课外材料(未修过相关课程者参考)

✧ 文法/正规式/有限自动机 — 基础知识

- 形式语言概念
- 正规语言及其描述
- 上下文无关文法及语言

形式语言概念

✧ 字母表（*Alphabet*）

– 形式符号的集合

– 常用 Σ 表示

– 举例

英文字母表 { *a*, *b*, …, *z*, *A*, *B* , …, *Z* }
汉字表 { …, 自, …, 动 , …, 机, … }
化学元素表 { H, He, Li, …,Une }
Σ = { *a*, *n*, *y*, 任,意 }

形式语言概念

✧ 字符串 （*String*）

- 字母表Σ上的一个字符串（串），或称为字（*word*），为Σ中字符构成的一个有限序列。空串（*empty string*），常用 ε 表示，不包含任何字符。
- 字符串 *w* 的长度，记为 $|w|$，是包含在 *w* 中字符的个数

举例 $|\varepsilon| = 0$, $|bbaba| = 5$

✧ 字符串的连接（concatenation）运算

– 设 x, y 为串，且 $x = a_1a_2 \ldots a_m$, $y = b_1b_2 \ldots b_n$，则 x 与 y 的连接

$$x\,y = a_1a_2 \ldots a_m b_1b_2 \ldots b_n$$

– 连接运算的性质

$$(x\,y)\,z = x\,(y\,z)$$

$$x = x\,\varepsilon = x$$

$$|x\,y| = |x| + |y|$$

形式语言概念

✧ 字母表上的运算

– 幂运算

设 Σ 为字母表，n 为任意自然数，

定义（1）$\Sigma^0 = \{ \varepsilon \}$

（2）设 $x \in \Sigma^{n-1}$，$a \in \Sigma$，则 $a\,x \in \Sigma^n$

（3）Σ^n 中的元素只能由（1）和（2）生成

– * 闭包 $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$

– + 闭包 $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

形式语言概念

✧ 语言 （*Languages*）

- 概念 设 Σ 为字母表，则任何集合 $L \subseteq \Sigma^*$ 是字母表 Σ 上的一个语言
- 举例

  { …, *English*, …, *words* , … }

  { *any*, 任意 }
- 比较 空语言 Φ 与仅含空字的语言 { ε }

形式语言概念

✧ 语言上的运算

– 两个语言 L 和 M 的联合（union）

$$L \cup M = \{\, w \mid w \in L \vee w \in M \,\}$$

– 两个语言 L 和 M 的连接（concatenation）

$$L \cdot M = \{\, w_1 w_2 \mid w_1 \in L \wedge w_2 \in M \,\}$$

– 语言 L 的闭包（closure）

$L^* = L^0 \cup L^1 \cup L^2 \cup \ldots = \cup_{i \geq 0} L^i$，其中

$L^0 = \{\varepsilon\}$， $L^1 = L$, $L^2 = LL$, …

$L^n = L^{n-1}L$

形式语言概念

✧ 程序设计语言

- C 源程序的集合用 Σ^* 表示（Σ 为 ASCII字符集）
- *L* 表示由满足 C 语言词法规则的单词构成的语言
- *M*表示由满足 C 语言语法规则的源程序构成的语言
- *S* 表示由正确的C语言源程序构成的语言
- 四者之间的关系

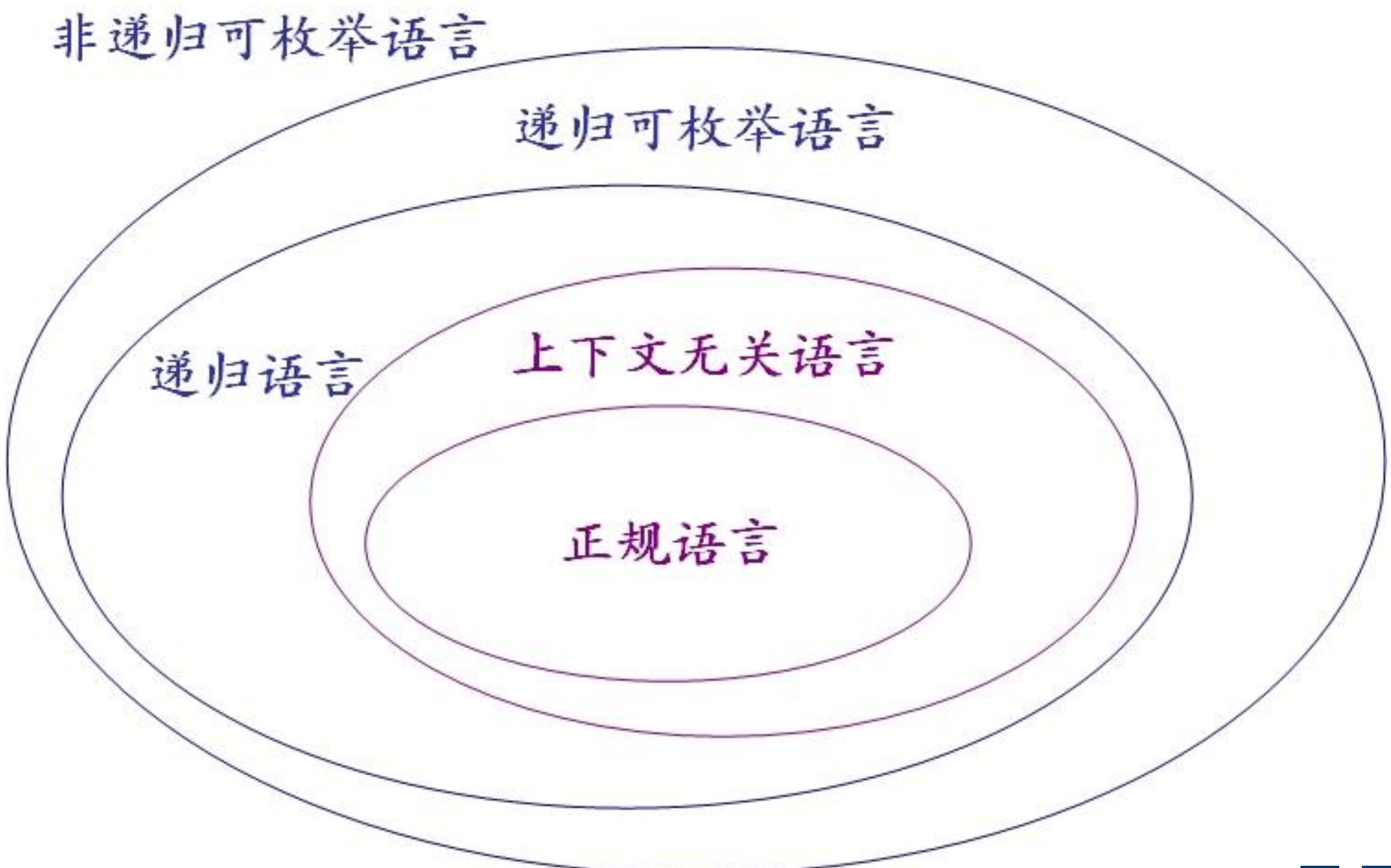
  $L \subseteq \Sigma^*$，$M \subseteq \Sigma^*$，$S \subseteq \Sigma^*$

  $M \subseteq L^*$, $S \subseteq L^*$

  $S \subseteq M$

形式语言概念

✧ 几个重要的形式语言类



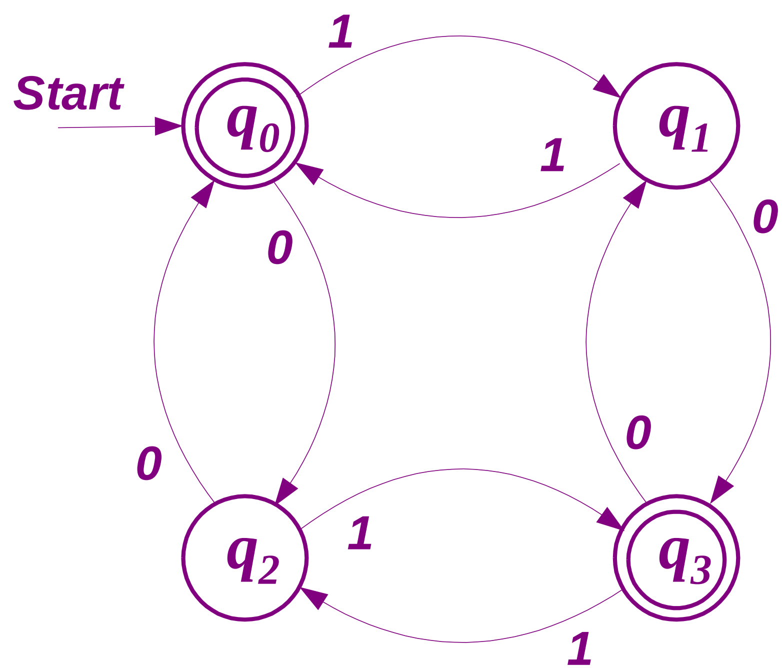
正规语言及其描述

✧ 描述正规语言的形式工具

- 3 型（正规）文法
- 有限自动机
- 正规表达式

✧ 有限自动机的五要素

– 有限状态集
– 有限输入符号集
– 转移函数
– 一个开始状态
– 一个终态集合

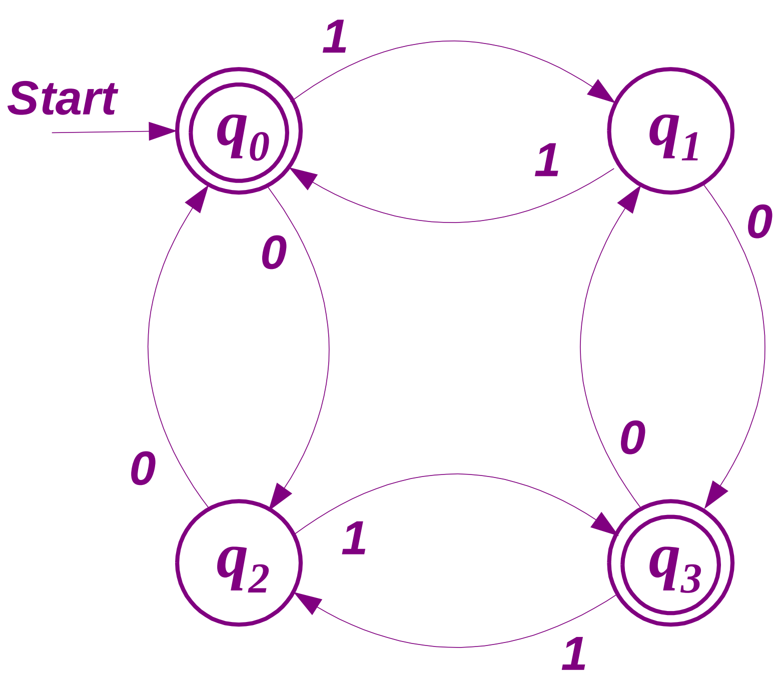


✧ 确定有限自动机的形式定义

一个确定有限状态自动机 ***DFA*** ***(deterministic finite automata)*** 是一个五元组 $A = (Q, \Sigma, \delta, q_0, F)$.

– 有限状态集
– 有限输入符号集
– 转移函数 $\delta: Q \times \Sigma \to Q$
– 一个开始状态 $q_0 \in Q$
– 一个终态集合 $F \subseteq Q$

✧ 转移图表示的 *DFA*



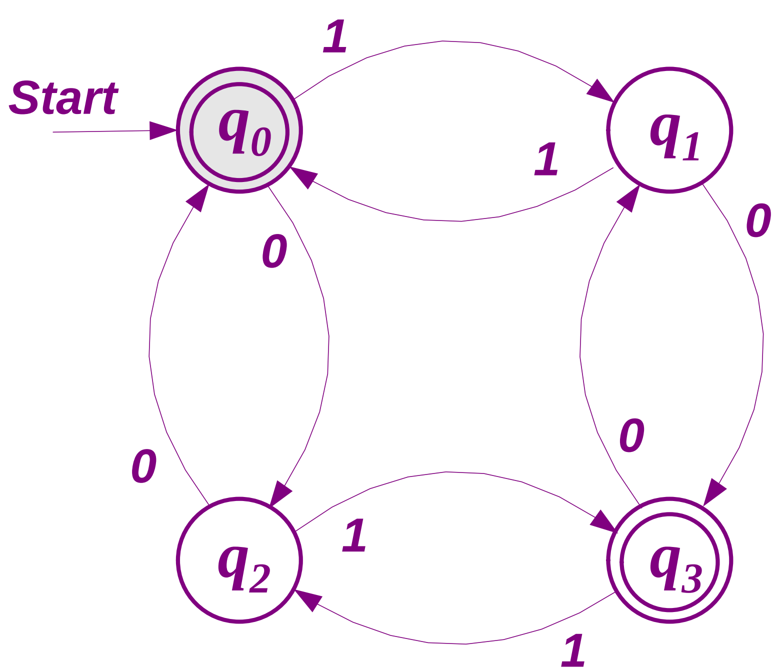
- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- $\delta(q_0,0) = q_2,\ \delta(q_0,1) = q_1$
  $\delta(q_1,0) = q_3,\ \delta(q_1,1) = q_0$
  $\delta(q_2,0) = q_0,\ \delta(q_2,1) = q_3$
  $\delta(q_3,0) = q_1,\ \delta(q_3,1) = q_2$
- q_0
- $F = \{q_0, q_3\}$

✧ 转移表表示的 DFA

	0	1
$\rightarrow *q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
$*q_3$	q_1	q_2

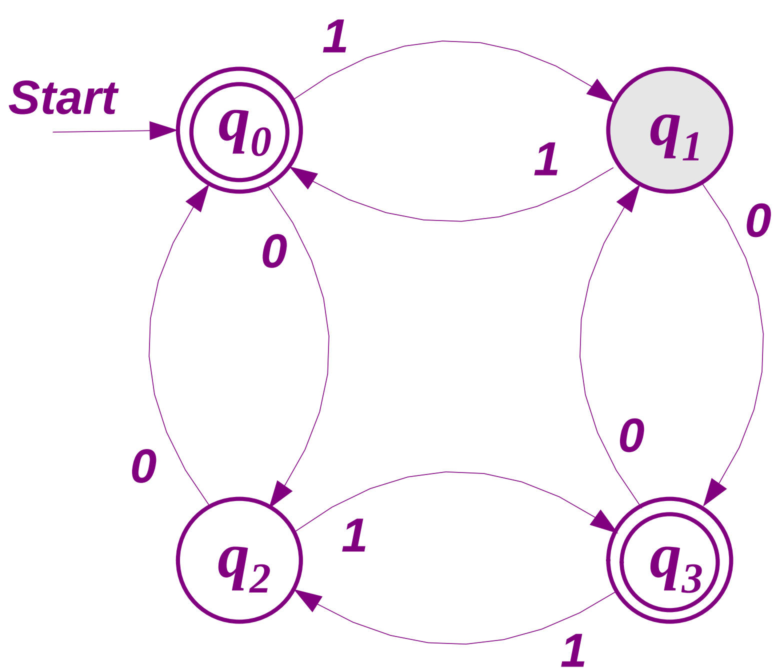
- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- $\delta(q_0,0) = q_2,\ \delta(q_0,1) = q_1$
  $\delta(q_1,0) = q_3,\ \delta(q_1,1) = q_0$
  $\delta(q_2,0) = q_0,\ \delta(q_2,1) = q_3$
  $\delta(q_3,0) = q_1,\ \delta(q_3,1) = q_2$
- q_0
- $F = \{q_0, q_3\}$

