

بنام خداوند بخشنده و مهربان

# نظریه زبان ها و ماشین ها

# فصل اول

مقدمه ای بر نظریه محاسبات

# فصل اول

- **مجموعه (Set) :** دسته ای از عناصر بدون هیچ ساختاری، صرفاً عضو مجموعه هستند.
- **نمایش مجموعه:**  $\{ \text{زوج } i, \{a,b,\dots\} : i > 0 \}$  ; مجموعه جهانی ( $U$ ) و مجموعه تهی ( $\emptyset$ )
- **عملگرهای مجموعه :** اجتماع، اشتراک، تفاضل، متمم و کاربرد آنها با یکدیگر ( قوانین دمورگان).
- **مفاهیم اساسی مجموعه ها:** زیر مجموعه بودن؛ مجموعه های مجزا یا جدا از هم (*Disjoint*)؛ زیر مجموعه توانی (*Powerset*).

# فصل اول

• مثال :  $S = \{a, b, c\}$  . در صورتی که  $S$  مجموعه محدود و متناهی باشد، داریم:

: مجموعه توانی  $2^S = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$

: کاردینالیتی  $|S| = 3, |2^S| = 8 \rightarrow |2^S| = 2^{|S|}$

• حاصلضرب دکارتی:

$$S_1 \times \dots \times S_n = \{(x_1, \dots, x_n) \mid x_i \in S_i, i: 1 \dots n\};$$

$$S = S_1 \times S_2 = \{(x, y) \mid x \in S_1, y \in S_2\}$$

# فصل اول

- **تابع (Function) :**  $F : S_1 \rightarrow S_2$  ; دامنه و برد (*Domain & Range*)؛ زوج های مرتب یکی از روش‌های نمایش تابع .
- اگر دامنه تابع  $F$  مساوی  $S_1$  باشد به آن تابع کلی یا تام (*Total Function*) می‌گویند و در غیر این صورت تابع جزئی (*Partial Function*) می‌گویند.
- **رابطه (Relation) :** تابع مجموعه‌ای از زوج های مرتب است که زوج های اول آنها فقط یکبار تکرار شده‌اند. اگر چنین نباشد، مجموعه را رابطه گویند (کلی تر از تابع).
- **رابطه هم ارزی :**  $X \equiv Y$  ; خاصیت‌های انعکاسی ، تقارن ، تعدی

# فصل اول

- مثال : رابطه هم ارزی را بررسی کنید؟ رابطه هم ارزی است.

$$X \equiv Y$$

اگر و فقط اگر

$$x \bmod 3 = y \bmod 3$$

- گرافها و درختها :

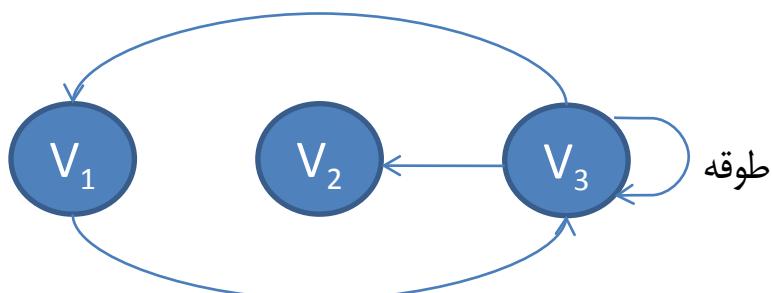
گراف ( $G(V,E)$ ) : گراف های جهت دار ; گراف های برچسبدار .

- مفاهیم :

راه (*Path*) ، جاده ، مسیر (*Walk*)

↓  
مسیر ساده

↓  
مسیر



# فصل اول

## • چرخه با پایه $V_i$ : (Cycle)

در یک چرخه با پایه  $V_i$ ، اگر به جز پایه هیچ رأسی تکرار نشود، چرخه را چرخه ساده گویند.  
(در چرخه یال، تکراری نباید وجود داشته باشد).

## • درخت:

گراف جهت داری که چرخه ندارد؛ ریشه (Root) (سطح ریشه صفر)، و  
ارتفاع (Height).

. درختهای باینری (Binary tree)؛ والد (Parent)، بچه (Child) و (Left & Right Child).

# فصل اول

- روش های اثبات : استقراء، برهان خلف.

- استقراء : (1)  $p(n)$  درست، فرض  $p(n+1)$  حکم

- مثال :

ثابت کنید در درختهای باینری با ارتفاع  $n$  دارای  $2^n$  برگ خواهیم بود؟ ( به روش استقراء )

$n = 0$  درست  $\rightarrow$  ریشه  $\rightarrow Len(0) = 1 = 2^0$  ارتفاع درخت

:  $Len(i) = 2^i$  ،  $i = 0, 1, \dots, n$  استقراء فرض

:  $Len(i+1) = 2^{i+1}$  ،  $i = 0, 1, \dots, n$  استقراء حکم

# فصل اول

برای ساختن یک درخت باینری با ارتفاع  $n+1$  از درختی با ارتفاع  $n$ ، میتوانیم حداقل دو برگ به هر برگ درخت با ارتفاع  $n$ ، اضافه نماییم.

$$(n+1) = 2 \cdot 2 = 2 \rightarrow Len \rightarrow Len(n+1) = 2 Len(n)$$

• **مثال :** ثابت کنید که  $\sqrt{2}$  عدد گویا نیست؟ (به روش برهان خلف)

عدد گویا عددی است که میتواند بصورت یک عدد کسری نوشته شود بطوریکه صورت و مخرج آن عامل مشترک (Common Factor) نداشته باشد.

برهان خلف : عکس آنچه که می خواهیم اثبات کنیم.

# فصل اول

$$\sqrt{2} = \frac{n}{m}$$
 عددگویا فرض

$$\sqrt{2} = \frac{n}{m} \rightarrow \sqrt{2}m = n \rightarrow \underbrace{2m^2}_{\text{همواره زوج}} = n^2$$

زوج  $n$  → زوج  $n^2$  دارای عامل های مشترک نیستند

$$\rightarrow n = 2k \rightarrow n^2 = 4k^2 \rightarrow 2m^2 = 4k^2$$

$$\rightarrow m^2 = \underbrace{2k^2}_{\text{همواره زوج}} \rightarrow m^2 \text{ زوج} \rightarrow m \text{ زوج}$$

دارای عامل های مشترک میباشند → زوج  $n, m$

عدد گویا نیست  $\sqrt{2}$  → نقض فرض اولیه

# فصل اول

- مفاهیم اساسی در تئوری محاسبات :  
زبانها ، گرامرها ، آتماتا (ماشین ها).
- زبانها :  
مفهوم پایه : نشانه های الفبا *String* ; *Alphabet* (رشته دنباله متناهی از نشانه های الفباست) ; الحاق رشته ها (*Concatenation*) .  
 $\Sigma = \{a,b\} \rightarrow String : abab, bba, aa, \dots$

