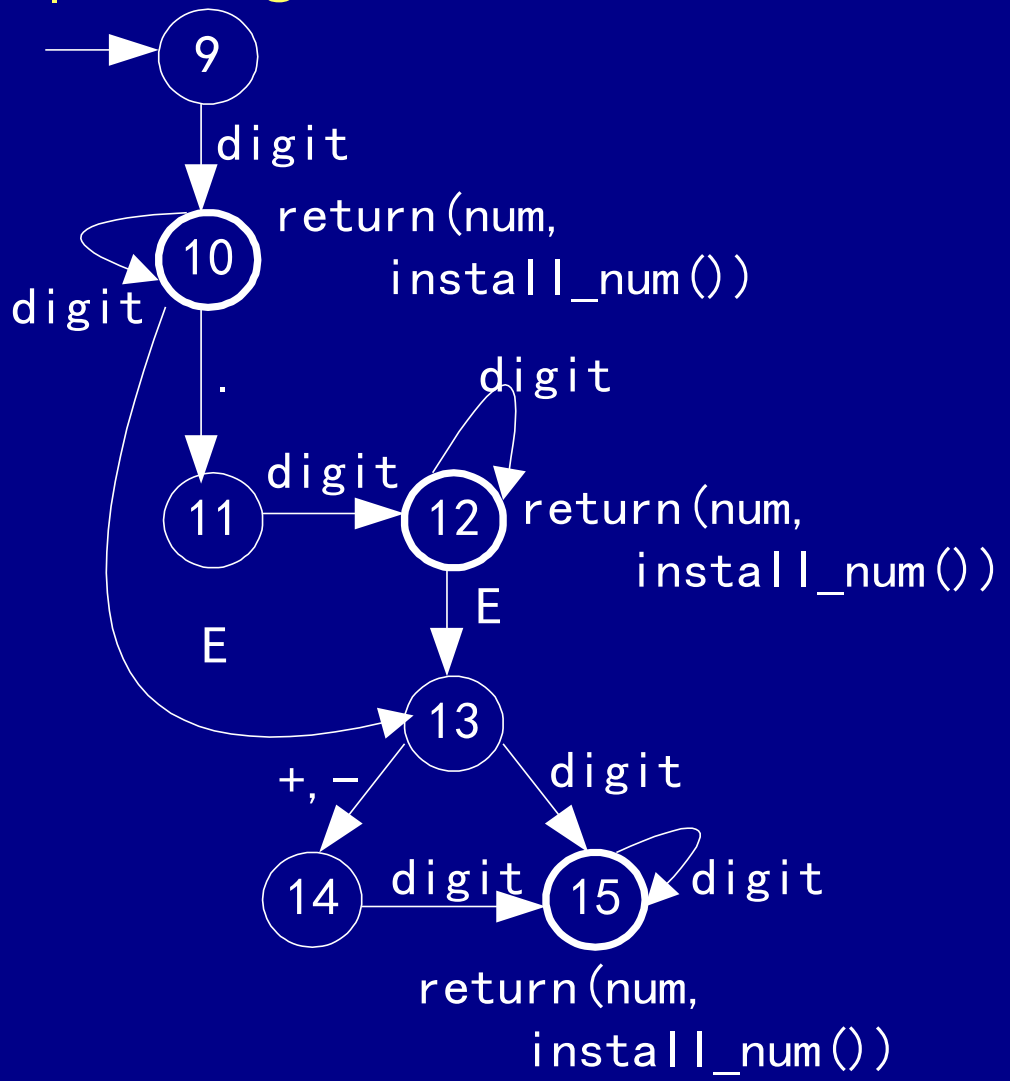
例2.8 识别表2.1中记号relation、id和num的转换图

relation = < | <= | <> | > | >= | =



id = char(char|digit)*

```
optional_fraction = ( . digits )?
optional_exponent = ( E (+|-)? digits )?
num = digits
      optional_fraction
      optional_exponent
```