✧ DFA 如何接受输入符号串



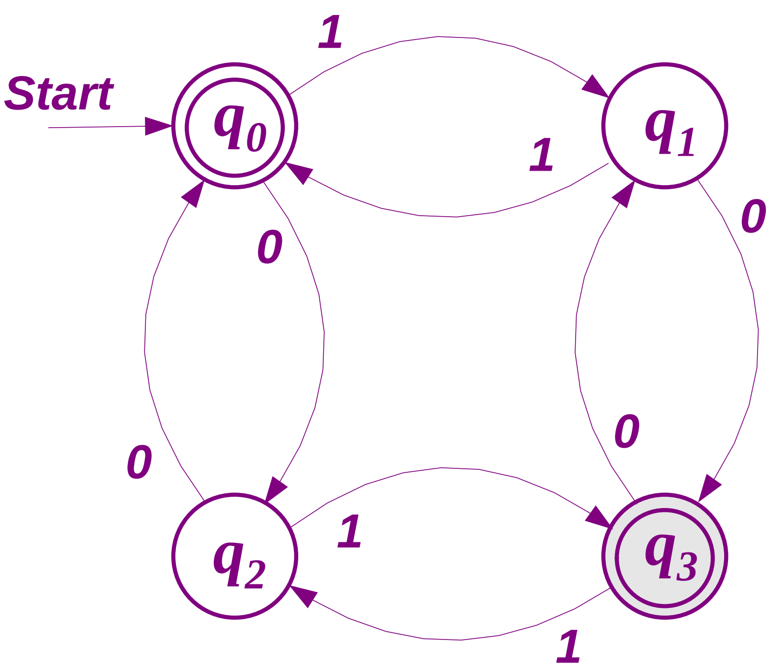
1	0	0	1	0	1

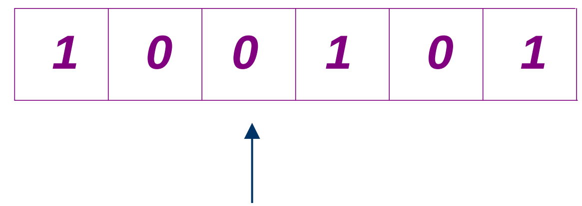
✧ DFA 如何接受输入符号串



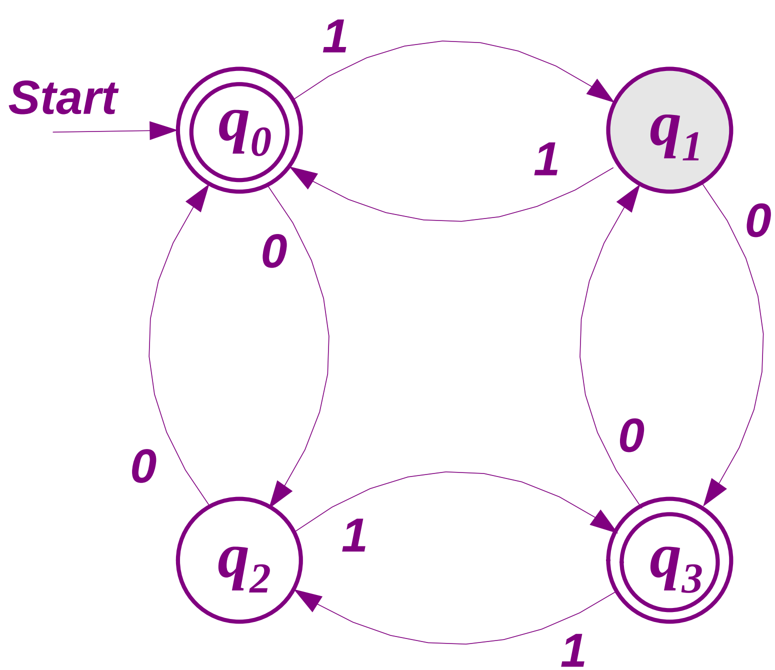
1	0	0	1	0	1

✧ DFA 如何接受输入符号串



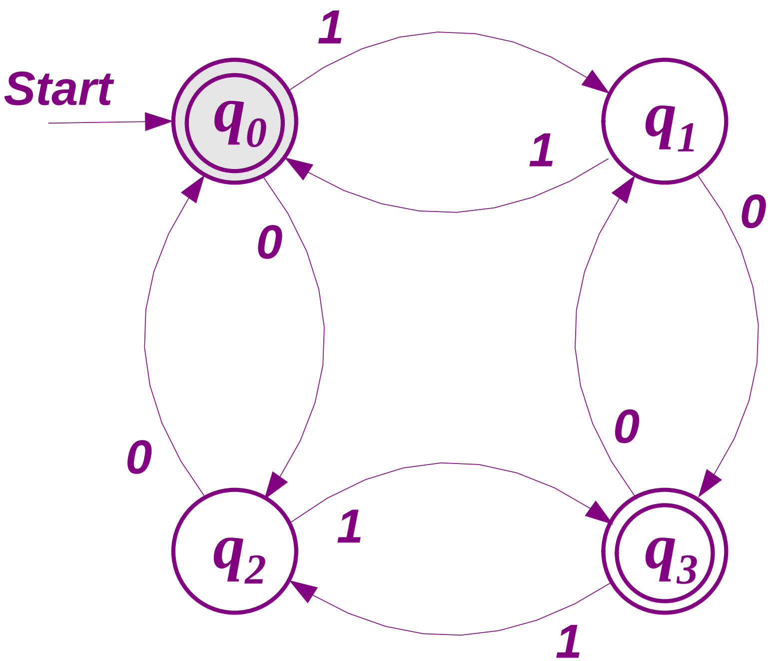


✧ DFA 如何接受输入符号串



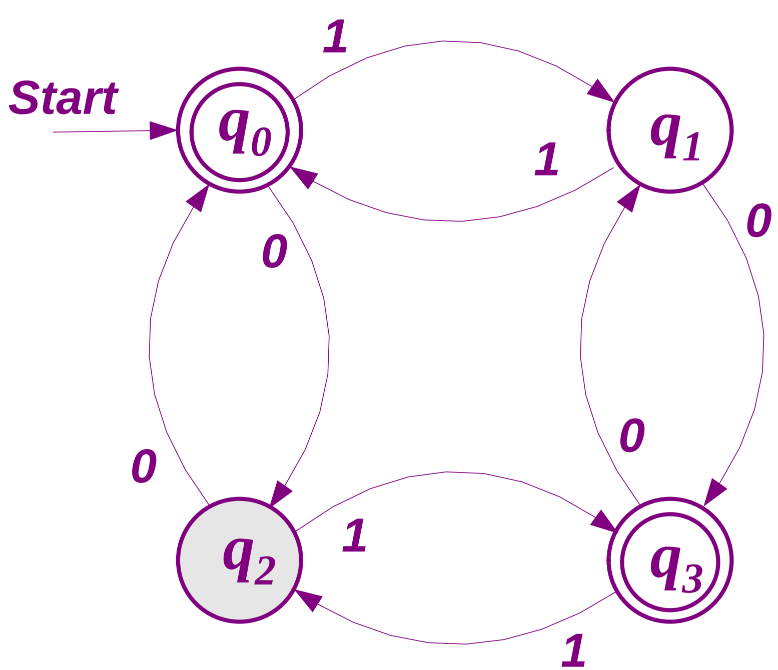
1	0	0	1	0	1

✧ DFA 如何接受输入符号串



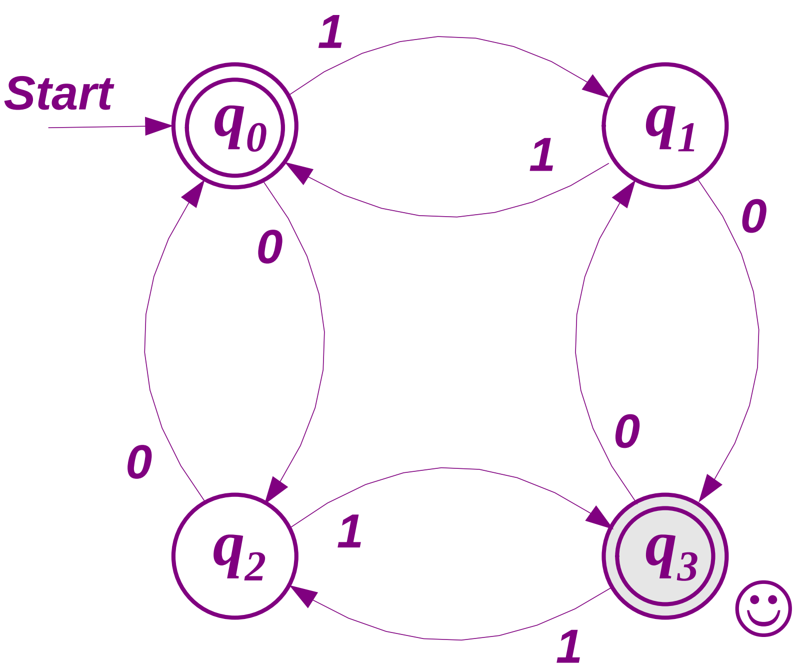
1	0	0	1	0	1

✧ DFA 如何接受输入符号串



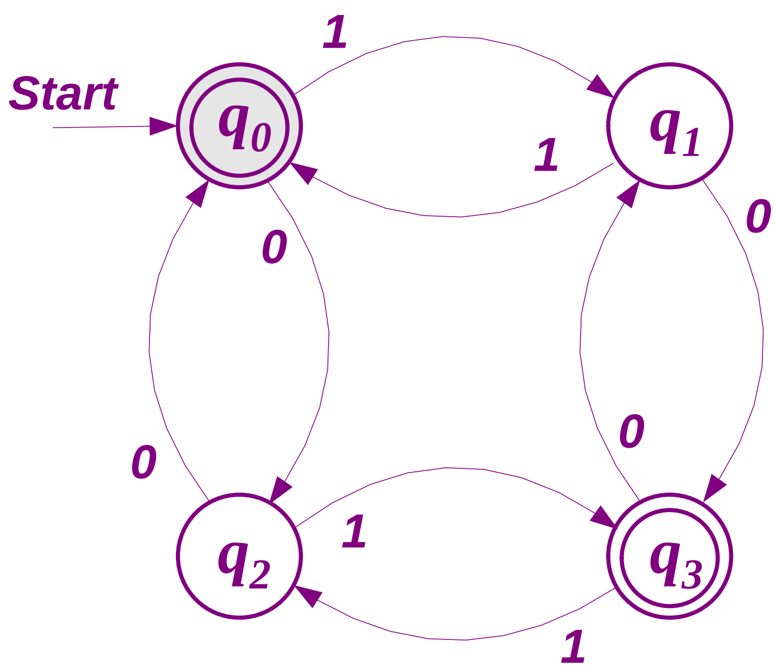
1	0	0	1	0	1

✧ DFA 如何接受输入符号串



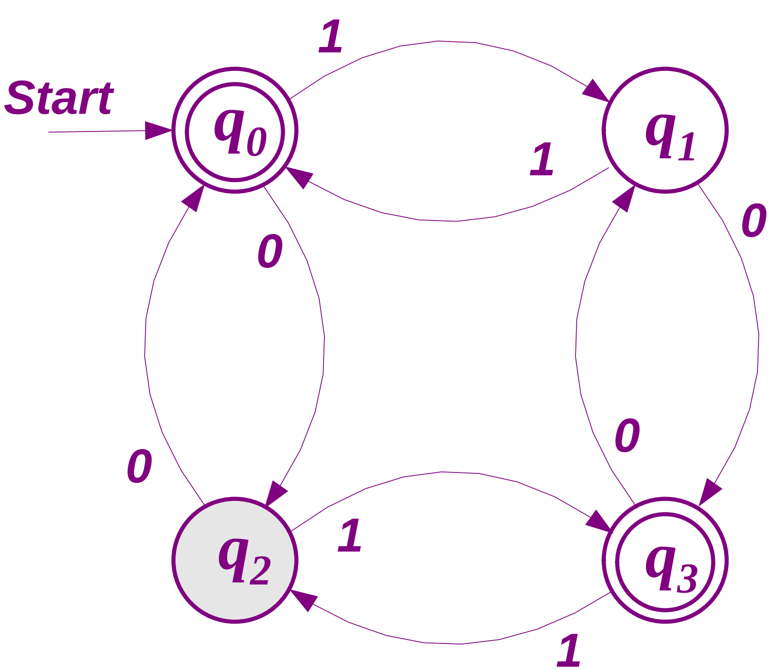
1	0	0	1	0	1

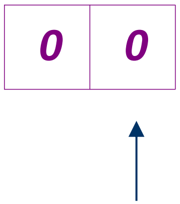
✧ DFA 如何接受输入符号串



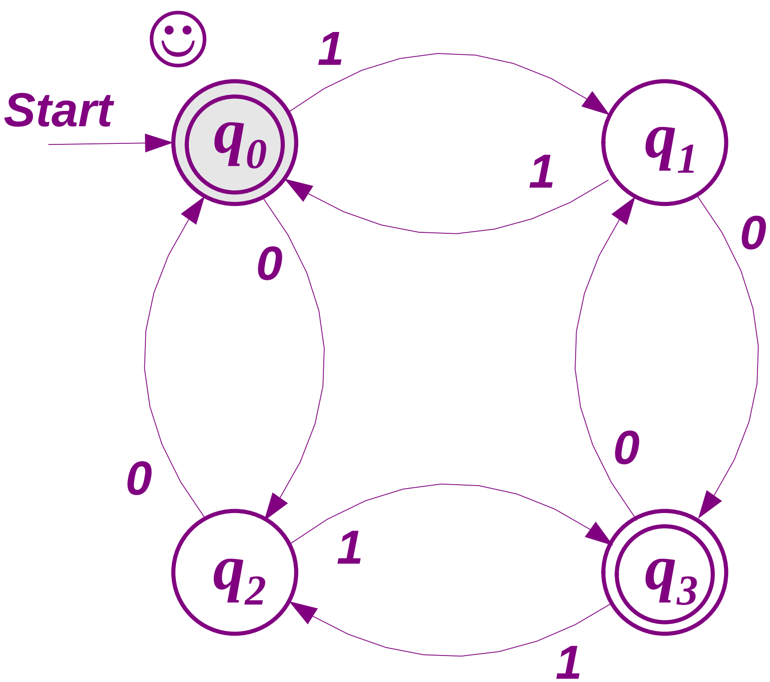
0	0

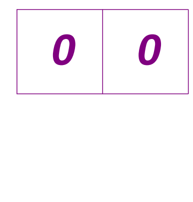
✧ DFA 如何接受输入符号串



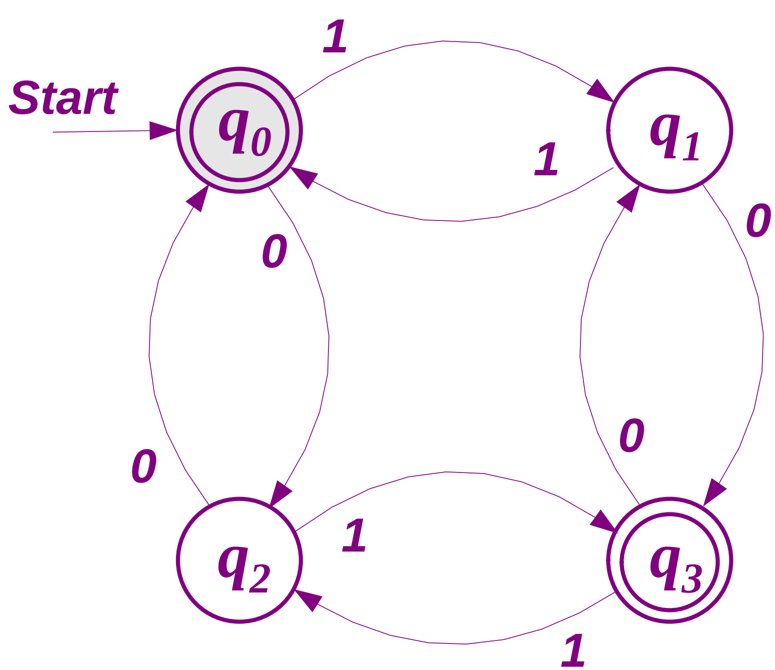


✧ DFA 如何接受输入符号串



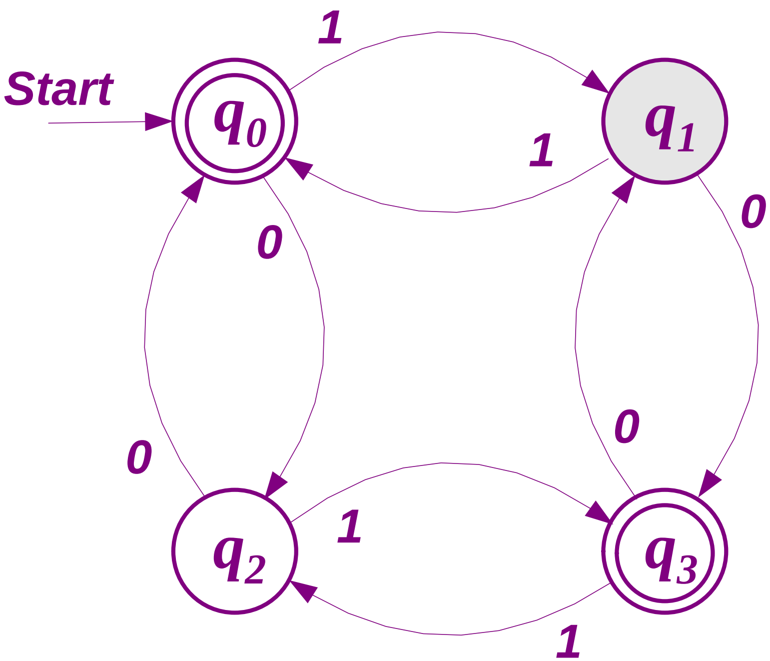


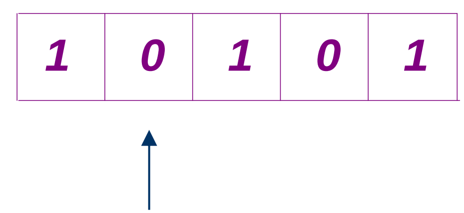
✧ DFA 如何接受输入符号串



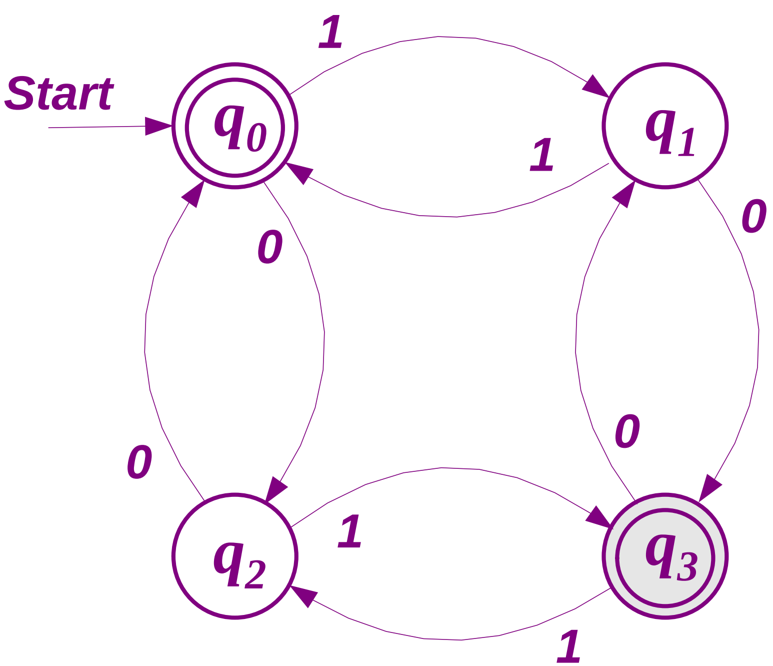
1	0	1	0	1

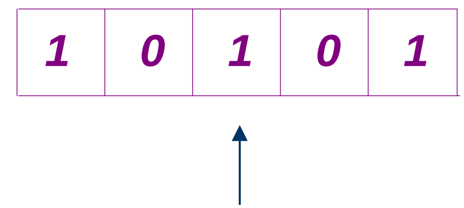
✧ DFA 如何接受输入符号串



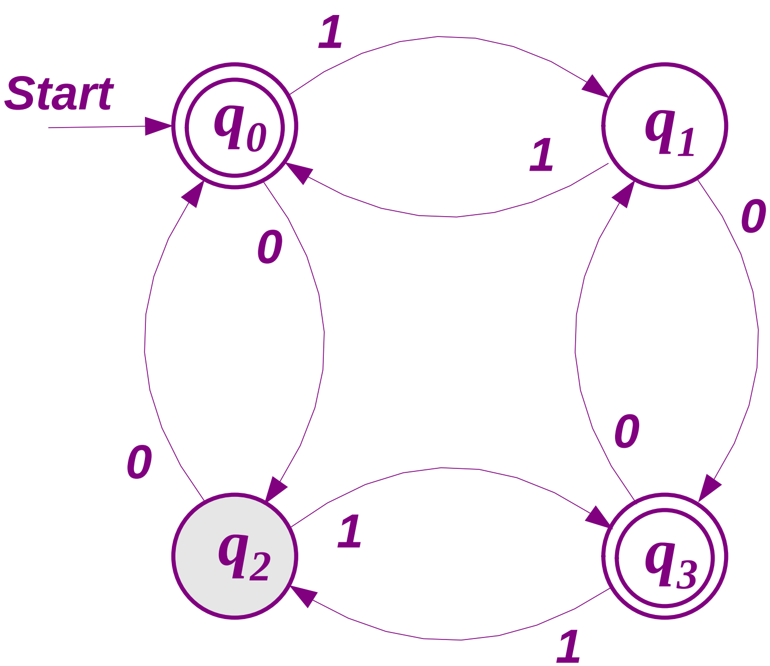


✧ DFA 如何接受输入符号串



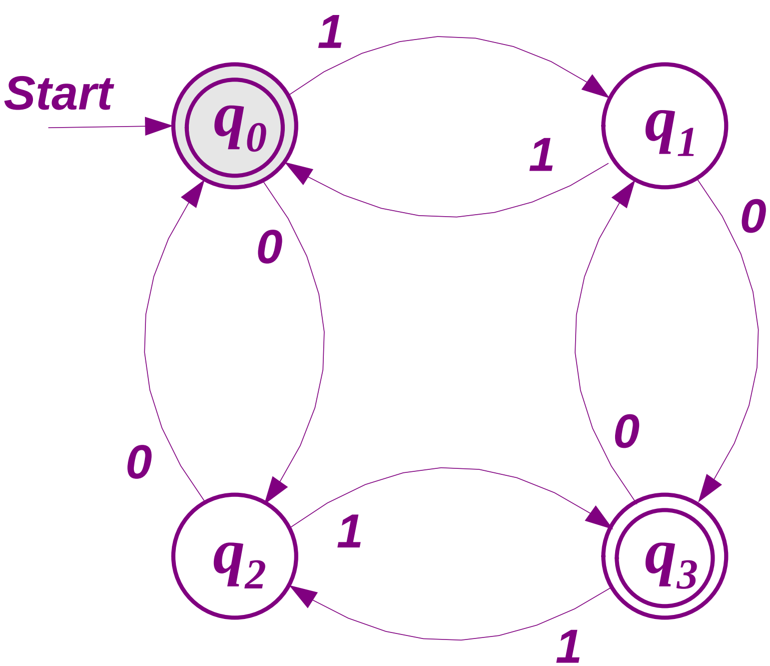


✧ DFA 如何接受输入符号串



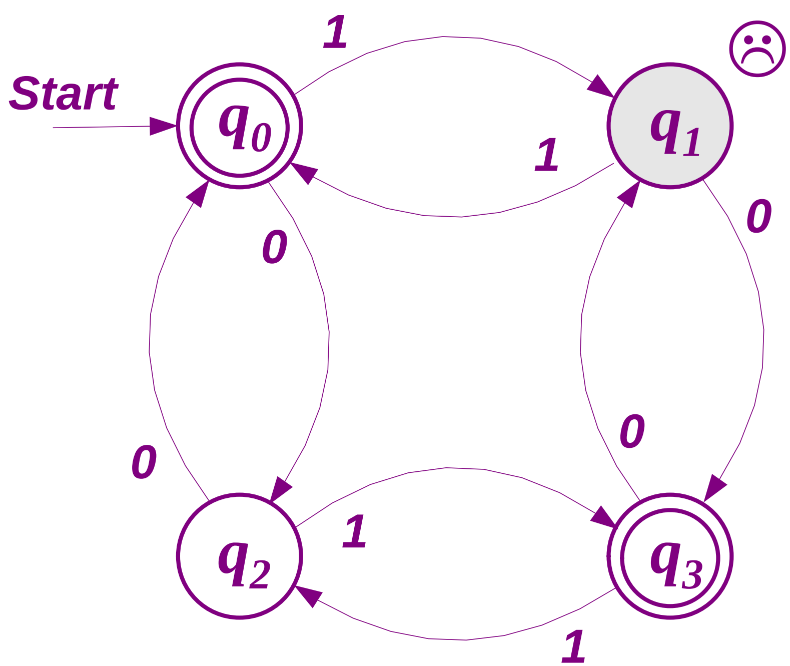
1	0	1	0	1

✧ DFA 如何接受输入符号串



1	0	1	0	1

✧ DFA 如何接受输入符号串



1	0	1	0	1

✧ 扩展转移函数适合于输入字符串

– 设一个 DFA $A = (Q, \Sigma, \delta, q_0, F)$

$\delta: Q \times \Sigma \to Q$

– 扩充定义 $\delta': Q \times \Sigma^* \to Q$

对任何$q \in Q$，定义：

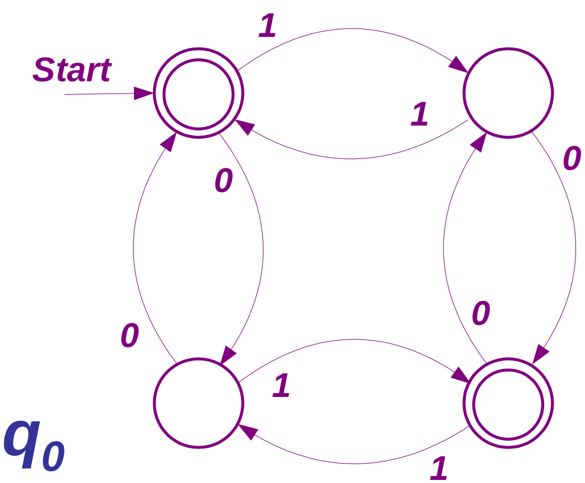
1 $\delta'(q, \varepsilon) = q$

2 若$w = xa$, 其中 $x \in \Sigma^*$, $a \in \Sigma$, 则

$$\delta'(q, w) = \delta(\delta'(q, x), a)$$

✧ 扩展转移函数适合于输入字符串

	0	1
$\rightarrow *q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
$*q_3$	q_1	q_2



– 举例

$$\delta'(q_0, \varepsilon) = q_0$$

$$\delta'(q_0, 0) = \delta(q_0, 0) = q_2$$

$$\delta'(q_0, 00) = \delta(q_2, 0) = q_0$$

$$\delta'(q_0, 001) = \delta(q_0, 1) = q_1$$

$$\delta'(q_0, 0010) = \delta(q_1, 0) = q_3$$

✧ DFA 的语言

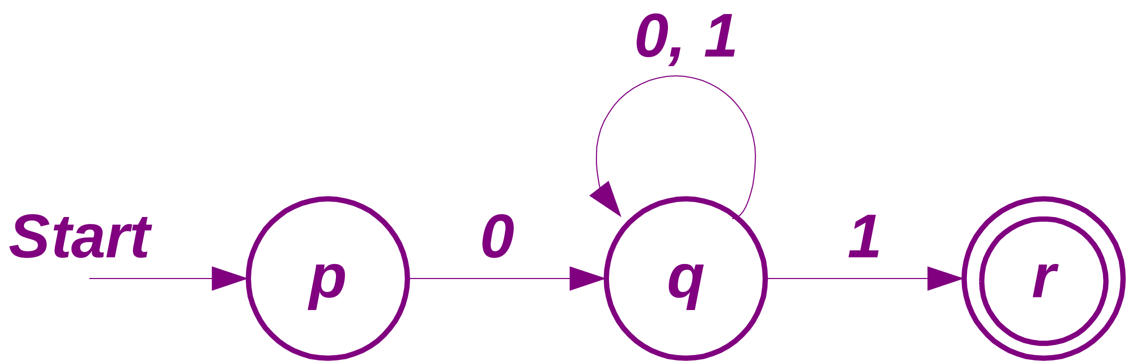
- 设一个 DFA $A = (Q, \Sigma, \delta, q_0, F)$
- 定义 A 的语言：

$$L(A) = \{ w \mid \delta'(q_0, w) \in F \}$$