- قوانین در رشته ها :  
 $w = a_1 a_2 \dots a_n , v = b_1 b_2 \dots b_m \rightarrow u = wv = a_1 \dots a_n b_1 \dots b_m$   
(اثبات توسط استقراء) طول رشته :  $|w| = n$  ;  $|w^k| = k|w|$   
 $w^R = a_n \dots a_2 a_1$  : معکوس رشته  
 $(uv)^R = v^R u^R \quad u, v \in \Sigma^+$   
(اثبات توسط استقراء)  
 $(w^R)^R = w$

# فصل اول

ر ش ته ت هی  $\lambda$  ,  $|\lambda| = 0 \rightarrow \lambda w = w\lambda = w$

زیر ر ش ته پ سوند W → U ر ش ته (Prefix Substring) , V زیر ر ش ته پ سوند (Suffix Substring)

ت کر ار ر ش ته W ب ه ت ع داد n ب ا ر  $w^n = w \dots w ; (w^0 = \lambda)$

• اگر  $\Sigma$  یک الف باشد،  $\Sigma^*$  م جم وعه تمام ر ش ته هایی است که با اتصال صفر یا بیش تر از

ح رو ف  $\Sigma$  بدست آید. ( $\Sigma^* \in \lambda$ )

• گرچه  $\Sigma$  م تناهی است، ولی  $\Sigma^*$  نام تناهی هستند (عدم محدودیت روی طول ر ش ته ها).

$$\Sigma^+ = \Sigma^* - \{\lambda\}$$

# فصل اول

- یک زبان زیر مجموعه‌ای از  $\Sigma^*$  است. یک رشته در زبان  $L$ ، یک جمله (*Sentence*) می‌باشد.

• مثال :

$$\Sigma = \{a, b\} \rightarrow \text{مجموعه } \{a, aa, aab\} \text{ یک زبان متناهی}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, \dots\} \rightarrow L = \{a^n b^n, n \geq 0\}$$

- اکثر زبانهای قابل استفاده، متناهی هستند و چون زبانها مجموعه هستند، لذا عملیات اجتماع، اشتراک و تفاصل دو زبان قابل توصیف است.

# فصل اول

متهم یک زبان  $\bar{L} = \Sigma^* - L$

عكس یک زبان  $L^R = \{w^R : w \in L\}$  (مثال قبل)  $L^R = \{b^n a^n, n \geq 0\}$

اتصال دو زبان  $L_1 L_2 = \{xy | x \in L_1, y \in L_2\}$   $L^n = L \dots L$  (اتصال  $n$  بار)

: نکته  $(L_1 L_2)^R = L_2^R L_1^R$

$L^0 = \{\lambda\}, L^1 = L$

$L^* = L^0 \cup L^1 \cup L^2 \cup \dots$

$L^+ = L^1 \cup L^2 \cup \dots$

# فصل اول

## • گرامرها :

تعریف صوری یا رسمی گرامر (Formal Definition) :

گرامر  $G = (V, T, S, P)$

مجموعه متناهی از قوانین مجموعه متناهی از متغیرها

$S \in V$  : (Starting Variable)  
یک متغیر خاص به نام نشانه شروع  
مجموعه متناهی از نشانه ها به نام ترمینال

بایستی فرض شود که مجموعه های  $T$  و  $V$  غیر تهی و مجزا ( جدا از هم ) می باشند.

قوانین :  $Y \in (VUT)^*$  ,  $X \in (VUT)^+$  ,  $X \rightarrow Y$

قوانین را میتوان به تعداد دلخواه استفاده کرد تا رشته مورد نظر بدست آید.

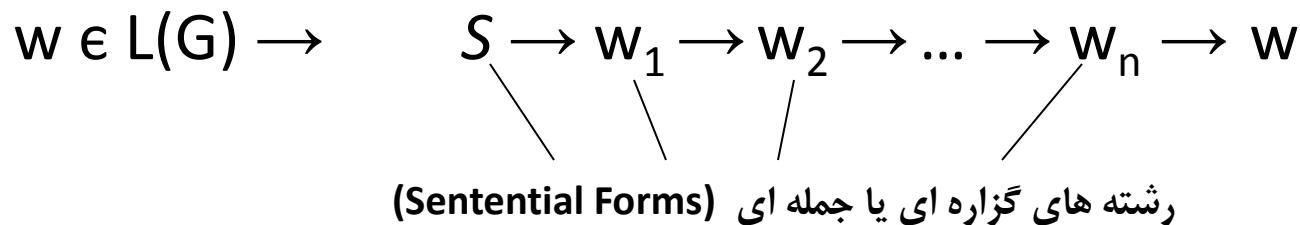
$W_1 \rightarrow W_2 \rightarrow \dots \rightarrow W_n$  (می گوییم  $W_n$  از  $W_1$  مشتق شده است.)

\* : به معنای تعداد نامشخص حتی صفر یا  $W_1 \xrightarrow{*} W_n$

# فصل اول

- تعریف دیگری از گرامر :
- اگر  $G(V, T, S, P)$  گرامر باشد، آنگاه مجموعه زبان  $L(G)$  تولید شده توسط گرامر  $G$  است.

$$L(G) = \left\{ w \in T^* : S \xrightarrow{*} w \right\}$$



# فصل اول

• مثال : گرامر  $G$  به شکل زیر وجود دارد. زبان آنرا بیابید؟

$$G = (\{S\}, \{a, b\}, S, P)$$

$$P : S \rightarrow aSb \mid \lambda$$

$$L(G) = \{a^n b^n : n \geq 0\}$$

$$S \xrightarrow{1} aSb \xrightarrow{1} aaSbb \xrightarrow{1} aaaSbbb \xrightarrow{1} \dots \quad a^n S b^n \xrightarrow{\lambda} a^n b^n$$

• مثال : گرامری بسازید که  $L = \{a^n b^{n+1} : n \geq 0\}$  را تولید کند؟

$$G = (\{S, A\}, \{a, b\}, S, P)$$

/ نسبت به گرامر قبلی یک  $b$  اضافه دارد \*/

$$P \left[ \begin{array}{l} S \rightarrow Ab \\ A \rightarrow aAb \\ A \rightarrow \lambda \end{array} \right]$$

