编译系统

# 第三章
# 词法分析

哈尔滨工业大学 陈鄞

# 第3讲（词法分析）要点

- 词法分析的任务：识别单词
  - 单词的描述
  - 单词的识别

# 1、单词的描述

- RG（正则文法）：$G$
  - $G \rightarrow L(G)$ 
- RE（正则表达式）：$r$　更直观、更紧凑
  - $r \rightarrow L(r)$
- RD（正则定义）：$d$　给RE命名
  - 形式：$d \rightarrow r$

# 2、单词的识别

- 理论基础：有穷自动机（FA）

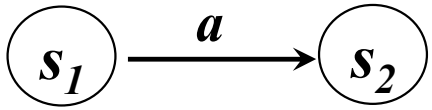
$$M=(S,\ \Sigma,\ \delta,\ s_0,\ F)$$

  - $S$：状态集合
  - $\Sigma$：输入符号集合
  - $\delta$：转换函数，$S\times\Sigma\rightarrow 2^S$
  - $s_0$：初始状态，$s_0\in S$
  - $F$：终止状态集合，$F\subseteq S$

# ①FA的表示

$M=(S,\Sigma,\delta,s_0,F)$

- 转换图
  - 节点　　状态
  - 边（有向）　　输入符号
- 转换表



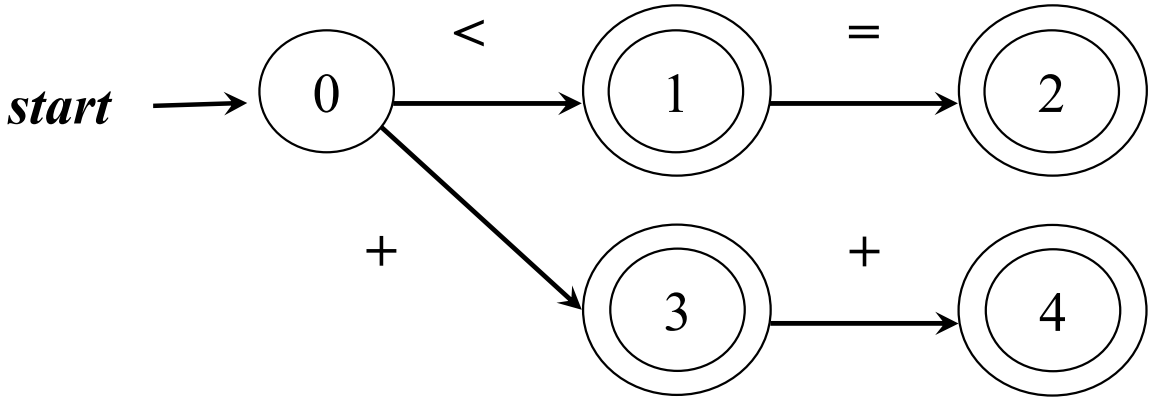

| $S$ \ $\Sigma$ | $a_1$ | $a_2$ | … | $a_n$ |
|---|---|---|---|---|
| $s_1$ | | | | |
| $s_2$ | | | | |
| … | | | | |
| $s_m$ | | | | |

## ②FA定义（接收/识别）的语言

- $M \rightarrow L(M)$
  - 给定输入串$x$，如果存在一个对应于串$x$的从初始状态到某个终止状态的转换序列，则称串$x$被该$FA$接收
  - 由一个有穷自动机$M$接收的所有串构成的集合称为是该$FA$定义（或接收）的语言，记为$L(M)$