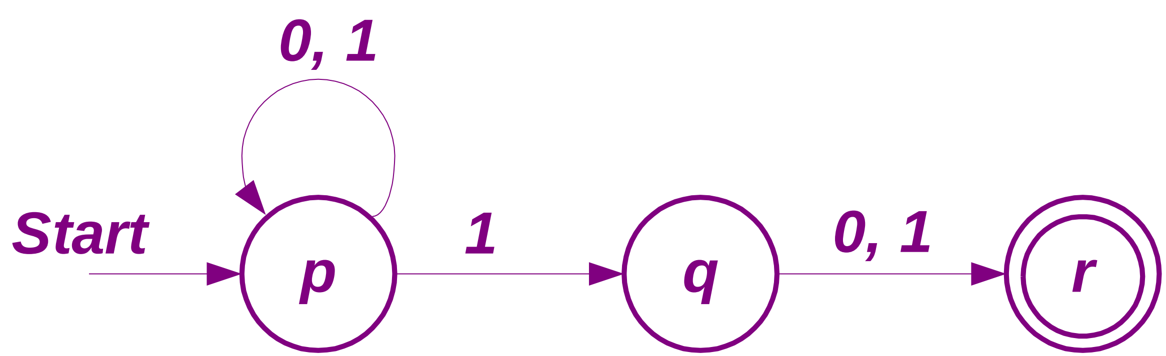
- 可以证明，如果存在一个 DFA $A = (Q, \Sigma, \delta, q_0, F)$，满足$L = L(A)$ ，则 L 是一个正规语言.

✧ 非确定有限自动机举例

(1)



(2)



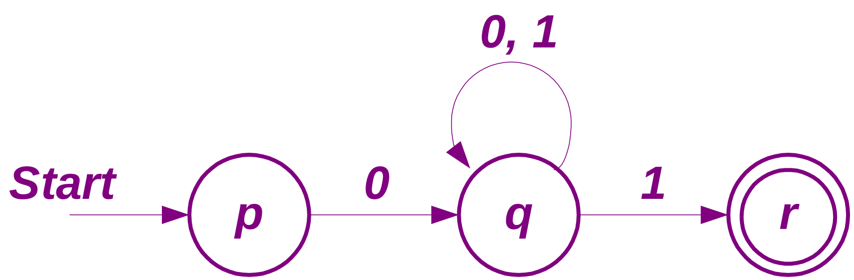
✧ 非确定有限自动机的形式定义

一个非确定有限状态自动机 ***NFA nondeterministic finite automata)*** 是一个五元组 $A = (Q, \Sigma, \delta, q_0, F)$.

- 有限状态集 — Q
- 有限输入符号集 — Σ
- 转移函数 — δ
- 一个开始状态 — q_0，$q_0 \in Q$
- 一个终态集合 — F，$F \subseteq Q$
- 与 ***DFA*** 唯一不同之处　$\delta: Q \times \Sigma \to 2^Q$

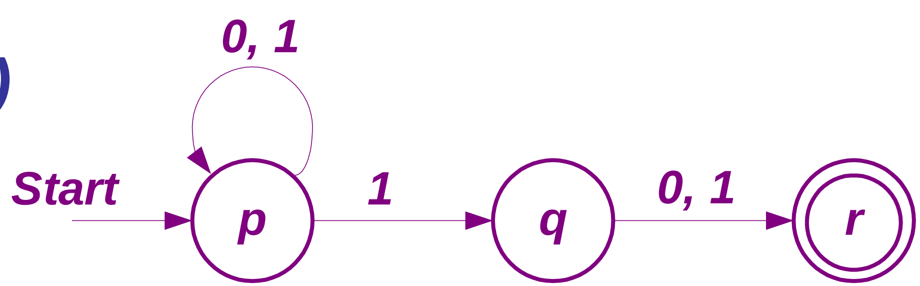
✧ 转移图和转移表表示的 *NFA*

(1)



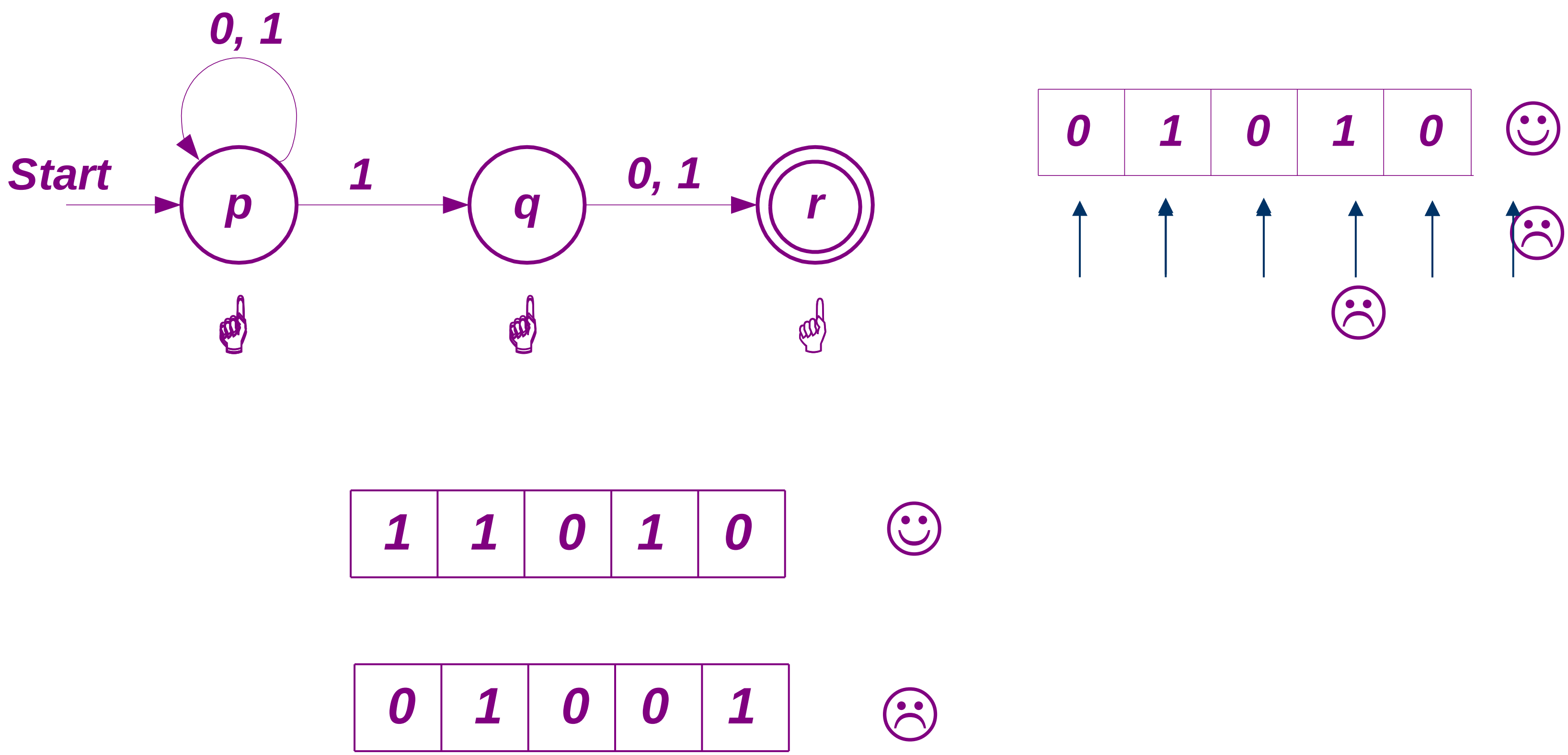
	0	1
→ p	{ q }	ϕ
q	{ q }	{ q, r }
* r	ϕ	ϕ

(2)



	0	1
→ p	{ p }	{ p, q }
q	{ r }	{ r }
* r	ϕ	ϕ

✧ NFA 如何接受输入符号串



✧ 扩展转移函数适合于输入字符串

- 设一个 NFA $A = (Q, \Sigma, \delta, q_0, F)$
- $\delta: Q \times \Sigma \rightarrow 2^Q$
- 扩充定义 δ': $Q \times \Sigma^* \rightarrow 2^Q$
- 对任何$q \in Q$，定义:

  1 $\delta'(q, \varepsilon) = \{ q \}$

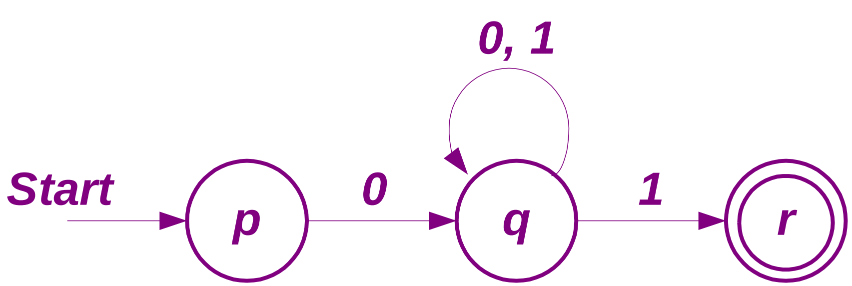
  2 若$w = xa$, 其中 $x \in \Sigma^*$, $a \in \Sigma$, 并且假设

  $\delta'(q, x) = \{ p_1, p_2, \ldots, p_k \}$， 则

$$\delta'(q, w) = \bigcup_{i=1}^{k} \delta(p_i, a)$$

✧ 扩展转移函数适合于输入字符串

	0	1
$\rightarrow p$	$\{q\}$	ϕ
q	$\{q\}$	$\{q, r\}$
$* r$	ϕ	ϕ



– 举例

$\delta'(p, \varepsilon) = \{p\}$

$\delta'(p, 0) = \{q\}$

$\delta'(p, 01) = \{q, r\}$

$\delta'(p, 010) = \{q\}$

$\delta'(p, 0100) = \{q\}$

$\delta'(p, 01001) = \{q, r\}$

✧ NFA 的语言

- 设一个 NFA $A = (Q, \Sigma, \delta, q_0, F)$
- 定义 A 的语言：

$$L(A) = \{ w \mid \delta'(q_0, w) \cap F \neq \phi \}$$

- 设 L 是 Σ 上的语言，如果存在一个 NFA $A = (Q, \Sigma, \delta, q_0, F)$，满足$L = L(A)$，则可以证明 L 也是一个正规语言.

✧ DFA 和 NFA 的等价性

– **定理:** L 是某个 DFA 的语言, 当且仅当 L 也是某个 NFA 的语言.

– **证明思路:** 分两步证明.

(1) 设 L 是某个 DFA D 的语言, 则存在一个 NFA N, 满足 $L(N) = L(D) = L$;

(2) 设 L 是某个 NFA N 的语言, 则存在一个 DFA D, 满足 $L(D) = L(N) = L$;

✧ 从 DFA 构造等价的 NFA

– 设 L 是某个 DFA $D = (Q, \Sigma, \delta_D, q_0, F)$ 的语言，则存在一个 NFA $N = (Q,\Sigma,\delta_N,q_0,F)$，其中$\delta_N$定义为

- 对$q \in Q$ 和$a \in \Sigma$，
  若 $\delta_D(q,a) = p$，则 $\delta_N(q,a) = \{p\}$.

可以证明：$L(N) = L(D) = L$.