〈3〉NFA （识别记号）的特点

NFA识别记号的最大特点是它的不确定性，即在当前状态下对同一字符有多于一个的下一状态转移。

〈4〉不确定性的表现

- 定义：move是1对多的；
- 转换图：同一状态有多于一条边标记相同字符转移到不同的状态；
- 转换矩阵：M[si,a]是一个状态的集合。

正规式：a(a|b)*abb

按定义：

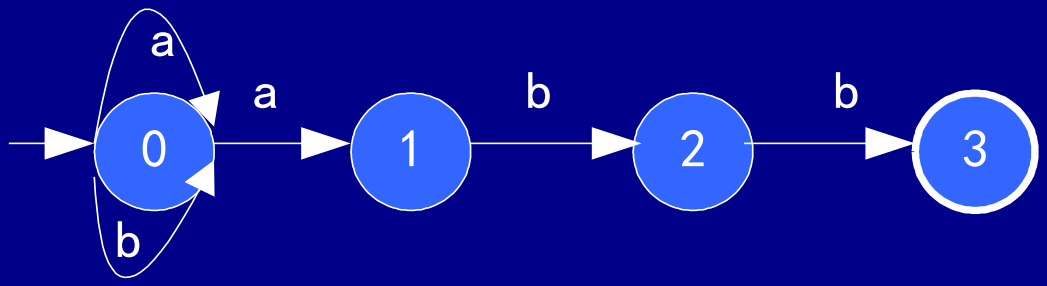
S={0, 1, 2, 3}, Σ={a, b} s0 = 0, F={3}

move={ move(0, a)=0, move(0, a)=1,

move(0, b)=0, move(1, b)=2,

move(2, b)=3}

按状态转换图：



按状态转换矩阵：

	a	b
0	{0, 1}	{0}
1	–	{2}
2	–	{3}
3	–	–

0是初态，3是终态

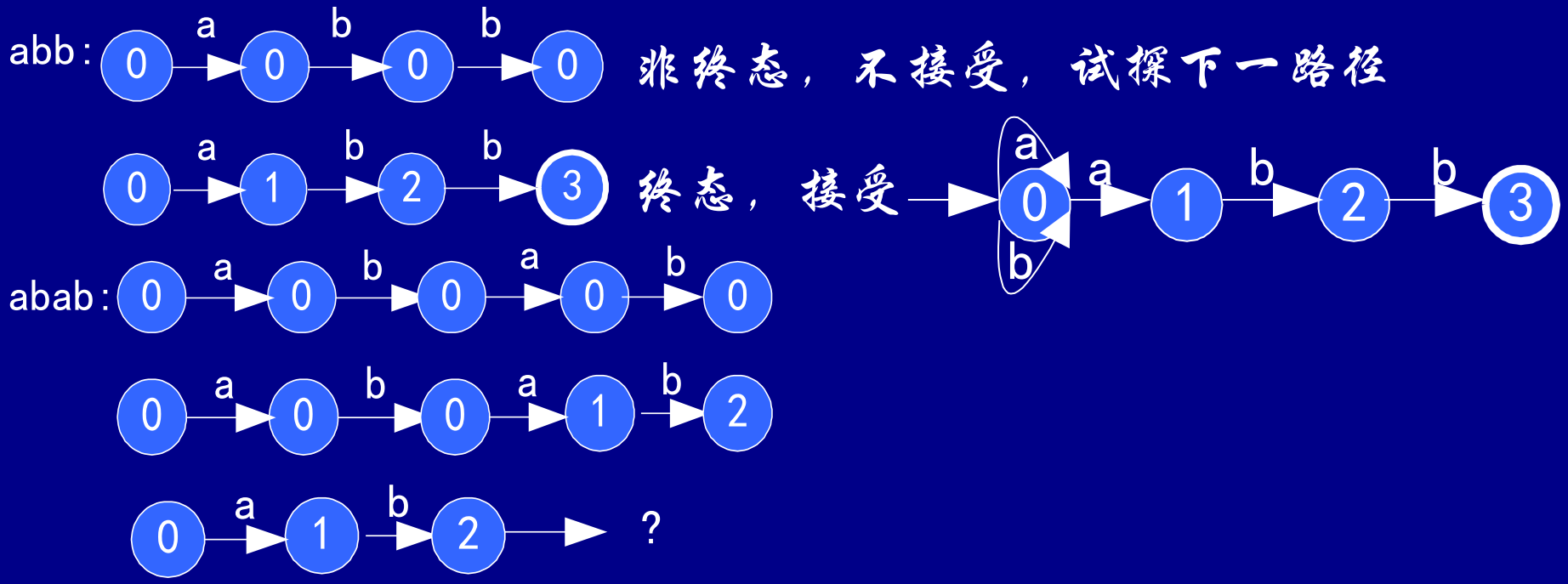
〈5〉 NFA识别输入序列的一般方法

反复试探所有路径，直到到达终态，或者到达不了终态。

〈6〉 不确定性识别记号的困惑

识别输入序列时，在当前状态下遇到同一字符，应转移到哪个下一状态？

例2.9 在正规式 (a|b)*abb的NFA上识别输入序列abb和abab：



〈7〉NFA识别记号存在的问题

1. 只有尝试了全部可能的路径，才能确定一个输入序列不被接受，而这些路径的条数随着路径长度的增长成指数增长。
2. 识别过程中需要进行大量回溯，时间复杂度升高且算法趋于复杂。