# ②FA定义（接收/识别）的语言

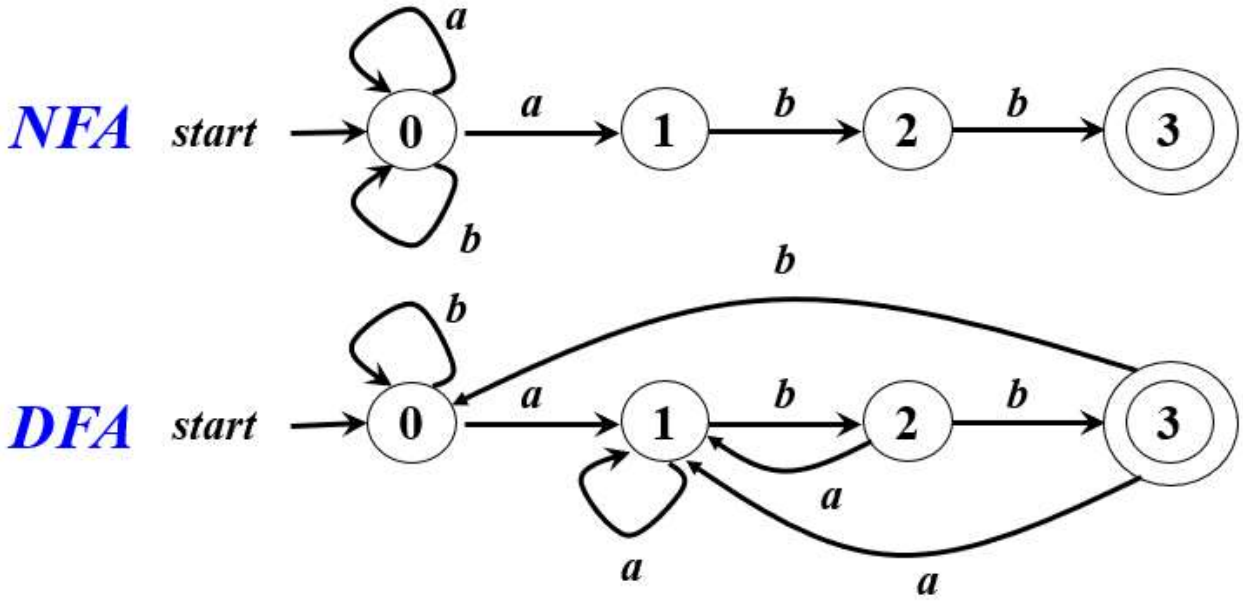
- $M \rightarrow L(M)$
- 最长子串匹配原则
  - 当输入串的多个前缀与一个或多个模式匹配时，总是选择最长的前缀进行匹配




  - 在到达某个终态之后，只要输入带上还有符号，DFA就继续前进，以便寻找尽可能长的匹配

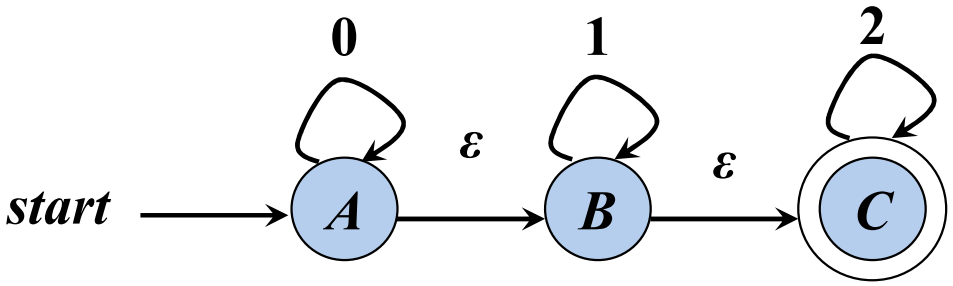
## ③FA的分类　　$M=(S,\Sigma,\delta,s_0,F)$

- *DFA*（确定的有穷自动机）　更适用
  - $\delta$：$S\times\Sigma\rightarrow S$
- *NFA*（非确定的有穷自动机）　更直观
  - $\delta$：$S\times\Sigma\rightarrow 2^S$



# ③FA的分类　　$M=(S,\Sigma,\delta,s_0,F)$

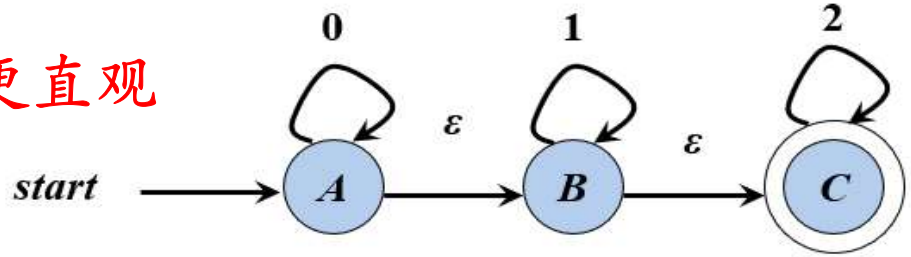
- *DFA*（确定的有穷自动机）
  - $\delta$: $S\times\Sigma\rightarrow S$
- *NFA*（非确定的有穷自动机）
  - $\delta$: $S\times\Sigma\rightarrow 2^S$　　不带“ε-边”
  - $\delta$: $S\times(\Sigma\cup\{\varepsilon\})\rightarrow 2^S$　带“ε-边”