✧ 从 *NFA* 构造等价的 *DFA*（子集构造法）

– 设 L 是某个 *NFA* $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ 的语言，则存在一个 *DFA*

$$D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D),$$

其中

- $Q_D = \{ S \mid S \subseteq Q_N \}$
- 对 $S \in Q_D$ 和 $a \in \Sigma$，

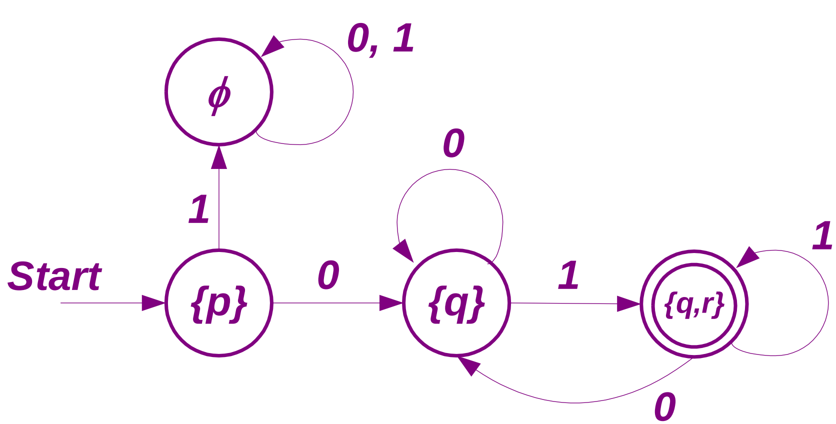
$$\delta_D(S, a) = \bigcup_{q \in S} \delta_N(q,a).$$

- $F_D = \{S \mid S \subseteq Q_N \wedge S \cap F_N \neq \phi \}$

可以证明：$L(D) = L(N) = L$.

✧ 子集构造法举例

	0	1
$\rightarrow p$	$\{q\}$	ϕ
q	$\{q\}$	$\{q, r\}$
$*\,r$	ϕ	ϕ

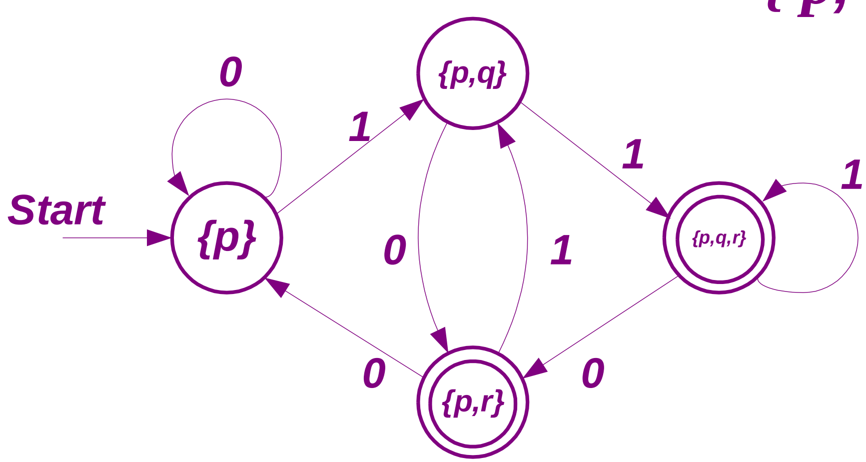


	0	1
ϕ	ϕ	ϕ
$\rightarrow \{p\}$	$\{q\}$	ϕ
$\{q\}$	$\{q\}$	$\{q, r\}$
$*\{r\}$	ϕ	ϕ
$\{p, q\}$	$\{q\}$	$\{q, r\}$
$*\{p, r\}$	$\{q\}$	ϕ
$*\{q, r\}$	$\{q\}$	$\{q, r\}$
$*\{p, q, r\}$	$\{q\}$	$\{q, r\}$

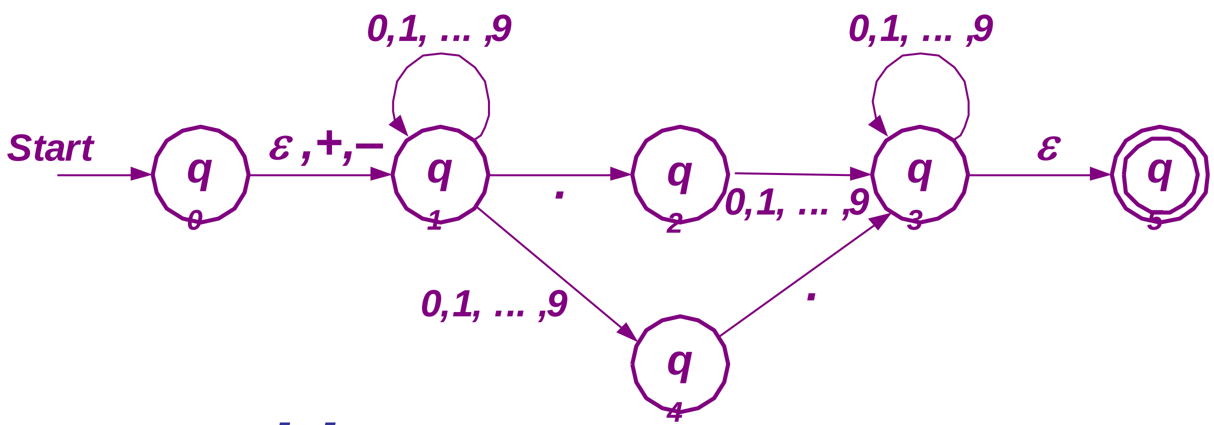
✧ 子集构造法举例

	0	1
→ p	{ p }	{ p, q }
q	{ r }	{ r }
* r	ϕ	ϕ

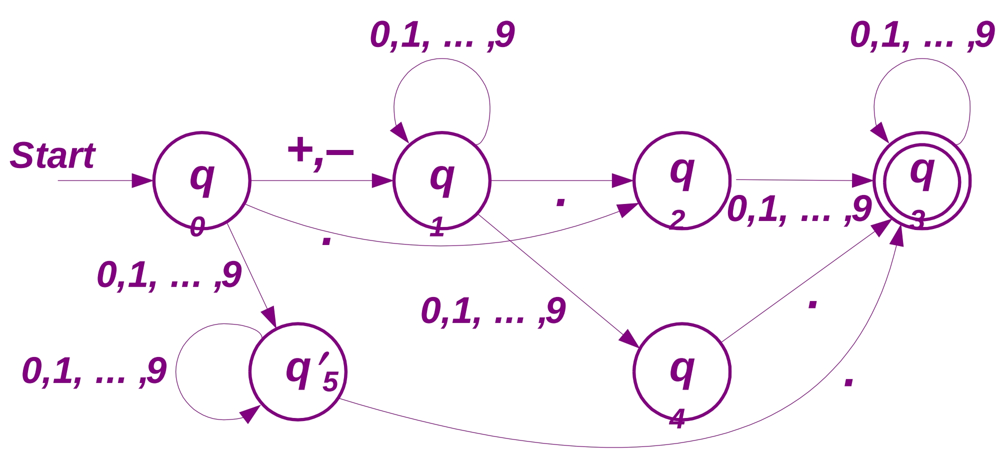
	0	1
→ { p }	{ p }	{ p, q }
{ p, q }	{ p, r }	{ p, q, r }
*{ p, r }	{ p }	{ p, q }
*{ p, q, r }	{ p, r }	{ p, q, r }



✧ 带ε-转移的非确定有限自动机(ε - NFA)举例



比较：*NFA without ε*



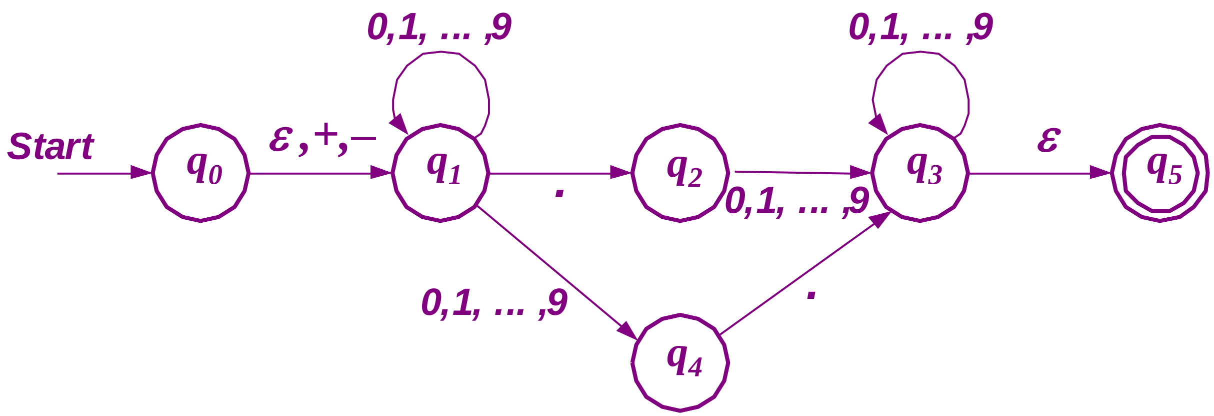
✧ 带 ε-转移的非确定有限自动机（ε-NFA）的形式定义

一个 ε-NFA 是一个五元组 $A = (Q, \Sigma, \delta, q_0, F)$.

- ✧ 有限状态集 —— Q
- ✧ 有限输入符号集 —— Σ
- ✧ 转移函数 —— δ
- ✧ 一个开始状态 —— q_0　　$q_0 \in Q$
- ✧ 一个终态集合 —— F　　$F \subseteq Q$
- ✧ 与 NFA 的不同之处

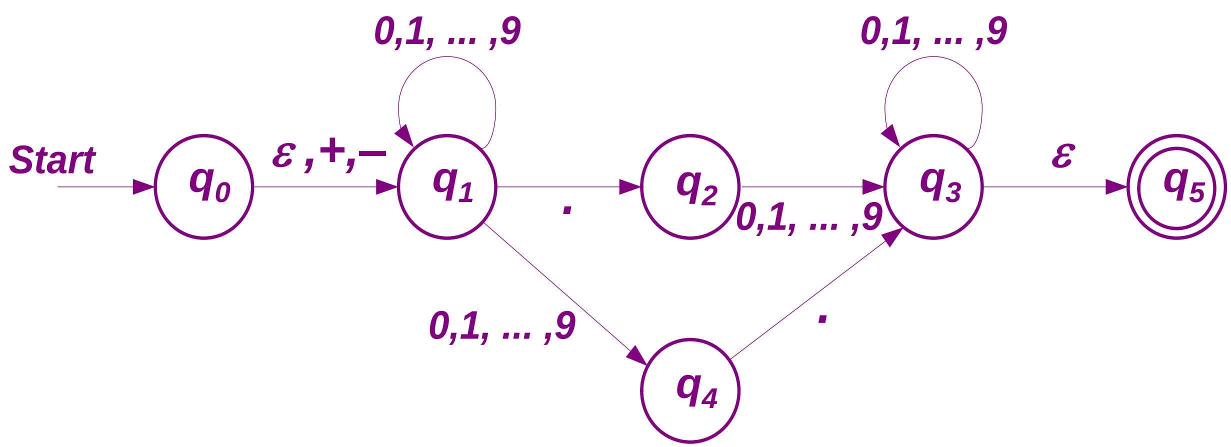
$$\delta:\ Q \times \Sigma \cup \{\varepsilon\} \to 2^Q$$

✧ 转移图和转移表表示的 ε - NFA



	ε	$+,-$	.	$0,1,\ldots,9$
→ q_0	$\{q_1\}$	$\{q_1\}$	ϕ	ϕ
q_1	ϕ	ϕ	$\{q_2\}$	$\{q_1,q_4\}$
q_2	ϕ	ϕ	ϕ	$\{q_3\}$
q_3	$\{q_5\}$	ϕ	ϕ	$\{q_3\}$
q_4	ϕ	ϕ	$\{q_3\}$	ϕ
* q_5	ϕ	ϕ	ϕ	ϕ

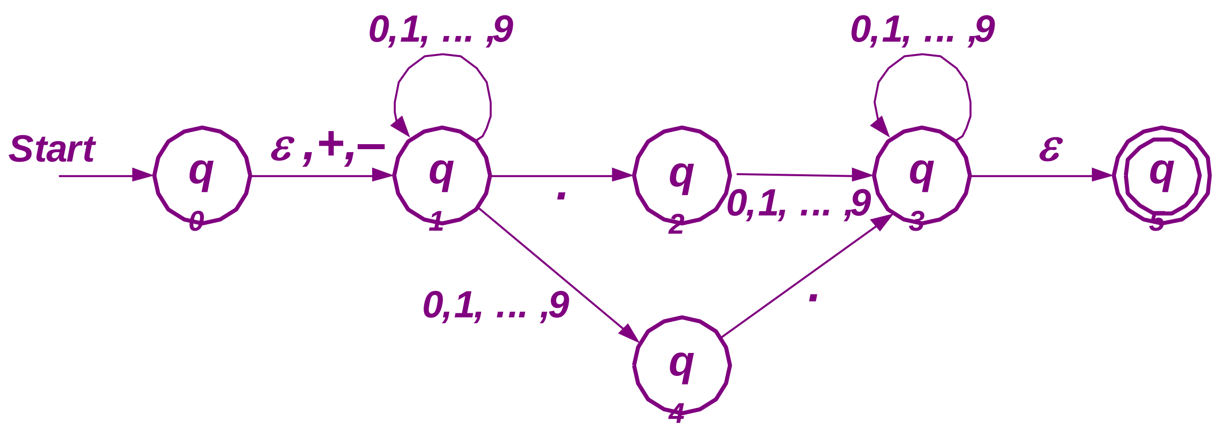
✧ ε - NFA 如何接受输入符号串



– 该 ε - ***NFA*** 可以接受的字符串如:

- ***3.14***
- ***+.314***
- ***– 314.***

✧ ε- 闭包（closure）



– 状态 q 的ε- 闭包，记为**ECLOSE(q)**，定义为从 q 经所有的ε路径可以到达的状态（包括q自身），如：

- $ECLOSE(q_0) = \{q_0, q_1\}$
- $ECLOSE(q_2) = \{q_2\}$
- $ECLOSE(q_3) = \{q_3, q_5\}$

正规语言及其描述

✧ ε-闭包

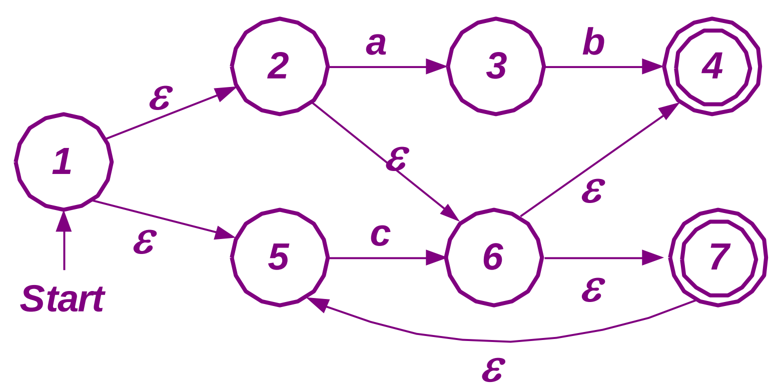
– 设 ε - NFA $A = (Q, \Sigma, \delta, q_0, F)$，$q \in Q$，$ECLOSE(q)$ 为满足如下条件的最小集：

（1） $q \in ECLOSE(q)$

（2） if $p \in ECLOSE(q)$ and $r \in \delta(p, \varepsilon)$, then $r \in ECLOSE(q)$

– 对于右图，有：

- $ECLOSE(1) = \{1,2,4,5,6,7\}$
- $ECLOSE(2) = \{2,4,5,6,7\}$
- $ECLOSE(7) = \{5,7\}$



✧ 扩展转移函数适合于输入字符串

- 设一个 ε - NFA $E = (Q, \Sigma, \delta, q_0, F)$
- $\delta: Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q$
- 扩充定义 δ': $Q \times \Sigma^* \rightarrow 2^Q$
- 对任何$q \in Q$，定义：

  1 $\delta'(q, \varepsilon) = ECLOSE(q)$

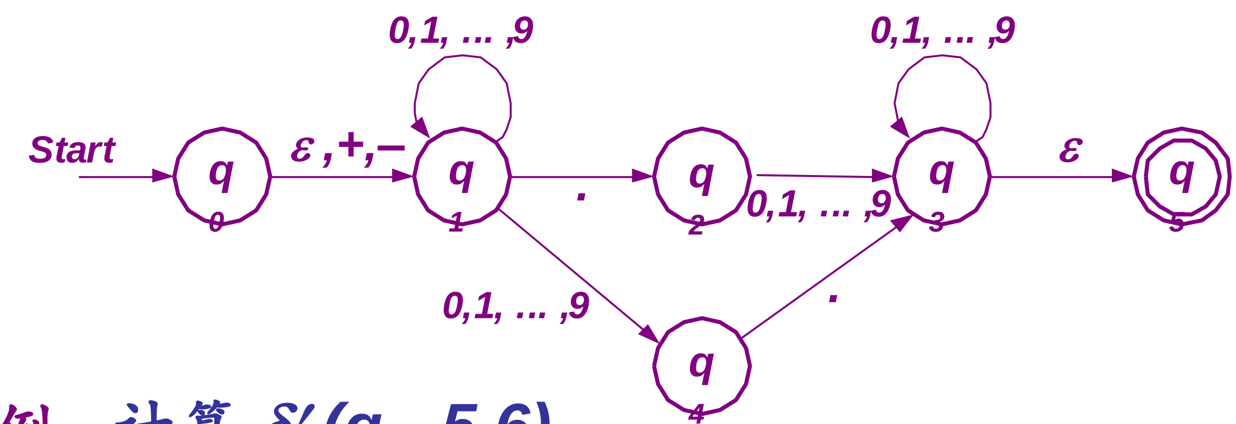
  2 若$w = xa$，其中 $x \in \Sigma^*$，$a \in \Sigma$，假设

  $\delta'(q, x) = \{p_1, p_2, \ldots, p_k\}$，并且

  令 $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$，则

  $$\delta'(q, w) = \bigcup_{j=1}^{m} ECLOSE(r_j)$$

✧ 扩展转移函数适合于输入字符串



– 举例 计算 $\delta'(q_0, 5.6)$

- $\delta'(q_0, \varepsilon) = ECLOSE(q_0) = \{q_0, q_1\}$
- $\delta(q_0, 5) \cup \delta(q_1, 5) = \{q_1, q_4\}$
  $\delta'(q_0, 5) = ECLOSE(q_1) \cup ECLOSE(q_4) = \{q_1, q_4\}$
- $\delta(q_1, .) \cup \delta(q_4, .) = \{q_2, q_3\}$
  $\delta'(q_0, 5.) = ECLOSE(q_2) \cup ECLOSE(q_3) = \{q_2, q_3, q_5\}$
- $\delta(q_2, 6) \cup \delta(q_3, 6) \cup \delta(q_5, 6) = \{q_3\}$
  $\delta'(q_0, 5.6) = ECLOSE(q_3) = \{q_3, q_5\}$

✧ ε - NFA 的 语 言

- 设一个 ε - NFA $E = (Q, \Sigma, \delta, q_0, F)$
- 定义 E 的语言：

$$L(E) = \{ w \mid \delta'(q_0, w) \cap F \neq \phi \}$$

- 设 L 是 Σ 上的语言，如果存在一个 ε-NFA $E = (Q, \Sigma, \delta, q_0, F)$，满足 $L = L(E)$，则可以证明 L 也是一个正规语言.