# فصل اول

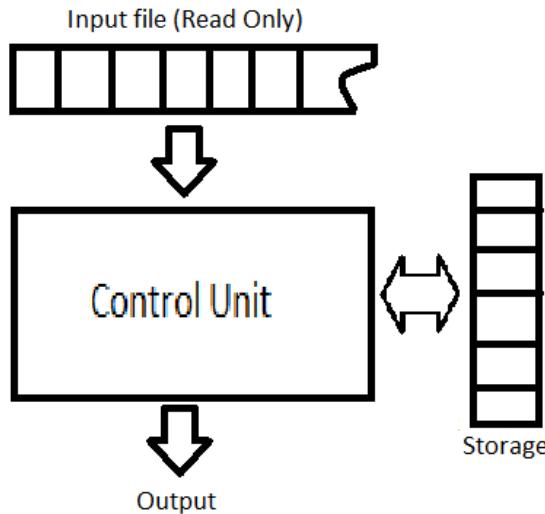
- مثال:  $\Sigma = \{a,b\}$  و  $n_a(w) = n_b(w)$  است و گرامر  $G$

به شکل زیر وجود دارد:

$$P : \begin{cases} S \rightarrow SS \\ S \rightarrow \lambda \\ S \rightarrow aSb \\ S \rightarrow bSa \end{cases} \longrightarrow L(G) = \{w : n_a(w) = n_b(w)\}$$

- میتوان به کمک استقراء ثابت کرد که  $L$  زبان تولید شده توسط گرامر  $G$  میباشد.

# فصل اول



## • آتماتا (Automata):

• آتماتا : مدل انتزاعی از کامپیوتر که دارای مکانیزم زیر است :

1. ورودی : رشته الفبا ؛ در فایل ورودی (هر نشانه در هر سلول)؛ تشخیص انتهای ورودی.

2. حافظه موقت (Storage): تعداد نامتناهی سلول و در داخل هر سلول یک نشانه (قابل خواندن و نوشتن).

3. واحد کنترل : هر زمان می تواند در یکی از وضعیتهاي داخلی (Internal States) باشد. تعداد وضعیت های نامتناهی است و آتماتون در هر زمان در یک وضعیت خاص است و حروف خاص را از ورودی می خواند. وضعیت بعدی توسط تابع تغییر وضعیت (Transition Function) تعیین می گردد و باعث خروجی یا تغییر در حافظه موقت می شود.

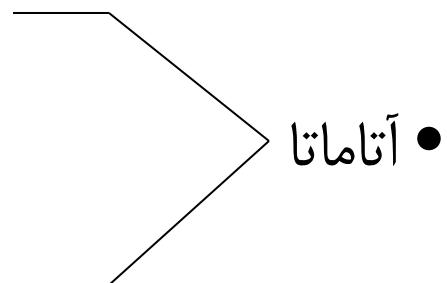
# فصل اول

- آتماتای متناهی معین و نامعین :  
*(Deterministic & Nondeterministic Finite Automata – DFA & NFA)*

در آتماتای معین اگر وضعیت فعلی، ورودی، و محتوای حافظه را بدانیم، میتوان رفتار بعدی آتماتا را تعیین کرد ولی در آتماتای نامعین اینگونه نیست و در آتماتای نامعین می تواند در هر زمان چند حرکت مختلف داشته باشد.

پذیرنده (*Acceptor*) : آتماتایی که خروجی آن قبول یا عدم قبول است. (ورودی خود را قبول یا رد می کند)

تراگذار (*Transducer*) : آتماتایی که خروجی آن بصورت رشته است.



- آتماتا

# فصل اول

## • کاربرد ۱ :

برای زبان های برنامه نویسی ، کامپایلر نوشته می شود. کامپایلر نیاز به تعریف دقیق و رسمی آن زبان برنامه نویسی دارد. لذا می توان در زبان های برنامه نویسی ، از گرامر یا آتماتا برای پذیرش یا عدم پذیرش یک قطعه کد توسط آن زبان برنامه نویسی استفاده نمود.

# فصل اول

: *Acceptor* مثال

گرامر و آتماتای مربوط به مجموعه متغیرهای زبان پاسکال را بررسی نمایید؟

$<id> \rightarrow <Letter><A> | <Underline><A>$

$<A> \rightarrow <Letter><A> | <Digit><A> | <Underline><A> | \lambda$

$<Letter> \rightarrow A | B | \dots | Z | a | b | \dots | z$

$<Digit> \rightarrow 0 | 1 | \dots | 9$

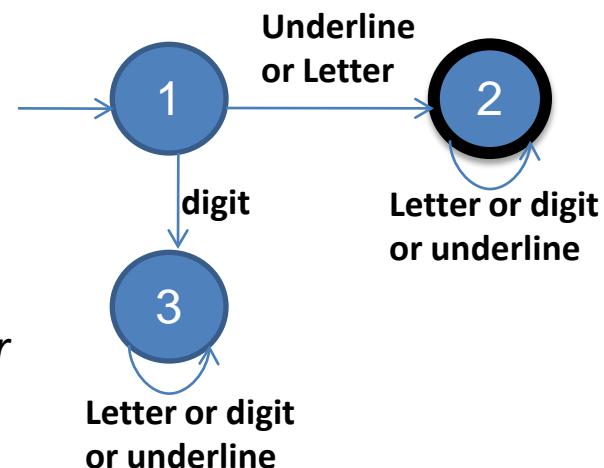
$<Underline> \rightarrow _$

اشتقاق مربوط به متغیر  $a_0$  : مثلاً

States :

- 1) Start
- 2) Acceptor
- 3) Not Acceptor

« آتماتای پذیرنده »



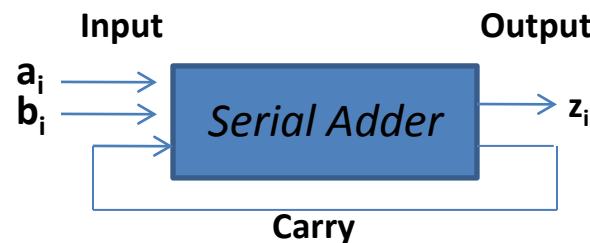
# فصل اول

## کاربرد ۲ :

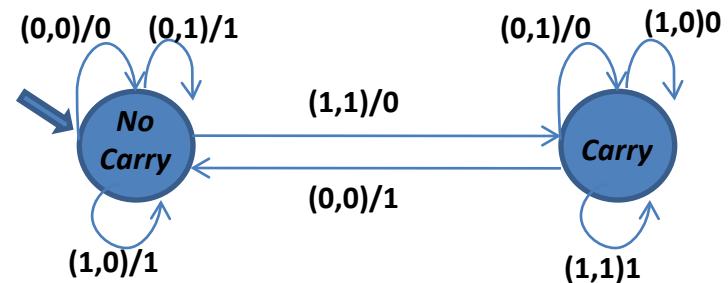
طراحی رقمی ؛ آتماتا در طراحی رقمی ، توصیف علمی بسیار سطح بالایی از یک مدار منطقی و پیاده سازی منطقی آن با ترااتریستور ، گیت و فلیپ فلاپ است.