问题：

是否可以构造这样的有限自动机，它识别正规式所描述的字符串，且在任何一个状态下遇到同一字符最多有一个状态转移？

2.3.2 确定的有限自动机（Deterministic Finite Automaton, DFA）

定义2.5 DFA是NFA的一个特例，其中：

（1）没有状态具有ε状态转移（ε-transition），即状态转换图中没有标记ε的边；

（2）对每个状态s和每个字符a，最多有一个下一状态。

■

与NFA相比，DFA的特征（确定性）

定义：　move（si, a）是一个状态转移函数；

转换图：　从一个节点出发的边上标记均不相同；

转换矩阵：M[si,a]是一个状态。

且字母表不包括ε。

对NFA施加两条限制：

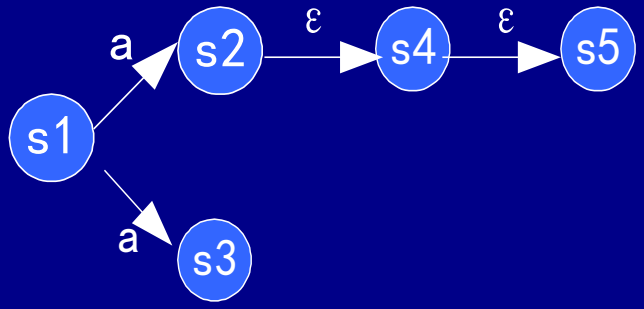
限制1：没有ε状态转移

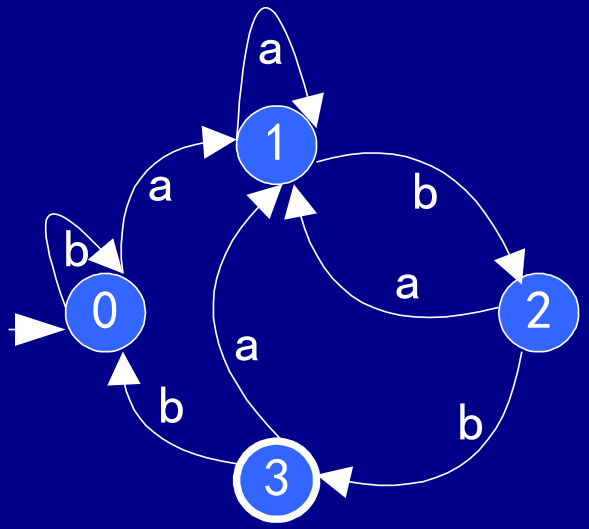
限制2：同一状态下没有重复字符的状态转移

例2.10 正规式(a|b)*abb的DFA和识别输入序列abb和abab：

识别abb：0a1b2b3，状态，接受

识别abab：0a1b2a1b2，？





	a	b
0	1	0
1	1	2
2	1	3
3	1	0

将在DFA上识别输入序列的过程形式化为算法，该算法被称为模拟器（模拟DFA的行为）或驱动器（用DFA的数据驱动分析动作）。

算法与DFA一起，即构成识别记号的词法分析器的核心。它的最大特点是算法与模式无关，仅DFA与模式相关。

算法2.1 模拟DFA

输入　DFA D和输入字符串x(eof)。D的初态为s0，终态集为F。

输出　若D接受x，回答“yes”，否则回答“no”。

方法　用下述过程识别x：

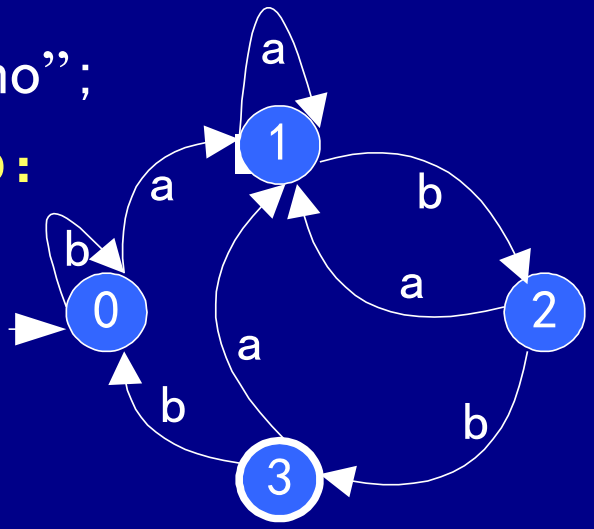
```
s:=s0;   ch:=nextchar;        -- 初值
while  ch≠eof                 -- 循环
loop   s:=move(s,ch);   ch:=nextchar;
end    loop;
if     s is in F              -- 返回
then return “yes”; else return “no”;
end if;
```

用算法2.1识别abb：

1. s=0, ch=a
2. s=1, ch=b
3. s=2, ch=b
4. s=3, ch=eof
5. yes

用算法2.1识别abab：

1. s=0, ch=a
2. s=1, ch=b
3. s=2, ch=a
4. s=1, ch=b
5. s=2, ch=eof
6. no



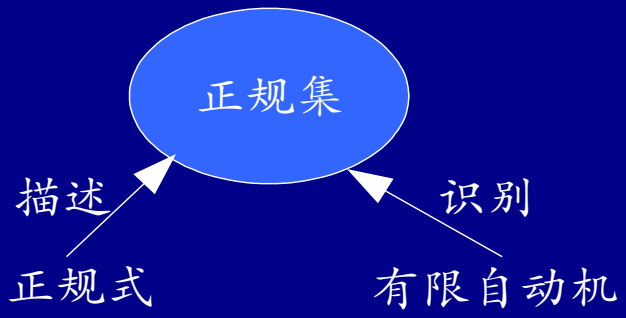
2.3.3 有限自动机的等价

定义2.6 若有限自动机M和M'识别同一正规集，则称M和M'是等价的，记为M=M'。 ■

特别提示：

正规式与有限自动机从两个侧面表示正规集。正规式是描述，自动机是识别。

因此，当它们表示相同集合时，均存在等价的问题。



想一想，有几种可能的等价？