✧ ε - NFA 与 DFA 的等价性

– 定理: *L* 是某个 ε - NFA 的语言, 当且仅当 *L* 也是某个 DFA 的语言.

– 证明思路: 分两步证明.

(1) 设 *L* 是某个 DFA *D* 的语言, 则存在一个 ε - NFA *E*, 满足 $L(E) = L(D) = L$;

(2) 设 *L* 是某个 ε - NFA *E* 的语言, 则存在一个 DFA *D*, 满足 $L(D) = L(E) = L$.

✧ 从 DFA 构造等价的 ε - NFA

– 设 L 是某个 DFA $D = (Q, \Sigma, \delta_D, q_0, F)$ 的语言，则存在一个 ε - NFA $E = (Q, \Sigma, \delta_E, q_0, F_E)$，其中 δ_E 定义为

  - 对任何$q \in Q$, $\delta_E(q, \varepsilon) = \phi$
  - 对任何$q \in Q$ 和$a \in \Sigma$，若 $\delta_D(q,a) = p$, 则 $\delta_N(q,a) = \{p\}$.

可以证明$L(E) = L(D) = L$

✧ 从 ε - NFA 构造等价的 DFA（修改的子集构造法）

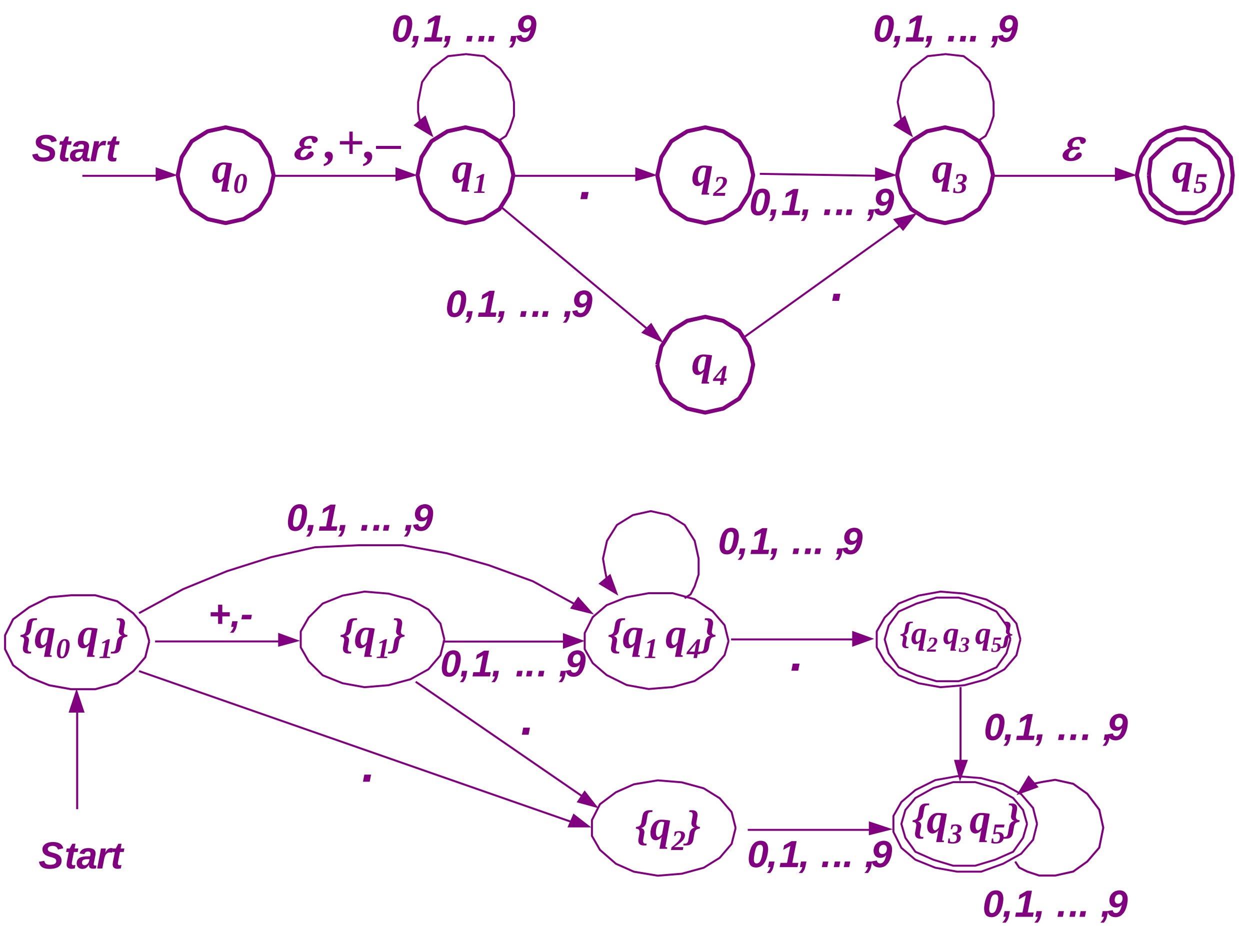
– 设 L 是某个 ε - NFA $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ 的语言，则存在一个 DFA $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$，其中

- $Q_D = \{ S \mid S \subseteq Q_E \wedge S = ECLOSE(S) \}$
- $q_D = ECLOSE(q_0)$
- $F_D = \{S \mid S \in Q_D \wedge S \cap F_E \neq \phi \}$
- 对 $S \in Q_D$ 和 $a \in \Sigma$，令 $S = \{ p_1, p_2, \ldots, p_k \}$，并设 $\bigcup_{i=1}^{k} \delta_E(p_i, a) = \{ r_1, r_2, \ldots, r_m \}$，则

$$\delta_D(S, a) = \bigcup_{j=1}^{m} ECLOSE(r_j).$$

可以证明: $L(D) = L(E) = L$.

✧ 修改的子集构造法举例


0,1, … ,9
0,1, … ,9
Start
q0
ε,+,–
q1
.
q2
0,1, … ,9
q3
ε
q5
0,1, … ,9
.
q4
0,1, … ,9
0,1, … ,9
+,-
{q0 q1}
{q1}
0,1, … ,9
{q1 q4}
.
{q2 q3 q5}
0,1, … ,9
.
.
{q2}
{q3 q5}
Start
0,1, … ,9
0,1, … ,9

✧ 正规表达式

- 用代数的方法表示正规语言
- 语义 作用于语言上的三种代数运算
  联合（*union*）
  连接（*concatenation*）
  （星）闭包（*closure*）
- 语法 不同应用有所不同，但都含有上述三种代数运算的表示形式；为方便起见，通常还需要引入一些助记符

正规语言及其描述

✧ 正规表达式（regular expression）

– 定义及解释

设正规表达式E代表的语言为L(E),归纳定义正规表达式如下：

基础 1 ε 和 ϕ 为正规表达式，且 $L(\varepsilon)=\{\varepsilon\}$、 $L(\phi)=\phi$

2 若a 为任一字符，则 a 为正规表达式，且 $L(a)=\{a\}$

3 任一变量（通常大写）L 为正规表达式，代表任意语言

归纳 1 若 E 和 F 为正规表达式，则 E | F 也为正规表达式，且满足 $L(E\,|\,F)=L(E)\cup L(F)$

2 若 E 和 F 为正规表达式，则 EF 也为正规表达式，且满足 $L(EF)=L(E)L(F)$

3 若 E 为正规表达式，则 E* 也为正规表达式，且满足 $L(E^*)=(L(E))^*$

4 若 E 为正规表达式，则 (E) 也为正规表达式，且满足 $L((E))=L(E)$

✧ 正规表达式算符优先级

算符优先级（***precedence***）依次为

- *（ ***closure*** ）
- • 连接（***concatenation***）
- |（ ***union*** ）

✧ 正规表达式举例

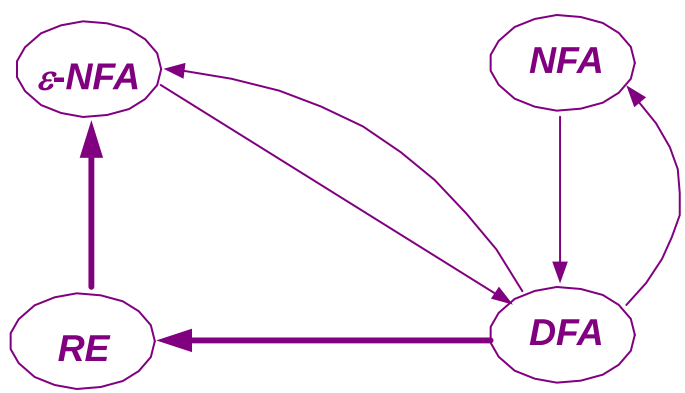
设计表示如下语言的正规表达式:
该语言中的每个字符串由交替的 **0** 和 **1** 构成

- *(01)* | (10)* | 0(10)* | 1(01)**
- $(\varepsilon \mid 1)\,(01)^*\,(\varepsilon \mid 0)$
- $(\varepsilon \mid 0)\,(10)^*\,(\varepsilon \mid 1)$

✧ 有限自动机与正规表达式的关系

结论：正规表达式所表示的语言是正规语言.

证明策略

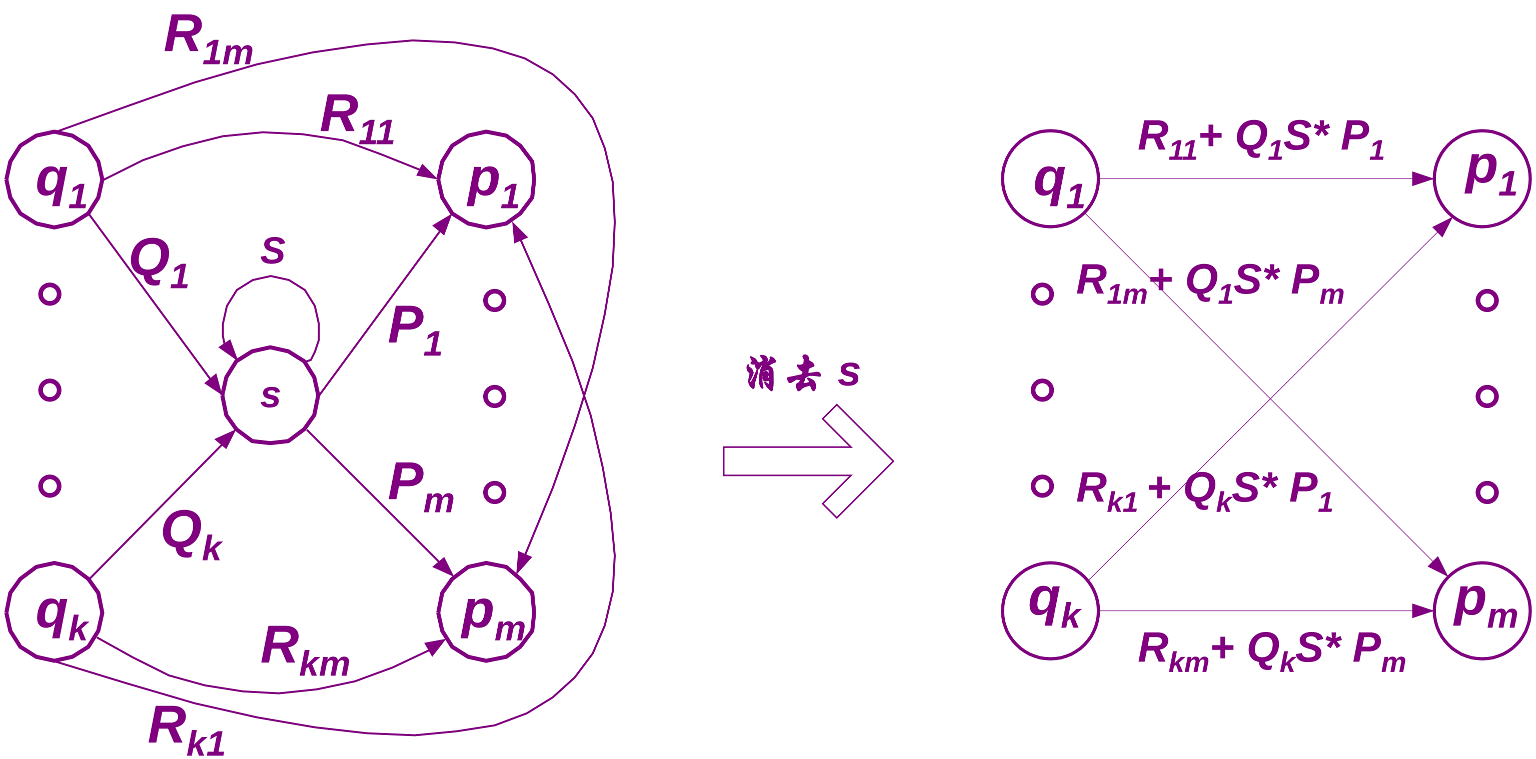


✧ 从有限自动机构造等价的正规表达式

思路（状态消去法）：

- 扩展自动机的概念，允许正规表达式作为转移弧的标记. 这样，就有可能在消去某一中间状态时，保证自动机能够接受的字符串集合保持不变.
- 在消去某一中间状态时，与其相关的转移弧也将同时消去，所造成的影响将通过修改从每一个前趋状态到每一个后继状态的转移弧标记来弥补.

✧ 从有限自动机构造等价的正规表达式（中间状态的消去）



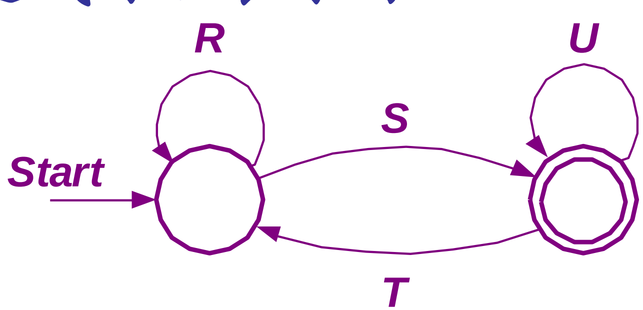
✧ 从有限自动机构造等价的正规表达式

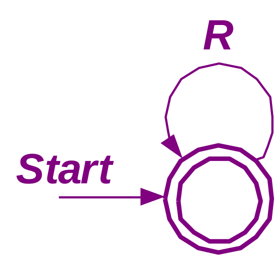
步骤（状态消去法）：

(1) 对每一终态q，依次消去除 q 和初态 q_0 之外的其它状态;

(2) 若$q \neq q_0$，最终可得到一般形式如下左图两状态自动机，该自动机对应的正规表达式可表示为 $(R \mid SU^*T)^*SU^*$.

(3) 若$q = q_0$，最终可得到如下右图的自动机，它对应的正规表达式可以表示为 R^*.

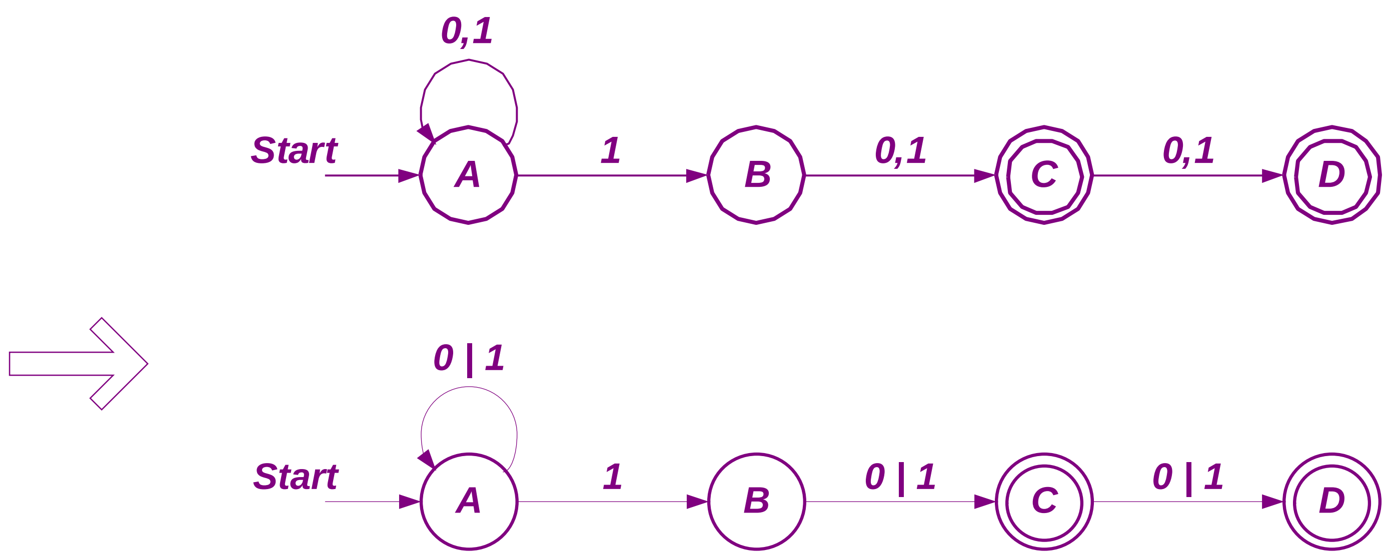




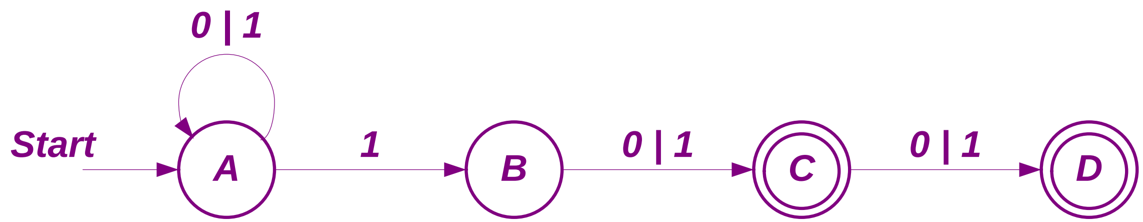
(4) 最终的正规表达式为每一终态对应的正规表达式之和（并）.