### • مثال : *Transducer*

در جمع کننده باینری (*Full Adder*)، برای سادگی کار، فرض بر فقط اعداد مثبت است.



$$\frac{\text{گراف}}{(a_i, b_i)/z_i} \rightarrow \text{یال}$$



«آتماتای تراگذار»

قوانين و گرامر

$$\begin{cases} 00 \rightarrow 0 \\ 01 \rightarrow 1 \\ 10 \rightarrow 1 \\ 11 \rightarrow 10 \end{cases}$$

# فصل اول

## • تمرین های مهم فصل اول :

(۱) گرامرهايی روی  $\Sigma = \{a,b\}$  پیدا کنيد بطور يك:

(الف)  $\left[ \begin{array}{l} S \rightarrow AaA \\ A \rightarrow bA|\lambda \end{array} \right] \text{ یا } S \rightarrow a|Sb|bS$

الف) تمام رشته هایی که فقط یک a دارند؟

(ب)  $\left[ \begin{array}{l} S \rightarrow AaA \\ A \rightarrow Aa|Ab|\lambda \end{array} \right] \text{ یا } S \rightarrow a|aS|bS|ba$

ب) تمام رشته هایی که حداقل یک a دارند؟

(ج)  $\left[ \begin{array}{l} S \rightarrow BABABAB \\ A \rightarrow a|\lambda \\ B \rightarrow bB|\lambda \end{array} \right]$

ج) تمام رشته هایی که حداکثر سه a دارند؟

# فصل اول

(۲) برای هر یک از زبانهای زیر ، گرامری پیدا کنید ؟

الف)  $\begin{cases} S \rightarrow AbB \\ A \rightarrow aAb | \lambda \\ B \rightarrow bB | \lambda \end{cases}$  یا  $S \rightarrow aSb | B$   $B \rightarrow bB | b$   $L_1 = \{a^n b^m : n \geq 0, m > n\}$  الف )

$L_1^3 = \{a^n b^m a^p b^q a^x b^y : n, m, p, q, x, y \geq 0, m > n, q > p, y > x\}$  ( ب )

ب)  $\begin{cases} S \rightarrow AAA \\ A \rightarrow aAb | bB \\ B \rightarrow bB | \lambda \end{cases} \longrightarrow \begin{cases} S \rightarrow CCC \\ C \rightarrow AbB \\ A \rightarrow aAb | \lambda \\ B \rightarrow bB | \lambda \end{cases}$

ج)  $\begin{cases} S \rightarrow \overbrace{SS}^{SAS} | A | \lambda \\ A \rightarrow aAb | bB \\ B \rightarrow bB | \lambda \end{cases} L_1^* ( ج )$

# فصل اول

(۳) گرامر زبان  $L$  را روی  $\Sigma = \{a\}$  پیدا کنید.

$$\begin{cases} S \rightarrow aB | aaB | A \\ B \rightarrow aaaB | \lambda \\ A \rightarrow aaaaaaA | \lambda \end{cases}$$

- $w = 0 \rightarrow 0 \bmod 3 \geq 0 \bmod 2 \quad \checkmark$
- $w = 1 \rightarrow 1 \bmod 3 \geq 1 \bmod 2 \quad \checkmark$
- $w = 2 \rightarrow 2 \bmod 3 \geq 2 \bmod 2 \quad \checkmark$
- $w = 3 \rightarrow 3 \bmod 3 \geq 3 \bmod 2 \quad \times$
- $w = 4 \rightarrow 4 \bmod 3 \geq 4 \bmod 2 \quad \checkmark$
- $w = 5 \rightarrow 5 \bmod 3 \geq 5 \bmod 2 \quad \checkmark$
- $w = 6 \rightarrow 6 \bmod 3 \geq 6 \bmod 2 \quad \checkmark$

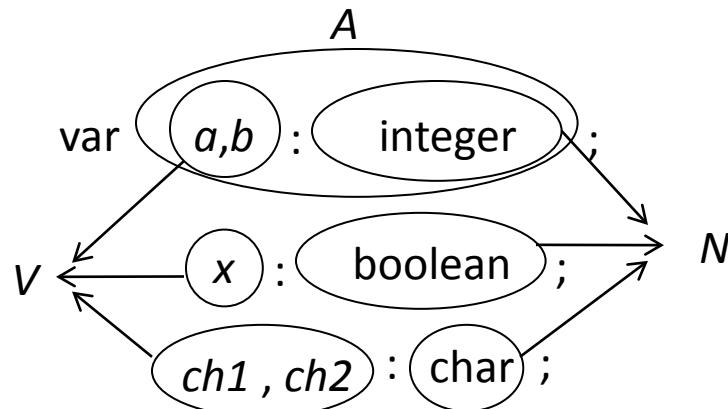
بایستی به صورتهای  $3k+2$  و  $3k+1$  و  $3k$  باشد.

$$\begin{array}{ccc} w = 2 & w = 1 & w = 0 \\ w = 5 & w = 4 & w = 6 \\ w = 7 & w = 12 & \end{array}$$

# فصل اول

۴) یک گرامر برای اعلان کلیه متغیرها در زبان پاسکال بنویسید ؟

```
var a , b : integer ;
x : boolean ;
ch1 , ch2 : char ;
```



$$\begin{aligned}
 S &\rightarrow \text{var } Space \ A \\
 A &\rightarrow V : N ; \mid AA \\
 N &\rightarrow \text{integer} \mid \text{real} \mid \text{double} \mid \text{char} \mid \text{boolean} \mid \dots \\
 V &\rightarrow V, V \mid BV_1 \\
 V_1 &\rightarrow CV_1 \mid BV_1 \mid \lambda \\
 B &\rightarrow a \mid b \mid \dots \mid z \\
 C &\rightarrow 0 \mid 1 \mid \dots \mid 9 \\
 Space &\rightarrow \emptyset
 \end{aligned}$$