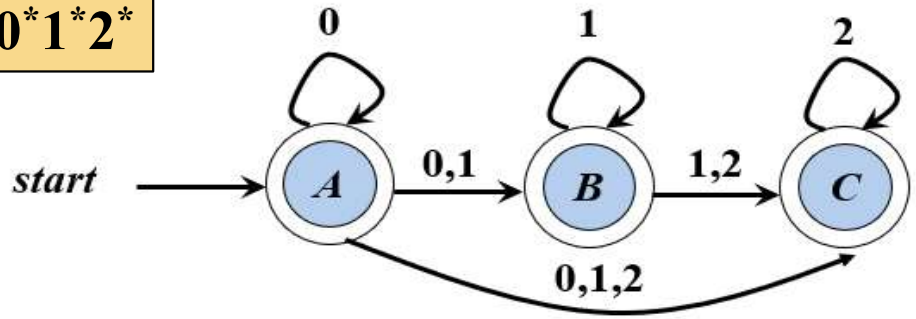

$r=0^*1^*2^*$

# ③FA的分类　　$M=(S,\Sigma,\delta,s_0,F)$

- *DFA*（确定的有穷自动机）
  - $\delta$：$S\times\Sigma\to S$
- *NFA*（非确定的有穷自动机）
  - $\delta$：$S\times\Sigma\to 2^S$　　不带“ε-边”
  - $\delta$：$S\times(\Sigma\cup\{\varepsilon\})\to 2^S$　　带“ε-边”更直观




$r=0^*1^*2^*$



# ④DFA的实现　$M=(S,\Sigma,\delta,s_0,F)$

- 函数体由一个循环构成

```
s = s0;
c = nextChar();
while(c! = eof){
        s = move(s, c);
        c = nextChar();
}
if (s在F中) return "yes";
else return "no";
```

表驱动

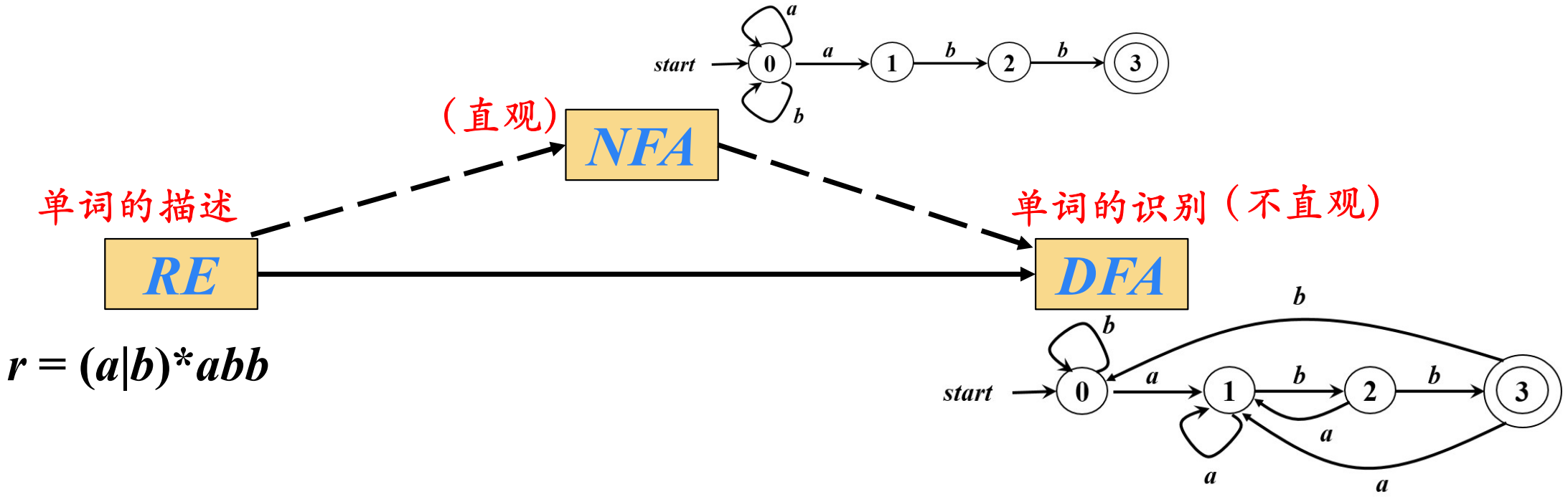
| $s$ \ $\Sigma$ | $a_1$ | $a_1$ | … | $a_n$ |
|---|---|---|---|---|
| $s_1$ | | | | |
| $s_2$ | | | | |
| … | | | | |
| $s_m$ | | | | |