✧ 从有限自动机构造等价的正规表达式

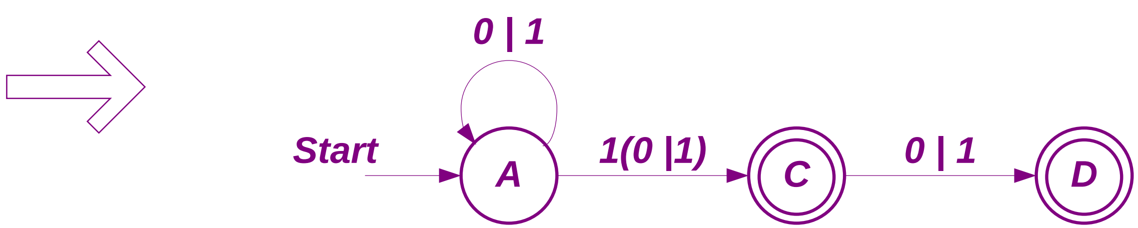
– 状态消去法举例

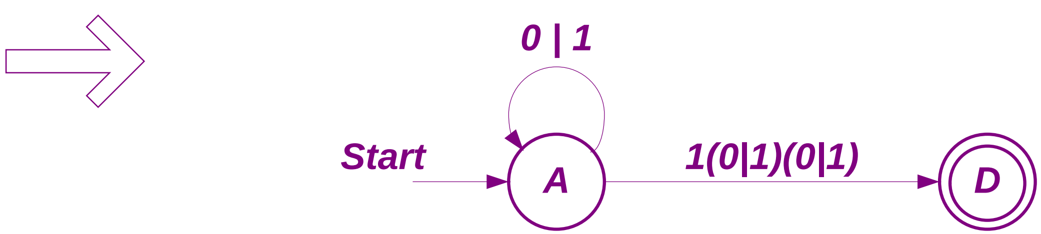


- 从有限自动机构造等价的正规表达式
  - 状态消去法举例



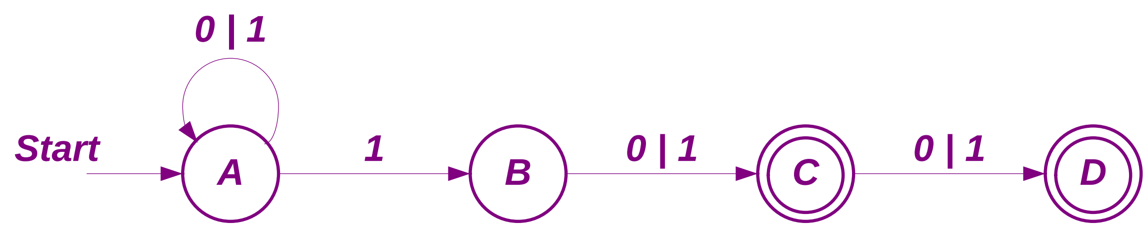
对于终态*D*



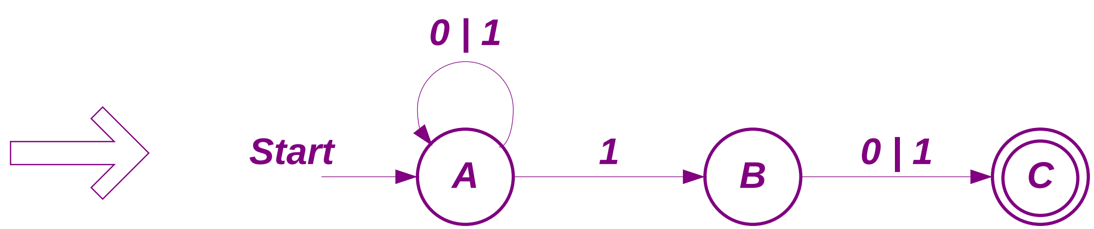


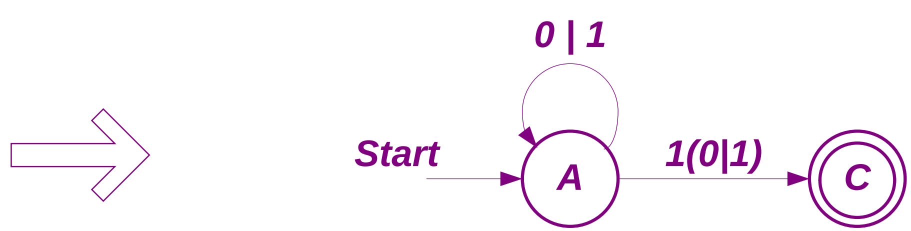
✧ 从有限自动机构造等价的正规表达式

– 状态消去法举例



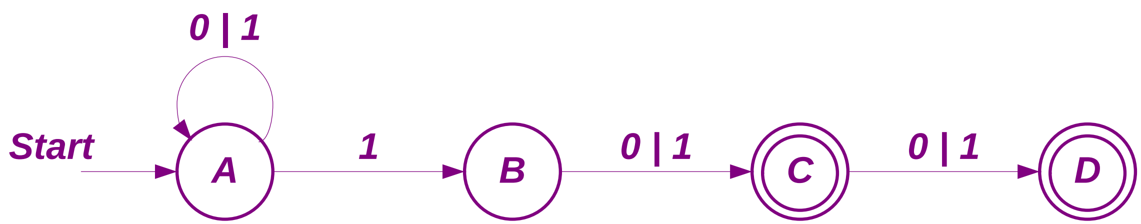
对于终态C



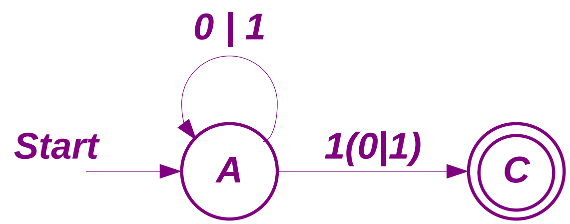


✧ 从有限自动机构造等价的正规表达式

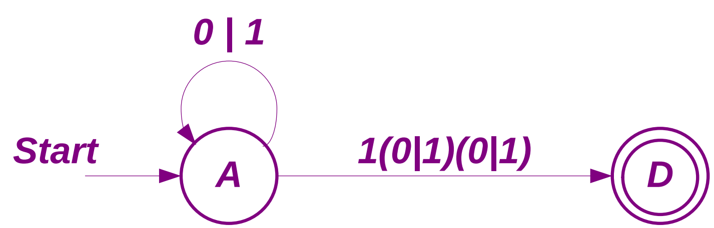
– 状态消去法举例



对于终态C



对于终态D



等价的正规表达式

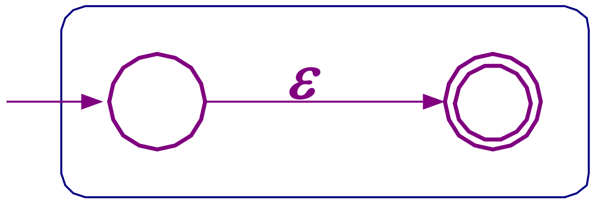
(0|1)*1(0|1) | (0|1)*1(0|1)(0|1)

✧ 从正规表达式构造等价的ε - NFA

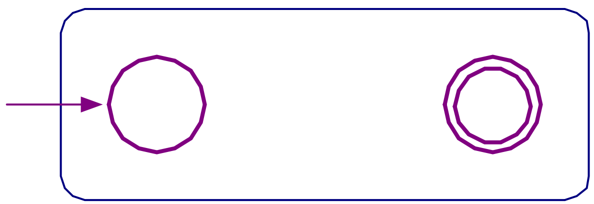
（归纳构造过程: *Thompson* 构造法）

基础:

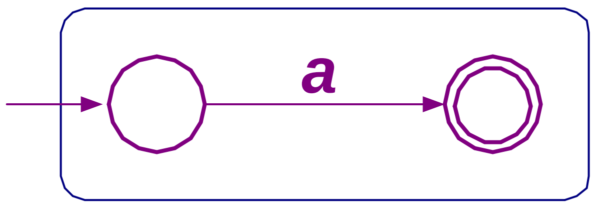
1 对于 ε，构造为



2 对于 ϕ，构造为



3 对于 a，构造为

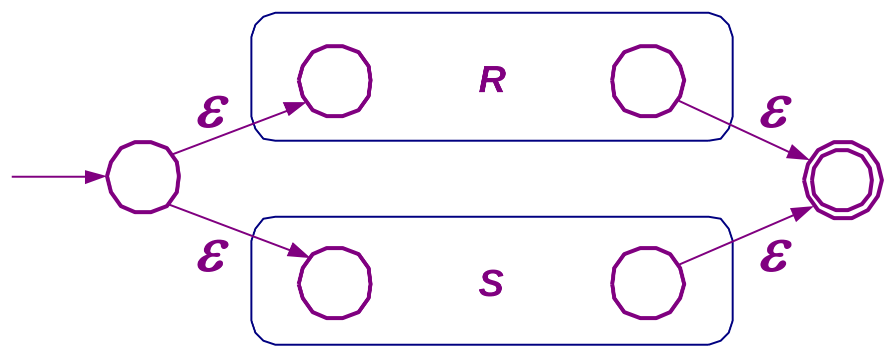


✧ 从正规表达式构造等价的ε-*NFA*

（归纳构造过程：*Thompson* 构造法）

归纳：

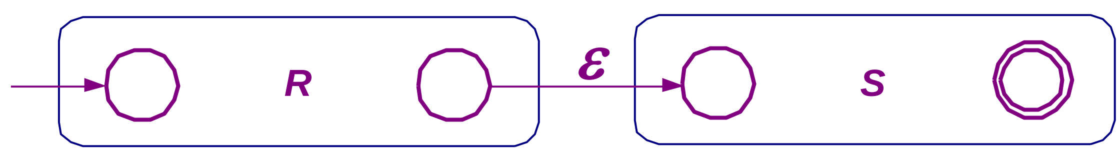
1 对于 ***R* | *S***，构造为



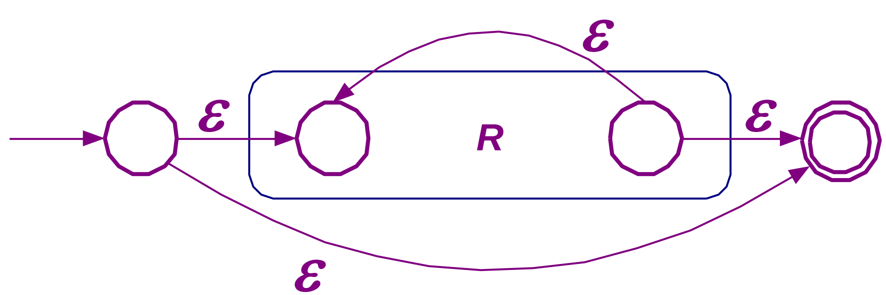
✧ 从正规表达式构造等价的ε - NFA
（归纳构造过程: *Thompson* 构造法）

归纳:

2 对于 ***RS***，构造为



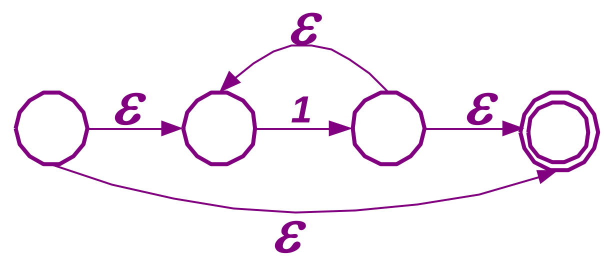
3 对于 ***R****，构造为



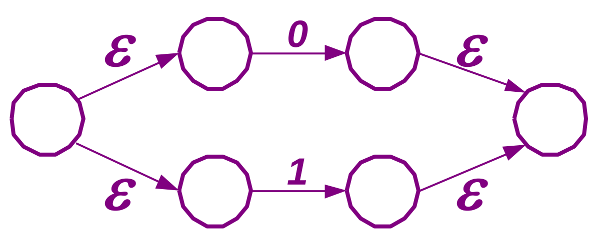
◇ 从正规表达式构造等价的ε - NFA

举例:设正规表达式 *1*0(0|1)**, 构造等价的ε-NFA.

1* ⇒



0|1 ⇒

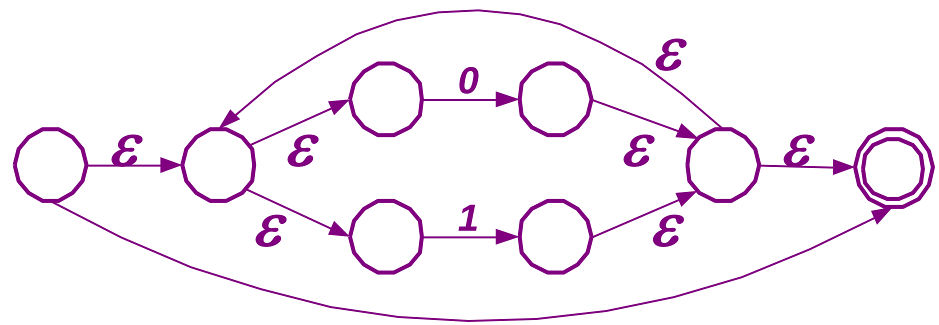


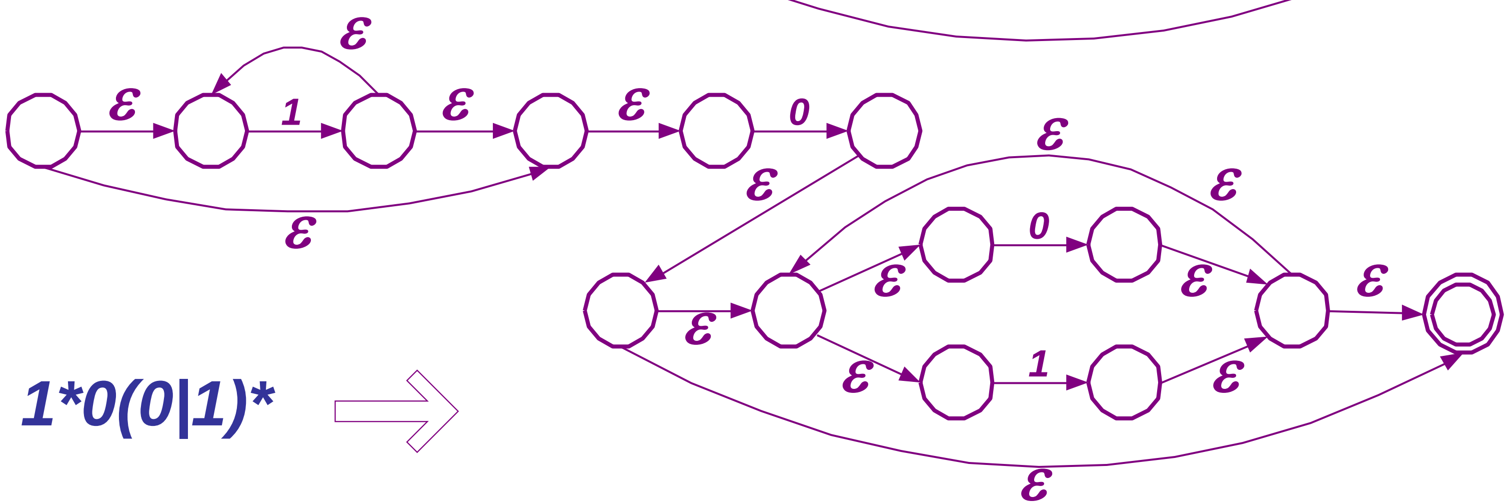
正规语言及其描述

✧ 从正规表达式构造等价的ε - NFA

举例:设正规表达式 *1*0(0|1)**, 构造等价的ε-*NFA*.

*(0|1)** ⇒





*1*0(0|1)** ⇒

✧ DFA 的化简

– 知识回顾：集合上的等价关系与划分

等价关系

设 Q 为一个集合，二元关系 R 是 Q 上的一个等价关系，当且仅当满足以下条件：

1. 自反性 对任何$a \in Q$, aRa 成立;
2. 对称性 对任何$a,b \in Q$, 如果 aRb 成立, 则有 bRa 成立;
3. 传递性 对任何$a,b,c \in Q$, 如果 aRb 和 bRc 成立, 则有 aRc 成立.

✧ DFA 的化简

– 知识回顾：集合上的等价关系与划分

设 Q 为一个集合，R 是 Q 上的一个等价关系，由 R 产生的所有等价类（或块）的集合构成 Q 的一个划分.

– 注释

1. 等价类 对任何$a \in Q$，a 所在的块用$[a]$表示，定义为

$$[a] = \{x \mid xRa\};$$

2. 每一元素都属于唯一的块 即满足

（1）$\cup_{a \in Q} [a] = Q$； 和

（2）对任何$a,b \in Q$，或者 $[a]=[b]$，或者 $[a]\cap[b]=\Phi$

正规语言及其描述

✧ DFA 的化简

– DFA 状态集合上的一个等价关系

设一个 DFA $D = (Q, \Sigma, \delta, q_0, F)$, 定义Q上的一个二元关系 R 为： 对任何 $p,q \in Q$,

$$pRq \ \text{ iff } \ \forall w \in \Sigma^*. \, (\delta'(p,w) \in F \leftrightarrow \delta'(q,w) \in F)$$

证明： 1. 自反性 对任何$q \in Q$, qRq 成立;

2. 对称性 对任何$p,q \in Q$, $pRq \rightarrow qRp$ 成立;

3. 传递性 对任何$p,q,r \in Q$, 设pRq 和 qRr 成立, 即对任何$w \in \Sigma^*$, $\delta'(p,w) \in F \leftrightarrow \delta'(q,w) \in F$ 和 $\delta'(q,w) \in F \leftrightarrow \delta'(r,w) \in F$ 成立; 由此, 也有 $\delta'(p,w) \in F \leftrightarrow \delta'(r,w) \in F$ 成立. 所以, qRr 成立

✧ DFA 的化简

– DFA 状态集合上的一个等价关系

若pRq, 称 p 和 q 等价（equivalent）. 若p 和q 不等价，则称 p 和q 是可区别的（distinguashable）.

关系 R 对应有限状态集 Q 的一个划分;
该划分的每个块是 Q 的一个子集;
同一划分块中的所有状态之间都是相互等价的;
分属不同划分块的任何两个状态之间都是可区别的.

– DFA 的优化 通过合并等价的（或不可区别的）状态.
关键：如何计算上述划分?