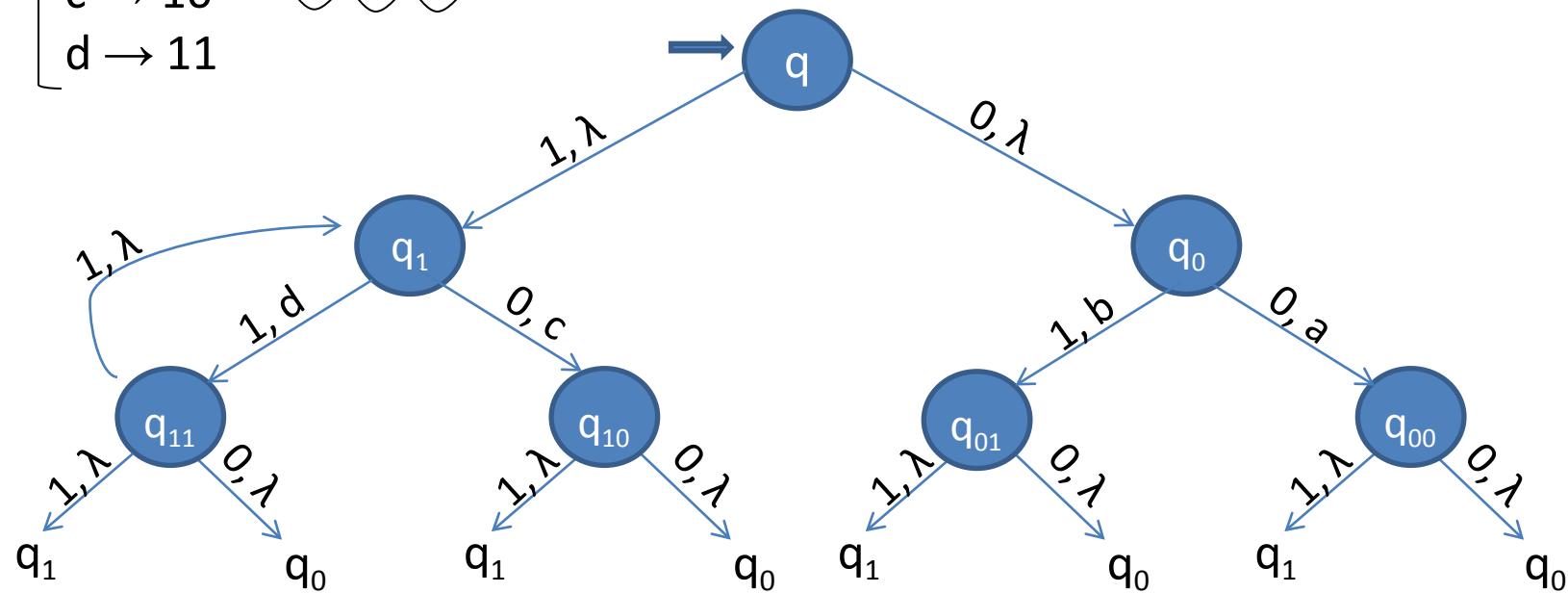
# فصل اول

۵) یک تراگزار برای رمزگشایی رشته ها روی  $\{0,1\}$  به پیام اصلی بسازید؟

رمزگشایی کدهای عددی به حرفی (ASCII). دو الفبای  $\{a,b,c,d\}$  و  $\{0,1\}$  را در نظر گرفته که یک رمز گذاری به صورت زیر دارد:

$$\begin{cases} a \rightarrow 00 \\ b \rightarrow 01 \\ c \rightarrow 10 \\ d \rightarrow 11 \end{cases}$$

$010011 \rightarrow \text{bad}$



# فصل دوم

آتماتای متناهی

# فصل دوم

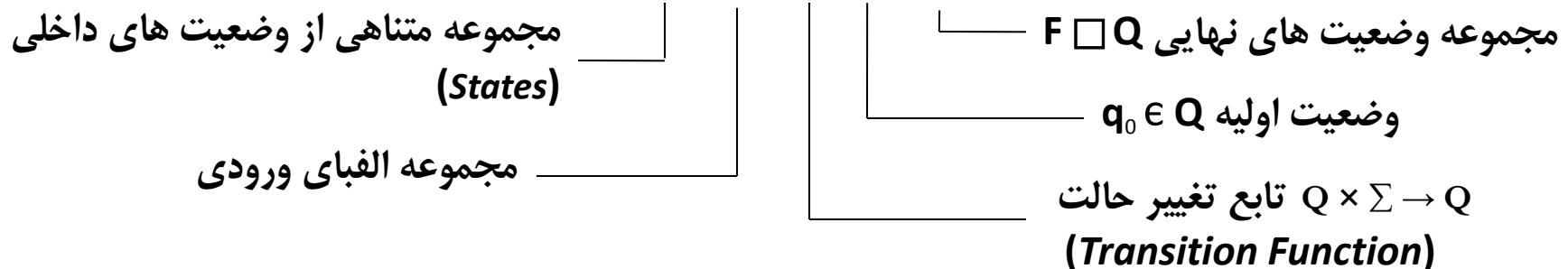
## • آتماتای متناهی - FA :

آتماتای متناهی (*Finite Automata*) دارای حافظه موقت نیست. بنابراین در به یادآوردن اطلاعات در طول محاسبه مشکل داشته و مقدار کمی اطلاعات می تواند در واحد کنترل نگهداری شود که آن هم بصورت قرار گرفتن در یک وضعیت انجام می شود.

## • تعریف رسمی یک پذیرنده متناهی معین - DFA :

DFA : *Deterministic Finite Acceptor*

$$M = (Q, \Sigma, \delta, q_0, F)$$



## فصل دوم

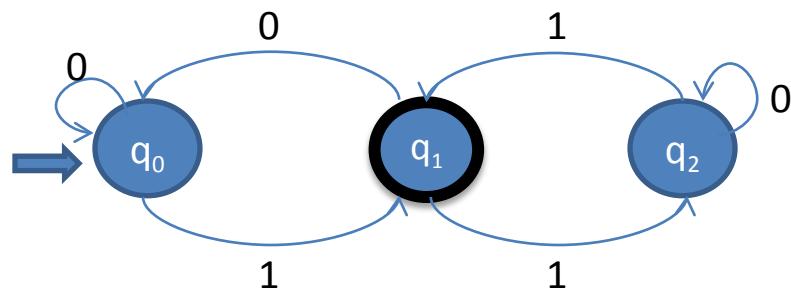
$$M = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\})$$

• مثال :

$$\delta : \begin{cases} \delta(q_0, 0) = q_0, \quad \delta(q_0, 1) = q_1 \\ \delta(q_1, 0) = q_0, \quad \delta(q_1, 1) = q_2 \\ \delta(q_2, 0) = q_2, \quad \delta(q_2, 1) = q_1 \end{cases} \longrightarrow$$

در برنامه نویسی : استفاده از آرایه های  
دو بعدی برای وضعیت اول بر حسب  
ورودی ها و وضعیت نهایی

- نمایش یک DFA به صورت یک گراف تغییر وضعیت است که رأس های آن وضعیت ها و یال های آن حروف ورودی هستند و دارای  $|Q|$  رأس است.



## فصل دوم

- میتوان تابع تغییر وضعیت توسعه یافته (*Extended Transition Function*) را به صورت  $\delta^* : Q \times \Sigma^* \rightarrow Q$  تعریف نمود.