# ⑤FA与RE的等价性

- $RE \Leftrightarrow FA$

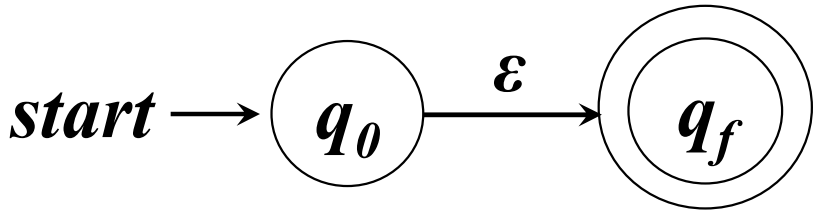
  $r \Leftrightarrow M$

- 词法分析器（*scanner*）的实现思路

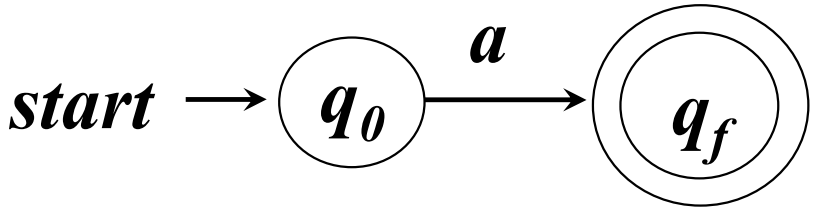


# ⑥从RE 到NFA

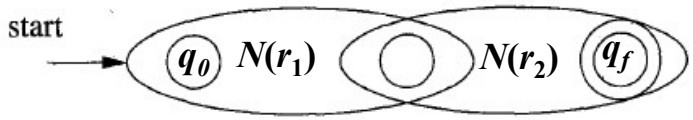
- ε对应的NFA



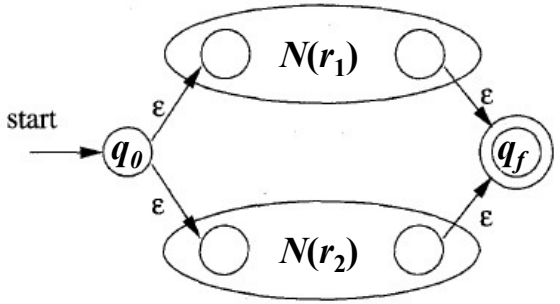

- 字母表Σ中符号a对应的NFA



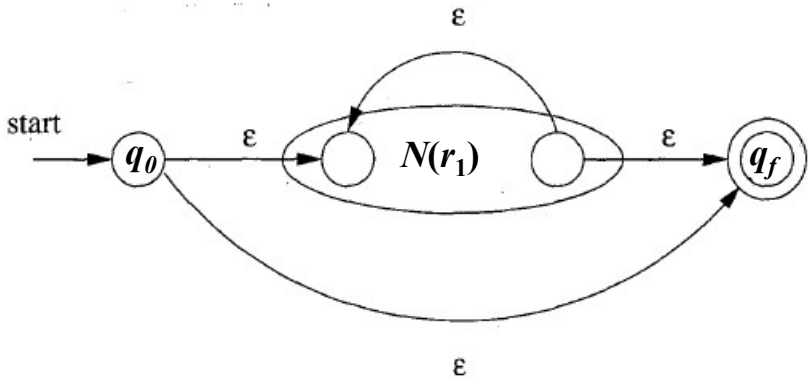
- 假设正则表达式$r_1$和$r_2$对应的$NFA$分别为$N(r_1)$和$N(r_2)$
  - $r = r_1 r_2$对应的$NFA$