✧ DFA 的化简

– 计算状态集划分的算法

填表算法（*table-filling algorithm*）基于如下递归地标记可区别的状态偶对的过程:

基础 如果 p 为终态，而 q 为非终态，则 p 和 q 标记为可区别的;

归纳 设 p 和 q 已标记为可区别的, 如果状态 r 和 s 通过某个输入符号 a 可分别转移到 p 和 q ，即 $\delta(r,a)=p$, $\delta(s,a)=q$ ，则 r 和 s 也标记为可区别的;

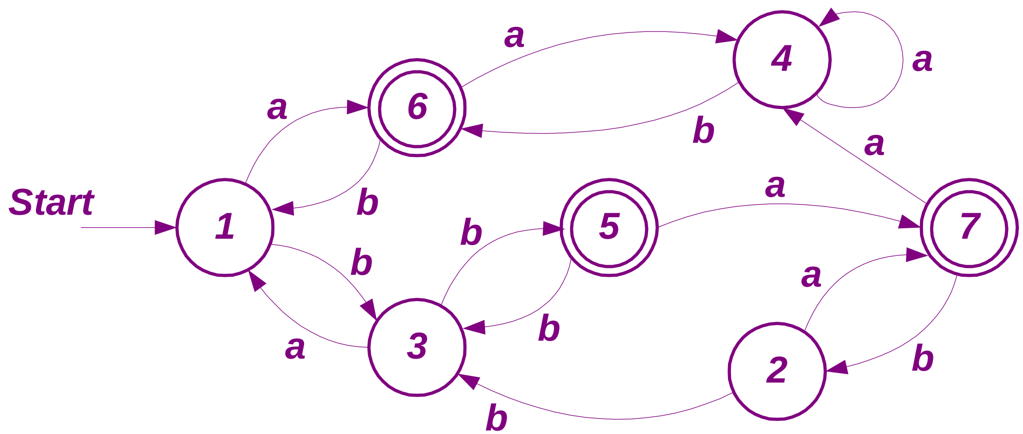
这是因为：若 p 和 q 可为字符串 w 区别, 则 r 和 s 可为字符串 aw 区别.

（$\because \delta'(r,aw)=\delta'(p,w)$, $\delta'(s,aw)=\delta'(q,w)$ ）

✧ DFA 的化简

– 填表算法举例

2						
3	X	X				
4	X	X	X			
5	X	X	X	X		
6	X	X	X	X	X	
7	X	X	X	X	X	
	1	2	3	4	5	6



(1) 区别所有终态和非终态

(2) 区别(1,3), (1,4), (2,3), (2,4), (5,6), (5,7)

(3) 区别 (3,4)

(4) 结束. 划分结果：{1,2}, {3}, {4}, {5}, {6,7}

✧ DFA 的最小化

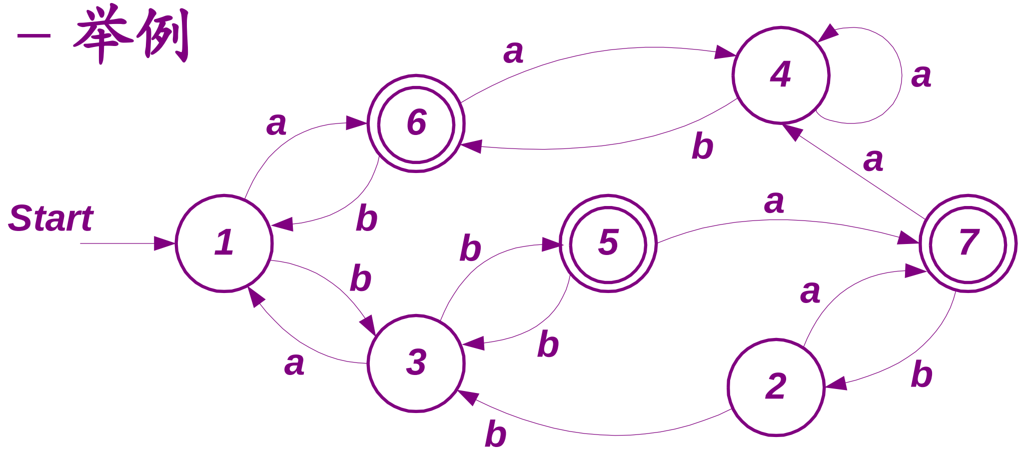
– 可以证明：

通过以下步骤，可以得到状态数最少的等价DFA

1. 删除所有从开始状态不可到达的状态及与其相关的边，设所得到的 DFA 为 $A = (Q, \Sigma, \delta, q_0, F)$；
2. 使用填表算法找出所有等价的状态偶对；
3. 根据 2 的结果计算当前状态集合的划分块，每一划分块中的状态相互之间等价，而不同划分块中的状态之间都是可区别的. 包含状态 q 的划分块用 $[q]$ 表示.
4. 构造与 A 等价的 DFA $B = (Q_B, \Sigma, \delta_B, [q_0], F_B)$，其中 $Q_B=\{ [q] \mid q \in Q\}$, $F_B = \{ [q] \mid q \in F\}$, $\delta_B([q], a)=[\delta(q,a)]$

✧ DFA 的最小化

– 举例



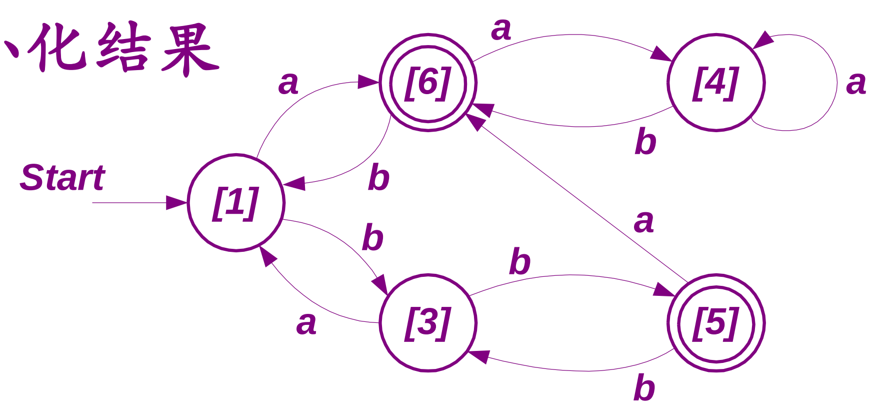
– 等价的状态偶对为:

（1, 2），（6, 7）

– 划分结果:

{1, 2 }, {3}, {4}, {5}, {6, 7 }

– 最小化结果



– 新的状态集合:

[1], [3], [4], [5], [6]

✧ 上下文无关文法(*context-free grammars*)

– 例子

$$E \rightarrow EOE$$
$$E \rightarrow (E)$$
$$E \rightarrow v$$
$$E \rightarrow d$$
$$O \rightarrow +$$
$$O \rightarrow *$$

✧ 上下文无关文法的四个基本要素

1. 终结符(*terminals*)的集合 有限符号集，相当于字母表
2. 非终结符(*nonterminals*)的集合 有限变量符号的集合
3. 开始符号(*start symbol*) 一个特殊的非终结符
4. 产生式(*productions*)的集合 形如：***<head>*** → ***<body>***

开始符号

E → *EOE* —— 产生式集合
E → (*E*)
E → *v*
E → *d*
O → +
O → *

非终结符 终结符

✧ 上下文无关文法的形式定义

一个上下文无关文法 CFG (context-free grammars) 是一个四元组 $G = (V_N, V_T, P, S)$.

非终结符的集合 → V_N

终结符的集合 → V_T

产生式的集合 → P

开始符号 → S

满足

$V_N \cap V_T = \Phi$

$S \in V_N$

产生式形如 $A \rightarrow \alpha$, 其中 $A \in V_N$, $\alpha \in (V_N \cup V_T)^*$

✧ 上下文无关文法举例

CFG G_{exp} = ({E,O}, { (,), +, ∗, v, d }, P, E).

其中产式集合 P 为

$$E \to EOE$$
$$E \to (E)$$
$$E \to v$$
$$E \to d$$
$$O \to +$$
$$O \to *$$

✧ 产生式集合的缩写记法

形如 $A \rightarrow \alpha_1, A \rightarrow \alpha_2, \ldots, A \rightarrow \alpha_n$ 的产生式集合可简缩记为 $A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$，如

$$
\begin{aligned}
E &\rightarrow EOE \\
E &\rightarrow (E) \\
E &\rightarrow v \\
E &\rightarrow d \\
O &\rightarrow + \\
O &\rightarrow *
\end{aligned}
\quad \Longrightarrow \quad
\begin{aligned}
E &\rightarrow EOE \mid (E) \mid v \mid d \\
O &\rightarrow + \mid *
\end{aligned}
$$

✧ 归约与推导

– 用于推理字符串是否属于文法所定义的语言

一种是自下而上的方法，称为递归推理（*recursive inference*），递归推理的过程习称为归约；另一种是自上而下的方法，称为推导（*derivation*）.

– 归约过程　将产生式的右部（*body*）替换为产生式的左部（ *head* ）.

– 推导过程　将产生式的左部（ *head* ）替换为产生式的右部（ *body* ）.

✧ 归约过程举例

对于CFG G_{exp} = ({E,O}, { (,),＋, ∗, v, d }, P , E)， P 为

（1） $E \to EOE$
（2） $E \to (E)$
（3） $E \to v$
（4） $E \to d$
（5） $O \to +$
（6） $O \to *$

递归推理出字符串 $v*(v+d)$ 的一个归约过程为

$$v*(v+d) \xrightarrow{(4)} v*(v+E) \xrightarrow{(6)} vO(v+E) \xrightarrow{(3)} vO(E+E)$$

$$\xrightarrow{(5)} vO(EOE) \xrightarrow{(1)} vO(E) \xrightarrow{(2)} vOE \xrightarrow{(3)} EOE \xrightarrow{(1)} E$$

✧ 推导过程举例

对于CFG $G_{exp} = (\{E,O\}, \{ (,), +, *, v, d \}, P, E)$，P 为

$$
\begin{aligned}
&(1)\quad E \rightarrow EOE \\
&(2)\quad E \rightarrow (E) \\
&(3)\quad E \rightarrow v \\
&(4)\quad E \rightarrow d \\
&(5)\quad O \rightarrow + \\
&(6)\quad O \rightarrow *
\end{aligned}
$$

递归推理出字符串 $v*(v+d)$ 的一个推导过程为

$$
E \xRightarrow{(1)} EOE \xRightarrow{(6)} E*E \xRightarrow{(2)} E*(E) \xRightarrow{(3)} v*(E)
$$

$$
\xRightarrow{(1)} v*(EOE) \xRightarrow{(5)} v*(E+E) \xRightarrow{(3)} v*(v+E) \xRightarrow{(4)} v*(v+d)
$$

上下文无关语言及其描述

✧ 推导关系

对于 CFG $G = (V_N, V_T, P, S)$，上述推导过程可用关系$\underset{G}{\Rightarrow}$描述. 设 $\alpha,\beta\in(V_N\cup V_T)^*$，$A\rightarrow\gamma$ 是一个产生式，则定义

$$\alpha A\beta \underset{G}{\Rightarrow} \alpha\gamma\beta .$$

若 G 在上下文中是明确的，则简记为$\alpha A\beta \Rightarrow \alpha\gamma\beta$.

✧ 扩展推导关系到自反传递闭包

定义上述关系的自反传递闭包，记为 $\overset{*}{\underset{G}{\Rightarrow}}$，可归纳定义如下:

基础 对任何 $\alpha\in(V_N\cup V_T)^*$，满足 $\alpha \overset{*}{\underset{G}{\Rightarrow}} \alpha$.

归纳 设 $\alpha,\beta,\gamma\in(V_N\cup V_T)^*$，若 $\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta$， $\beta \underset{G}{\Rightarrow} \gamma$ 成立，则

$$\alpha \overset{*}{\underset{G}{\Rightarrow}} \gamma .$$

✧ 最左推导 (*leftmost derivations*)

若推导过程的每一步总是替换出现在最左边的非终结符，则这样的推导称为**最左推导**. 为方便，最左推导关系用 $\underset{lm}{\Rightarrow}$ 表示，其自反传递闭包用 $\overset{*}{\underset{lm}{\Rightarrow}}$ 表示.

如对于文法 G_{exp}，下面是关于 $v*(v+d)$ 的一个最左推导:

$$E \overset{*}{\underset{lm}{\Rightarrow}} EOE \overset{*}{\underset{lm}{\Rightarrow}} vOE \overset{*}{\underset{lm}{\Rightarrow}} v*E$$

$$\overset{*}{\underset{lm}{\Rightarrow}} v*(E) \overset{*}{\underset{lm}{\Rightarrow}} v*(EOE) \overset{*}{\underset{lm}{\Rightarrow}} v*(vOE)$$

$$\overset{*}{\underset{lm}{\Rightarrow}} v*(v+E) \overset{*}{\underset{lm}{\Rightarrow}} v*(v+d)$$

$E \rightarrow EOE$
$E \rightarrow (E)$
$\boxed{E \rightarrow v \quad E \rightarrow d}$
$O \rightarrow +$
$O \rightarrow *$

✧ 最右推导 (*rightmost derivations*)

若推导过程的每一步总是替换出现在最右边的非终结符，则这样的推导称为**最右推导**. 为方便，最右推导关系用$\underset{rm}{\Rightarrow}$表示，其自反传递闭包用$\underset{rm}{\overset{*}{\Rightarrow}}$表示.

如对于文法G_{exp}，下面是关于 $v*(v+d)$ 的一个最右推导：

$$E \underset{rm}{\overset{*}{\Rightarrow}} EOE \underset{rm}{\overset{*}{\Rightarrow}} EO(E) \underset{rm}{\overset{*}{\Rightarrow}} EO(EOE)$$

$$\underset{rm}{\overset{*}{\Rightarrow}} EO(EOd) \underset{rm}{\overset{*}{\Rightarrow}} EO(E+d) \underset{rm}{\overset{*}{\Rightarrow}} EO(v+d)$$

$$\underset{rm}{\overset{*}{\Rightarrow}} E*(v+d) \underset{rm}{\overset{*}{\Rightarrow}} v*(v+d)$$

$E \rightarrow EOE$
$E \rightarrow (E)$
$E \rightarrow v$
$E \rightarrow d$
$O \rightarrow +$
$O \rightarrow *$

✧ 句型 (*sentential forms*)

设 CFG $G = (V_N, V_T, P, S)$，称 $\alpha \in (V_N \cup V_T)^*$ 为 G 的一个句型，当且仅当 $S \overset{*}{\Rightarrow} \alpha$.

若 $S \underset{lm}{\overset{*}{\Rightarrow}} \alpha$，则 α 是一个左句型;

若 $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha$，则 α 是一个右句型.

若句型 $\alpha \in V_T^*$，则称 α 为一个句子（*sentence*）.

上下文无关语言及其描述

✧ 上下文无关文法的语言

设 *CFG* $G = (V_N, V_T, P, S)$，定义 G 的语言为

$$L(G) = \{ w \mid w \in V_T^* \wedge S \overset{*}{\underset{G}{\Rightarrow}} w \}$$

✧ 上下文无关语言（*context-free languages*）

如果一个语言 L 是某个 *CFG* G 的语言，即

$$L(G) = L,$$

则 L 是上下文无关语言.

✧ 文法与语言的 Chomsky 分类方法

– 文法（*grammar*）是一个四元组

$$G = (V_N, V_T, P, S),$$

V_N、V_T、P 及 S 的含义如前. Chomsky 通过对产生式施加不同的限制，把文法及其对应的语言分成四种类型，即 0 型、1 型、 2 型和 3 型.

✧ 文法与语言的 Chomsky 分类方法

– **0 型**

0 型文法 $G = (V_N, V_T, P, S)$ 的产生式形如

$$\alpha \rightarrow \beta,$$

其中 $\alpha, \beta \in (V_N \cup V_T)^*$，但 α 中至少包含一个非终结符.

能够用 0 型文法定义的语言称为0 型语言或递归可枚举语言.

✧ 文法与语言的 Chomsky 分类方法

– 1 型

1 型文法 $G = (V_N, V_T, P, S)$ 的产生式形如

$$\alpha \to \beta,$$

满足 $|\alpha| \le |\beta|$，仅 $S \to \varepsilon$ 例外，且要求 S 不得出现在任何产生式的右部.

1 型文法也称为上下文有关文法（context-sensitive grammars）.

能够用 1 型文法定义的语言称为 1 型语言或 上下文有关语言.

✧ 文法与语言的 Chomsky 分类方法

– 2 型

2 型文法 $G = (V_N, V_T, P, S)$ 的产生式形如

$$A \rightarrow \beta,$$

其中$A \in V_N$，$\beta \in (V_N \cup V_T)^*$.

2 型文法也称为上下文无关文法.

能够用 2 型文法定义的语言称为2 型语言，或上下文无关语言.

✧ 文法与语言的 Chomsky 分类方法

– 3 型

3 型文法 $G = (V_N, V_T, P, S)$ 的产生式形如

$$A \rightarrow aB \text{ 或 } A \rightarrow a,$$

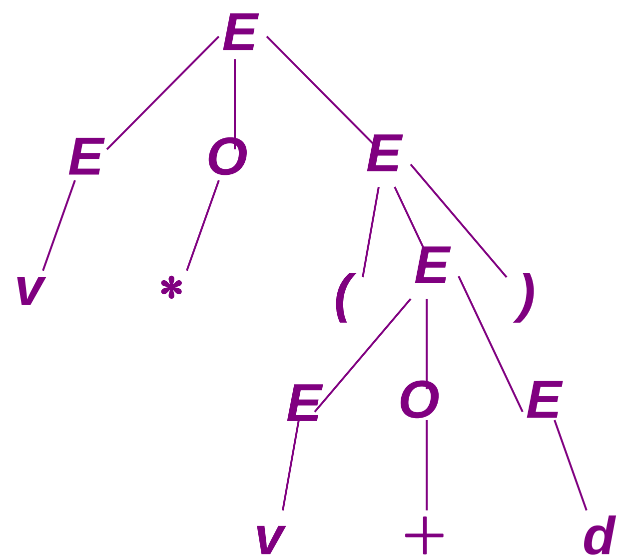
其中 $A, B \in V_N$， $a \in V_T \cup \{\varepsilon\}$.

能够用 3 型文法定义的语言称为3 型语言，或正规语言.

3 型文法也称为正规文法.