• تعریف بازگشتی :

$$\delta^*(q, \lambda) = q, \forall q \in Q, w \in \Sigma^*, a \in \Sigma$$

$$\delta^*(q, wa) = \delta(\delta^*(q, w), a)$$

مثالاً :

$\begin{cases} \delta(q_0, a) = q_1 \\ \delta(q_1, b) = q_2 \\ \delta^*(q_0, ab) = q_2 \end{cases}$	بنا به تعریف بازگشتی	$\begin{cases} \delta^*(q_0, ab) = \delta(\delta^*(q_0, a), b) \\ \delta^*(q_0, a) = \delta(\delta^*(q_0, \lambda), a) = \delta(q_0, a) = q_1 \\ \delta^*(q_0, ab) = \delta(q_1, b) = q_2 \end{cases}$
---	----------------------	--

## فصل دوم

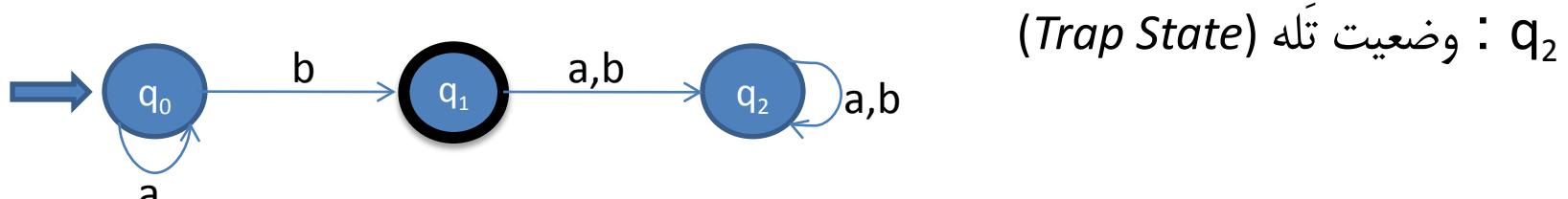
### • زبانهای DFA :

زبان یعنی مجموعه تمامی رشته هایی که توسط آتماتا پذیرفته میشود و زبانهای پذیرفته شده توسط DFA بنام  $M(Q, \Sigma, \delta, q_0, F)$  تمامی رشته ها بروی الفبای  $\Sigma$  است.

$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}$$

$$\overline{L(M)} = \{w \in \Sigma^* : \delta^*(q_0, w) \notin F\}$$

### • مثال : DFA زیر را در نظر بگیرید :



$$L(M) = \{a^n b : n \geq 0\}$$

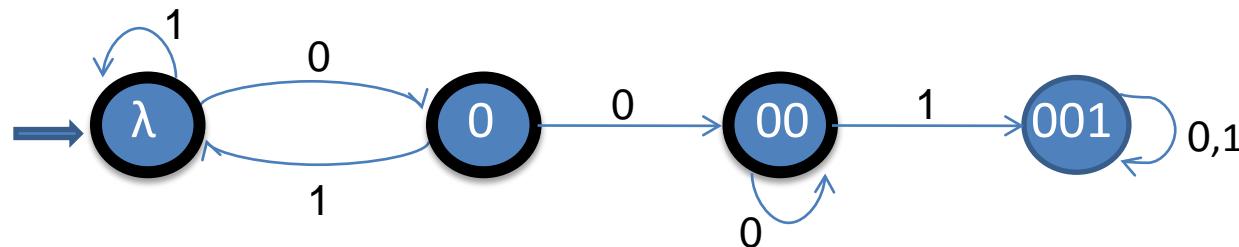
## فصل دوم

### • قضیه :

یک DFA بنام  $M = (Q, \Sigma, \delta, q_0, F)$  داریم و  $G_M$  گراف تغییر وضعیت مربوط به آن می‌باشد.

برای هر  $w \in \Sigma^*$  داریم  $\delta(q_i, w) = q_j$  اگر و فقط اگر یک راه با برچسب  $w$  از  $q_i$  به  $q_j$  وجود داشته باشد.

• مثال : یک DFA طراحی کنید که تمامی رشته های روی الفبای  $\{0,1\}$  را بجز رشته هایی که حاوی رشته "001" باشد بپذیرد ؟



• DFA بخاطر می‌آورد که ورودی قبلی چه بوده، اما نه از طریق حافظه، بلکه از طریق وضعیتی که در آن قرار دارد.

# فصل دوم

## • زبانهای منظم (*Regular Languages*) :

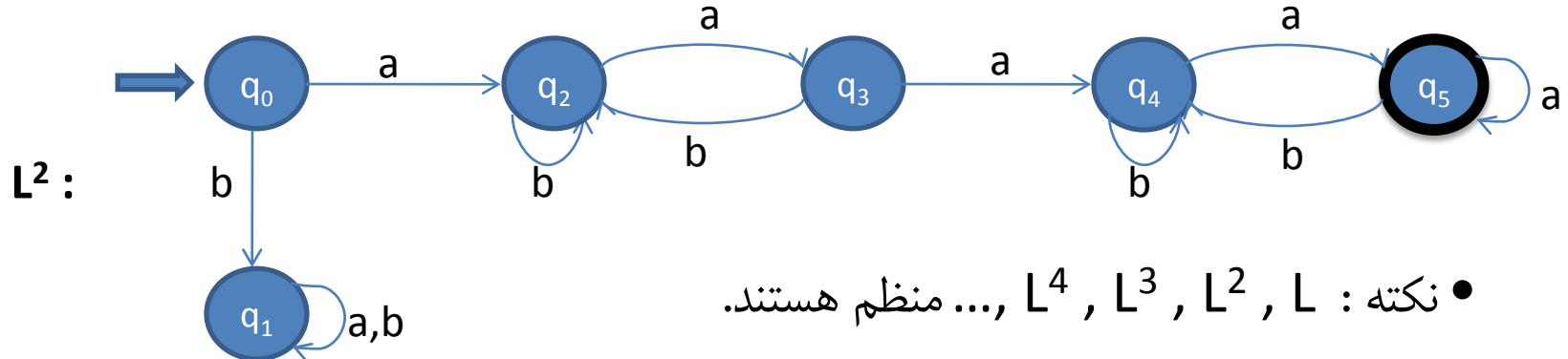
- هر DFA زبانی را می پذیرد. اگر تمام FA ها را در نظر بگیریم، یک مجموعه زبان در ارتباط با آنها خواهیم داشت که به این مجموعه یک خانواده (*Family*) می گویند.
- خانواده زبانهایی که توسط DFA به نام  $M$  وجود داشته باشد، زبان آن  $L(M)$  است.
- زبان  $L$  منظم است اگر و فقط اگر DFA بنام  $M$  باشد بطوریکه  $L = L(M)$ .