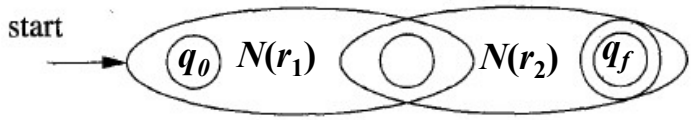

  - $r = r_1 | r_2$对应的$NFA$



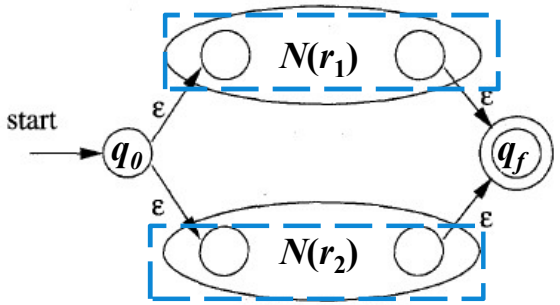

  - $r = (r_1)^*$对应的$NFA$



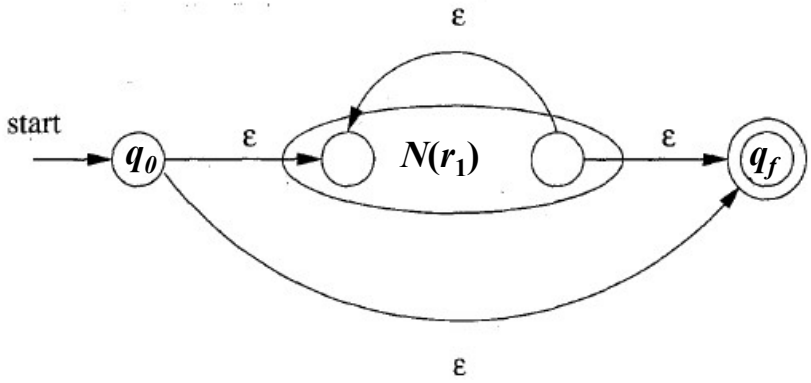
- 假设正则表达式$r_1$和$r_2$对应的$NFA$分别为$N(r_1)$和$N(r_2)$
  - $r = r_1 r_2$对应的$NFA$



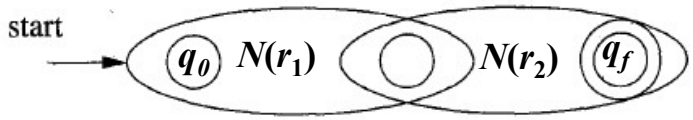

  - $r = r_1 | r_2$对应的$NFA$



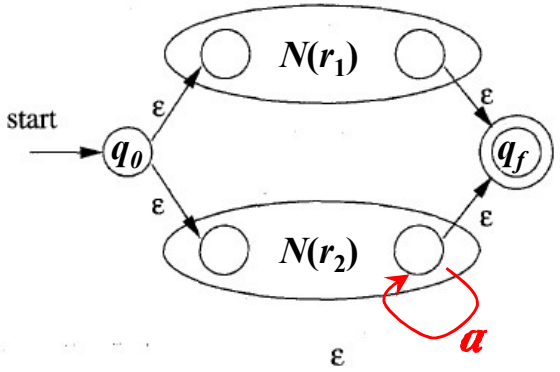

  - $r = (r_1)^*$对应的$NFA$



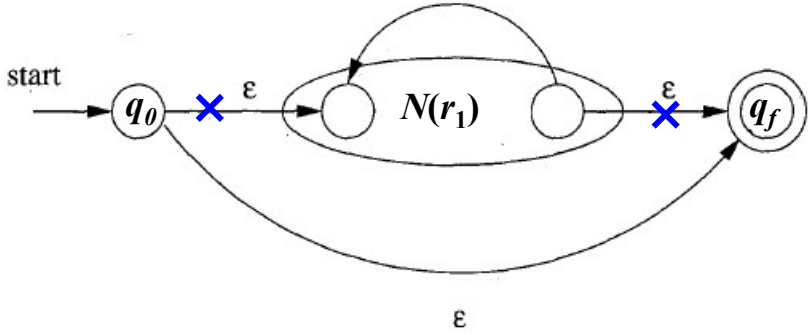
- 假设正则表达式$r_1$和$r_2$对应的$NFA$分别为$N(r_1)$和$N(r_2)$
  - $r = r_1 r_2$对应的$NFA$