✧ 语法分析树

– 归约过程自下而上构造了一棵树　如对于文法G_{exp}，关于 $v*(v+d)$ 的一个归约过程可以认为是构造了如下一棵树：



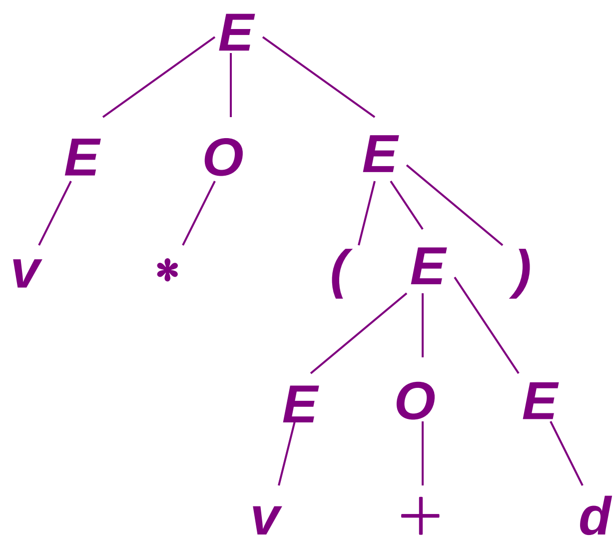
(1) $E \to EOE$
(2) $E \to (E)$
(3) $E \to v$
(4) $E \to d$
(5) $O \to +$
(6) $O \to *$

$$v*(v+d) \xrightarrow{(4)} v*(v+E) \xrightarrow{(6)} vO(v+E) \xrightarrow{(3)} vO(E+E)$$

$$\xrightarrow{(5)} vO(EOE) \xrightarrow{(1)} vO(E) \xrightarrow{(2)} vOE \xrightarrow{(3)} EOE \xrightarrow{(1)} E$$

✧ 语法分析树

– 推导过程自上而下构造了一棵树　如对于文法G_{exp}，关于 $v*(v+d)$ 的一个推导过程可以认为是构造了如下一棵树：



(1) $E \rightarrow EOE$

(2) $E \rightarrow (E)$

(3) $E \rightarrow v$

(4) $E \rightarrow d$

(5) $O \rightarrow +$

(6) $O \rightarrow *$

$$E \xrightarrow{(1)} EOE \xrightarrow{(6)} E*E \xrightarrow{(2)} E*(E) \xrightarrow{(3)} v*(E)$$

$$\xrightarrow{(1)} v*(EOE) \xrightarrow{(5)} v*(E+E) \xrightarrow{(3)} v*(v+E) \xrightarrow{(4)} v*(v+d)$$

✧ 语法分析树（*parse trees*）

对于 CFG $G = (V_N, V_T, P, S)$，语法分析树是满足下列条件的树：

(1) 每个内部结点由一个非终结符标记.

(2) 每个叶结点或由一个非终结符，或由一个终结符，或由 ε 来标记. 但标记为 ε 时，它必是其父结点唯一的孩子.

(3) 如果一个内部结点标记为 A，而其孩子从左至右分别标记为$X_1, X_2, \ldots, X_k$，则 $A \rightarrow X_1X_2\ldots X_k$ 是 P 中的一个产生式. 注意：只有 $k=1$ 时上述 X_i才有可能为ε，此时结点 A 只有唯一的孩子，且 $A \rightarrow \varepsilon$ 是P 中的一个产生式.

✧ 语法分析树（*parse trees*）

– 语法分析树的果实（ *yield* ）

设 CFG $G = (V_N, V_T, P, S)$. 将语法分析树的每个叶结点按照从左至右的次序连接起来，得到一个 $(V_N \cup V_T)^*$ 中的字符串，称为该语法树的果实.

G 的每个句型都是某个根结点为 S 的分析树的果实；这些分析树中，有些树的果实为句子，所有这些分析树的果实构成了 G 的语言.

✧ 归约、推导与分析树之间关系

设 CFG $G=(V_N, V_T, P, S)$. 以下命题是相互等价的:

(1) 字符串 $w \in T^*$ 可以归约（递归推理）到非终结符 A;

(2) $A \stackrel{*}{\Rightarrow} w$;

(3) $A \underset{lm}{\stackrel{*}{\Rightarrow}} w$;

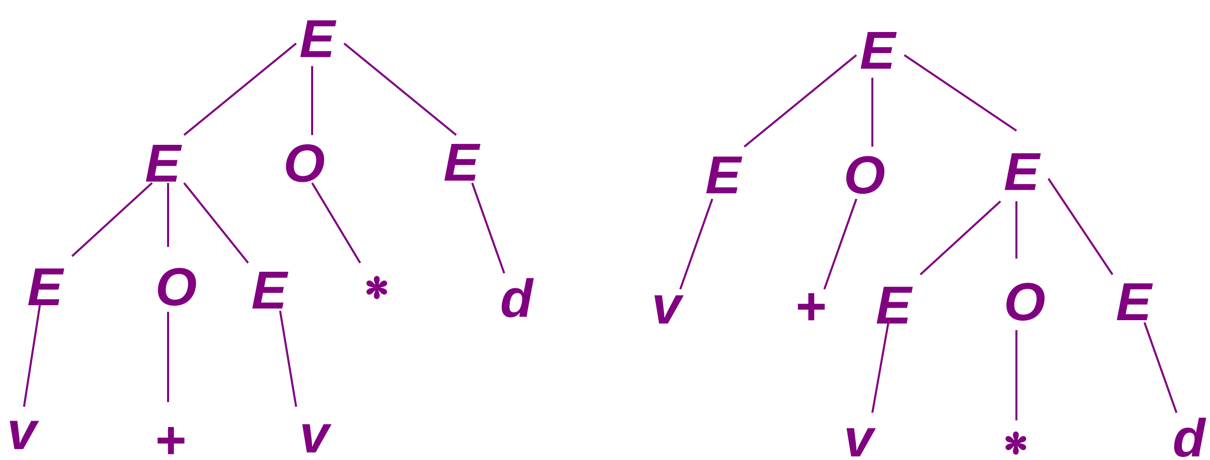
(4) $A \underset{rm}{\stackrel{*}{\Rightarrow}} w$;

(5) 存在一棵根结点为 A 的分析树，其果实为 w.

✧ 文法的二义性

– 二义文法（*ambiguous grammars*）举例

考虑右下文法，对于终结符串 $v + v * d$，存在两棵不同的分析树，它们的根结点都为开始符号 E，果实都为 $v + v * d$.



(1) $E \rightarrow EOE$

(2) $E \rightarrow (E)$

(3) $E \rightarrow v$

(4) $E \rightarrow d$

(5) $O \rightarrow +$

(6) $O \rightarrow *$

✧ 文法的二义性

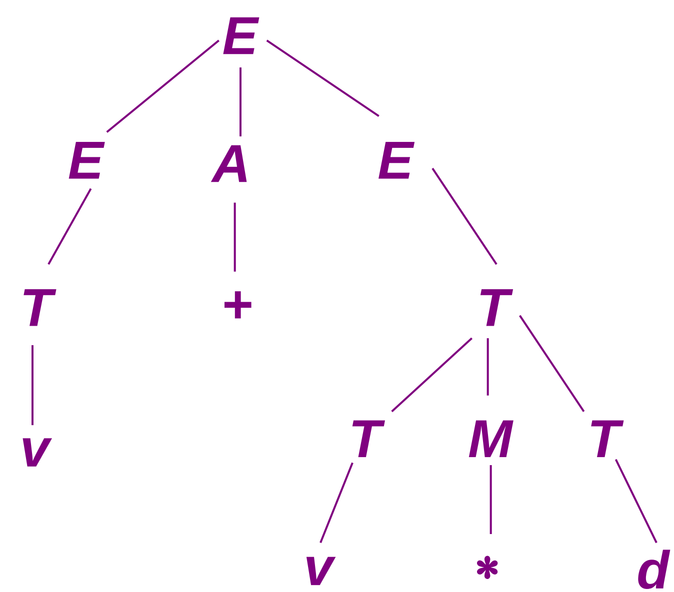
- 二义文法概念　CFG　$G = (V_N, V_T, P, S)$ 为二义的，如果对某个 $w \in T^*$，存在两棵不同的分析树，它们的根结点都为开始符号 S，果实都为 w. 如果对每一 $w \in T^*$，至多存在一棵这样的分析树，则 G 为无二义的.
- 二义性的判定　一个 CFG 是否为二义的问题是不可判定的，即不存在解决该问题的算法.
- 消除二义性　没有通用的办法可以消除文法的二义性. 但在实践中，对于常用的文法，可以找到特定的消除歧义性的办法.

✧ 消除二义性的几种文法变换方法

– 对于右上图的文法，采用**算符优先级联**方法将其变换为左下图的文法，对于该文法，串 $v+v*d$ 存在唯一的分析树.

$$E \to EOE \mid (E) \mid v \mid d$$
$$O \to + \mid *$$

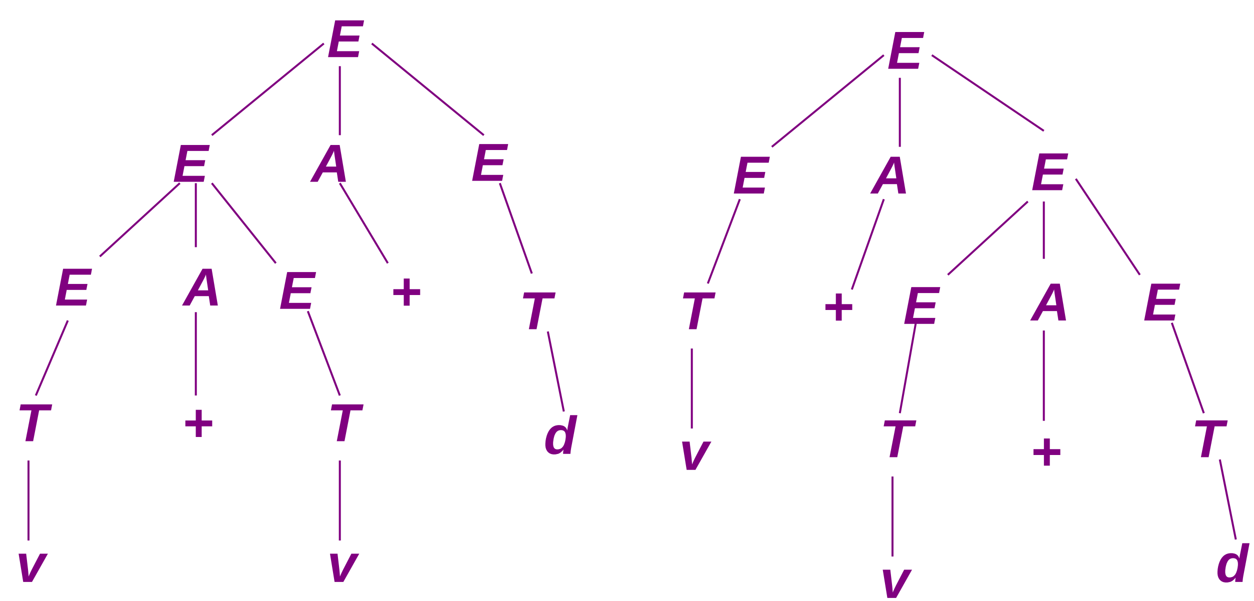
$$E \to E\,A\,E \mid T$$
$$T \to T\,M\,T \mid (\,E\,) \mid v \mid d$$
$$A \to +$$
$$M \to *$$



✧ 消除二义性的几种文法变换方法

– 右上图的文法仍然是二义文法，串 $v+v+d$ 存在不同的分析树（下图）.

$$
\begin{aligned}
E &\rightarrow E\,A\,E \mid T \\
T &\rightarrow T\,M\,T \mid (E) \mid v \mid d \\
A &\rightarrow + \\
M &\rightarrow *
\end{aligned}
$$

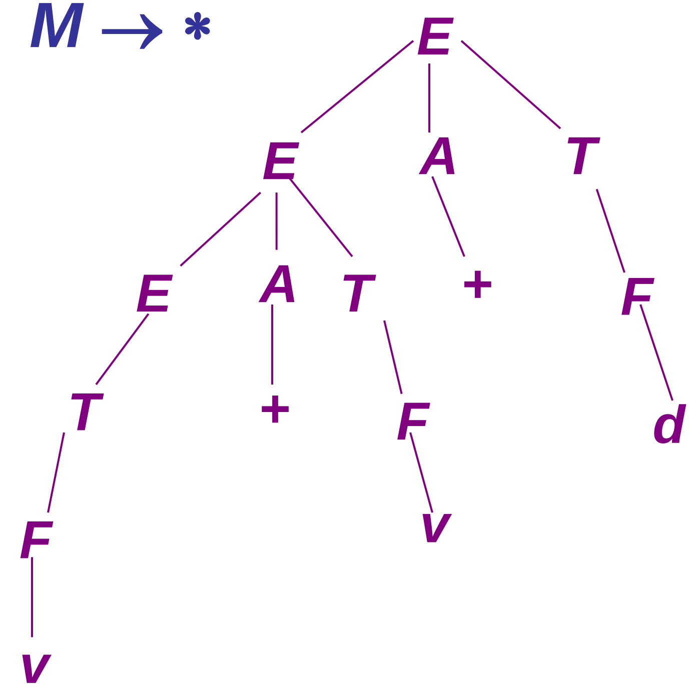


✧ 消除二义性的几种文法变换方法

$$E \to E\,A\,E \mid T$$
$$T \to T\,M\,T \mid (E) \mid v \mid d$$
$$A \to +$$
$$M \to *$$

– 采用左结合方法将右上图的文法变换为左下图，串 $v+v+d$ 存在唯一的分析树（右下图）.

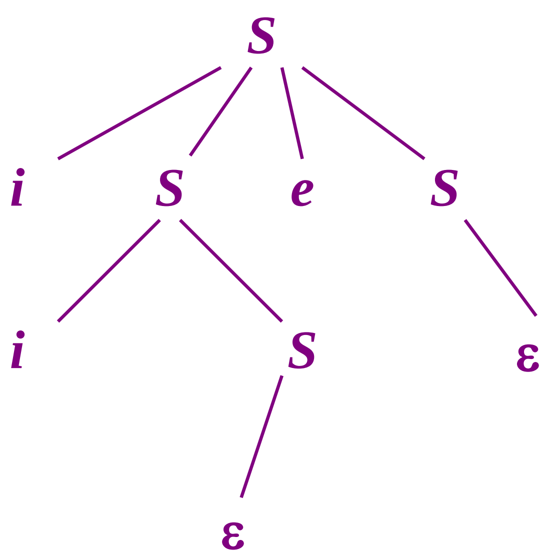
$$E \to E\,A\,T \mid T$$
$$T \to T\,M\,F \mid F$$
$$F \to (E) \mid v \mid d$$
$$A \to +$$
$$M \to *$$

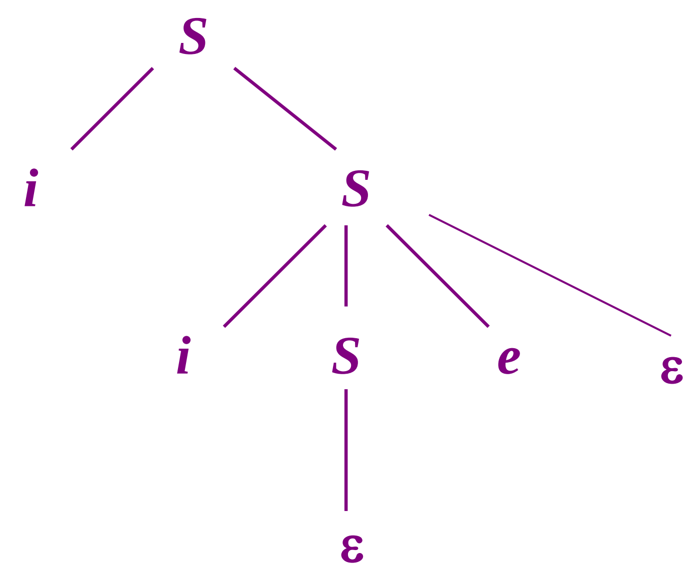


✧ 消除二义性的几种文法变换方法

– 悬挂else二义性

$S \rightarrow \varepsilon \mid i\,S \mid i\,S\,e\,S$　　　　$i\ i\ e$





✧ 消除二义性的几种文法变换方法

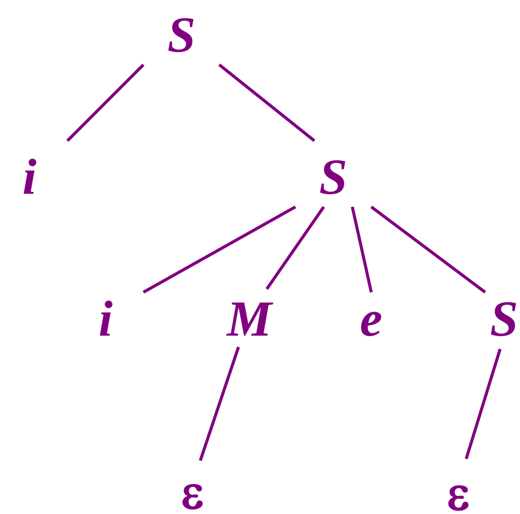
– 采用最近嵌套匹配方法消除悬挂 else 二义性

$$S \rightarrow \varepsilon \mid iS \mid iSeS$$

将右上部的文法变换为下面的文法

$$S \rightarrow \varepsilon \mid iS \mid iMeS$$
$$M \rightarrow \varepsilon \mid iMeM$$

串 *iie* 存在唯一的分析树（右图）



课后练习

1　试构造接受下列语言的一个 DFA：

$L = \{ w \mid w \in \{a, b\}^*$，$w$ 中 a 的数目可被 2 整除、b 的数目可被 3 整除 $\}$

2　设字母表为 $\{a,b\}$，试给出下列语言的正规表达式各一个：

(1)　$\{ w \mid w$ 至少包含 2 个 'a' $\}$

(2)　$\{ w \mid w$ 包含偶数个 'a' $\}$

(3)　$\{ w \mid w \in \{a, b\}^*$，w 的长度可被 3 整除 $\}$

课后练习

3 构造产生如下语言的上下文无关文法：

(1) $\{a^n b^{2n} c^m \mid n, m \geq 0\}$

(2) $\{a^n b^m c^{2m} \mid n, m \geq 0\}$

(3) $\{ a^m b^n \mid m \geq n \}$

(4) $\{ a^m b^n c^p d^q \mid m+n = p+q \}$

(5) $\{ uawb \mid u,w \in \{a, b\}^* \wedge |u| = |w| \}$

4 给出语言 $\{ a^m b^n \mid m \geq 2n \geq 0 \}$ 的二义文法和非二义文法各一个

5 适当变换文法，找到下列文法所定义语言的一个无二义的文法：$S \rightarrow SaS \mid SbS \mid ScS \mid d$

That's all for today.

Thank You