## فصل دوم

• مثال : نشان دهید که  $L^2 = \{awawa : w \in \{a,b\}^*\}$  منظم است ؟

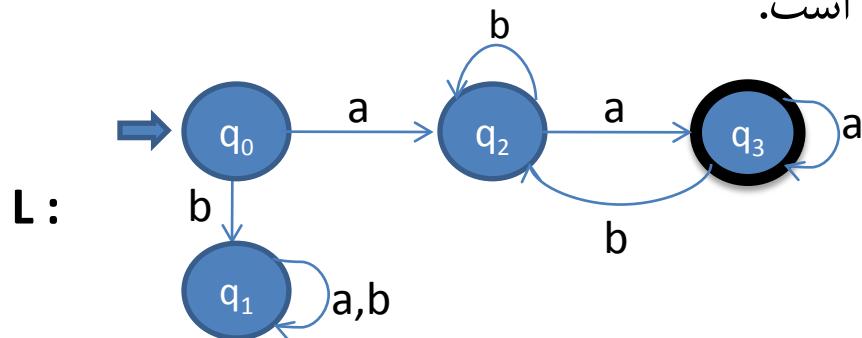
$$L = \{awa : w \in \{a,b\}^*\}$$

کافی است DFA آن را طراحی کنیم .



• نکته :  $L^4 , L^3 , L^2 , L$  : منظم هستند.

• اگر  $L$  منظم باشد، آنگاه  $\{\lambda\} - L$  نیز منظم است.



## فصل دوم

: (*Nondeterministic Finite Acceptor*) *NFA* – پذیرنده متناهی نامعین

$$M = (Q, \Sigma, \delta, q_0, F)$$

*DFA*                          همان تعریف

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$$

• تفاوت‌های *DFA* و *NFA* :

- (1) وضعیت بعدی در *NFA*؛ مثلاً  $\delta(q_1, a) = (q_0, q_2)$
  - (2) در *NFA*،  $\lambda$  به عنوان ورودی قابل قبول است و با ورودی  $\lambda$  تغییر وضعیت انجام می‌گیرد.
  - (3) در *NFA*، ممکن است مثلاً  $\delta(q_1, a)$  تهی باشد و هیچ تغییر وضعیتی برای  $a$  وجود ندارد.
- همانند *DFA*، *NFA* نیز با یک گراف تغییر وضعیت قابل نمایش بوده و میتواند رشته‌ای را پذیرد.

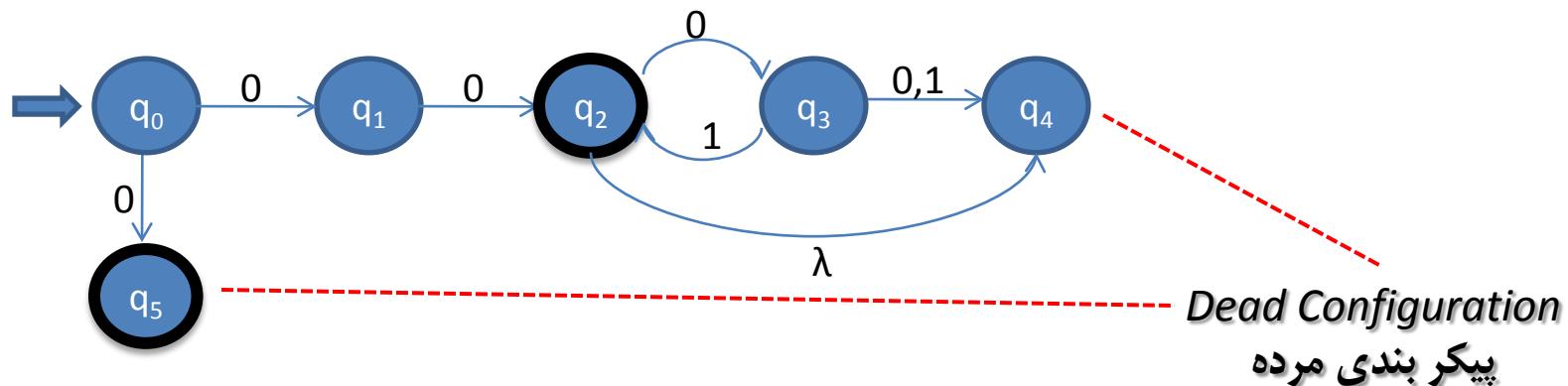
## فصل دوم

• مثال : یک NFA داریم که :

(1) دارای رشته ورودی  $\lambda$  می باشد.

$$\delta(q_0, 0) = (q_1, q_5) \quad (2)$$

$$\delta(q_1, 1) = \emptyset \quad (3)$$



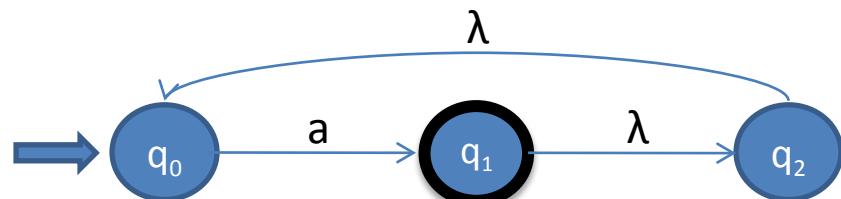
## فصل دوم

• مثال : يك NFA با تابع تغيير وضعیت مثلما

$$\delta^*(q_1, a) = \{q_0, q_1, q_2\}$$

$$\delta^*(q_2, \lambda) = \{q_0, q_2\}$$

$$\delta^*(q_2, aa) = \{q_0, q_1, q_2\}$$



$$\delta(q_0, a) = \delta(q_1, a) = \delta(q_2, a) = \{q_0, q_1, q_2\}$$

$$L(M) = \{a^n \mid n \geq 1\}$$

## فصل دوم

- زبان مورد قبول یک  $NFA$  بنام  $M = (Q, \Sigma, \delta, q_0, F)$  عبارتست از :

$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \cap F \neq \emptyset\}$$

$$\overline{L(M)} = \{w \in \Sigma^* : \delta^*(q_0, w) \cap F = \emptyset\}$$

- زبان  $NFA$  مربوط به مثال قبل چیست؟ رشته هایی بصورت  $a^n, n \geq 1$  را می پذیرد.  
یعنی :

$$L(M) = \{a^n \mid n \geq 1\} \rightarrow \overline{L(M)} = \{\lambda\}$$

- زبان  $NFA$  مربوط به مثال قبل چیست؟  
با رشته ورودی 00011 وارد  $q_4$  می شویم و  $\delta^*(q_4, 1) = \emptyset$  باشد. به این وضعیت یک پیکربندی مرده  $L(\overline{M}) = \{00011\}^*$  گویند. بنابراین (*Dead Configuration*)

## فصل دوم

کامپیوترهای رقمی (دیجیتالی) کاملاً معین هستند . وضعیت آنها در هر زمان از روی ورودی و وضعیت اولیه قابل پیش بینی است .