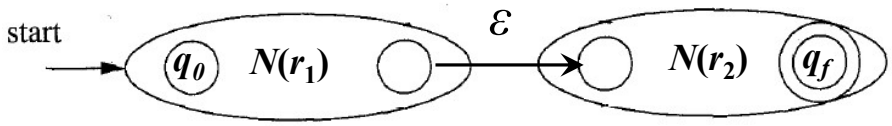

  - $r = r_1 | r_2$对应的$NFA$



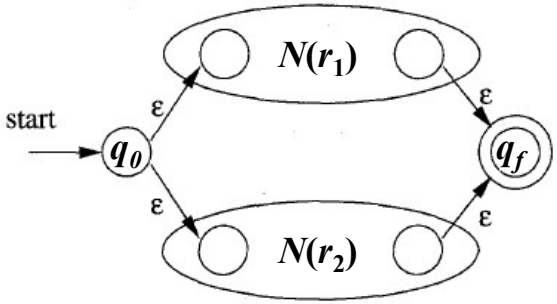

  - $r = (r_1)^*$对应的$NFA$



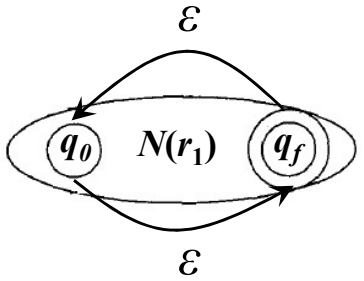
- 假设正则表达式$r_1$和$r_2$对应的$NFA$分别为$N(r_1)$和$N(r_2)$
  - $r = r_1 r_2$对应的$NFA$



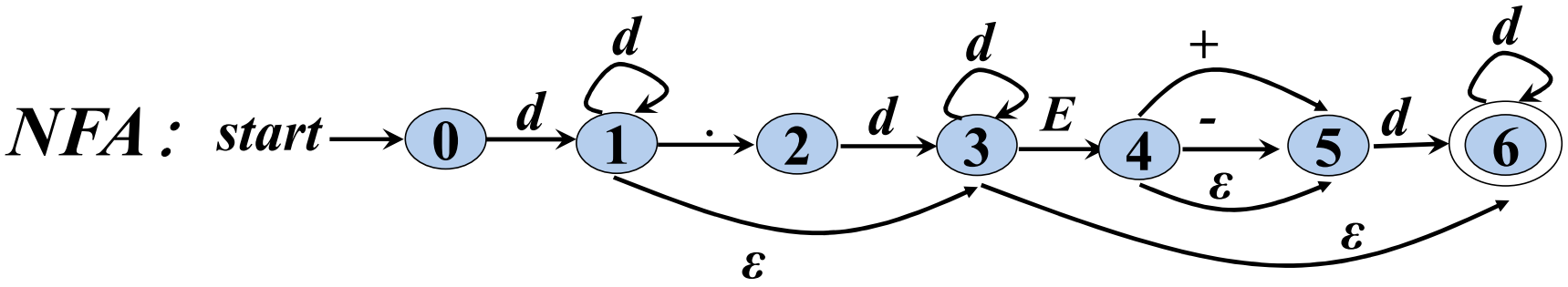

  - $r = r_1 | r_2$对应的$NFA$

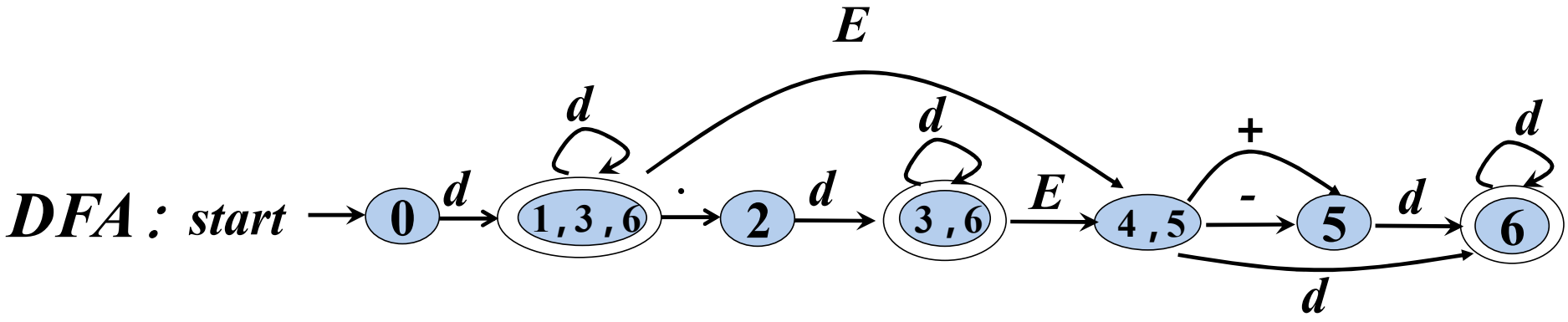



  - $r = (r_1)^*$对应的$NFA$



# ⑦从NFA到DFA：子集构造法



# 结束