- پس چرا به ماشین های نامعین نیاز داریم ؟  
الگوریتم های معین (مانند بازیها) مستلزم اتخاذ تصمیم هستند. غالباً بهترین حرکت مشخص نیست ، اما میتوان آنرا با یک جستجوی کامل همراه با بازگشت به عقب (*Back-Tracking*) پیدا کرد. وقتی چندین انتخاب داشته باشیم یکی را انتخاب کرده و آن را دنبال می کنیم. یک الگوریتم نامعین که بتواند بهترین انتخاب را انجام دهد ، قادر است بدون بازگشت به عقب مساله را حل نمایند.

## فصل دوم

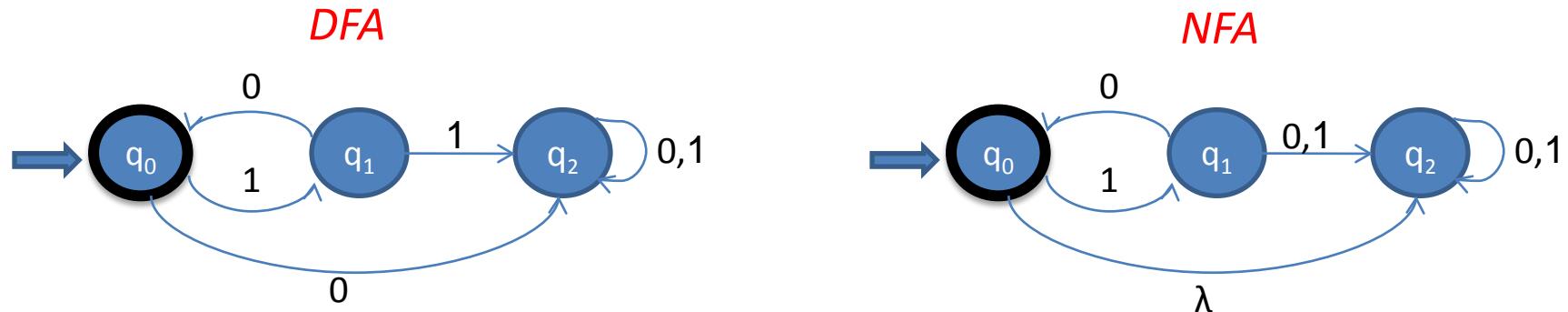
- ماشین های نامعین میتوانند بعنوان مدل برای الگوریتم های جستجو با بازگشت به عقب عمل کنند. (در مسائل عدم قطعیت *(Nondeterministic)*)
- عدم قطعیت مکانیزم موثری برای توصیف زبانهای پیچیده است. یک گرامر نیز دارای  $(S \rightarrow aSb \mid \lambda \mid bS)$  عدم قطعیت است.
- تفاوت عمدہای بین *NFA* و *DFA* وجود ندارد و بعضی نتایج را با *NFA* بهتر از *DFA* میتوان بدست آورد.

## فصل دوم

• هم ارزی  $NFA$  و  $DFA$  •

.  $L(M_1) = L(M_2)$  در صورتی مساویند که  $(M_1, M_2)$  دو آتماتون

آیا  $NFA$  و  $DFA$  زیر مساویند؟ •

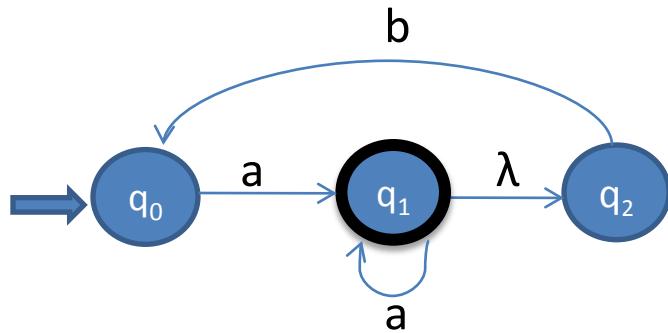


$$L(M) = \{(10)^n : n \geq 0\}$$

در هر دو آتماتا

## فصل دوم

•  $NFA$  زیر را به  $DFA$  تبدیل نمایید ؟



$NFA$  : وضعیت شروع  $q_0 \rightarrow DFA$  : وضعیت شروع  $q_0$

$$\delta(\{q_0\}, a) = \{q_1, q_2\}$$

حرکت غیرممکن و به معنای عدم قبول رشته است. بنابراین  $\delta(\{q_0\}, b) = \emptyset$  این وضعیت در  $DFA$  باید یک  $a$  غیرنهایی باشد.

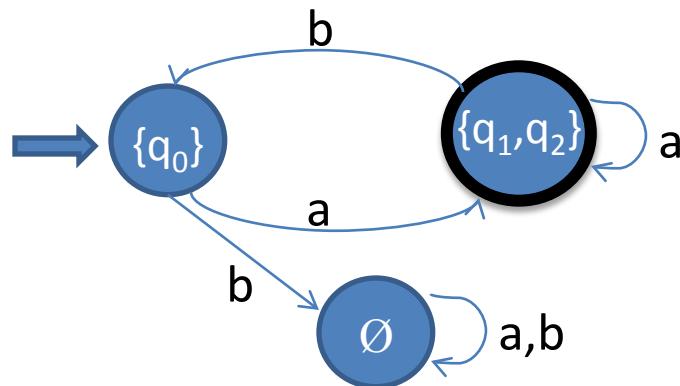
## فصل دوم

$\delta (\{q_1, q_2\}, a) = \{q_1, q_2\}$   $\longrightarrow$  چون با  $\lambda$  میتوان بین آنها حرکت کرد.  $\{q_1, q_2\}$

$\delta (\{q_1, q_2\}, b) = \{q_0\}$



هر وضعیتی که شامل  $q_1$  باشد نهایی است.



• قضیه :

اگر  $L_N$  زبان پذیرفته شده توسط آنگاه یک DFA  $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$  باشد، وجود دارد که  $L(M_N) = L(M_D)$  بطوریکه  $M_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$

## فصل دوم

• روال تبدیل  $NFA$  به  $DFA$  :

- (1) یک گراف  $G_D$  با رأس  $\{q_0\}$  ساخته و بعنوان رأس ابتدایی  $DFA$  در نظر بگیرید.
- (2) گامهای زیر را تکرار کنید تا هیچ یالی کم نباشد.
  - (2-1) هر یک از رئوس  $\{q_k, q_i, q_j, \dots, q_l\}$  از  $G_D$  را که یال خروجی برای  $a \in \Sigma$  ندارد، بگیرید.
  - (2-2) برای آن  $\delta^*(q_i, a), \delta^*(q_j, a), \dots, \delta^*(q_k, a)$  را محاسبه کنید.
  - (2-3) اجتماع تمام این  $\delta^*$  ها را پیدا کرده و مجموعه  $\{q_1, q_m, \dots, q_n\}$  را تشکیل دهید.
  - (2-4) یک راس بنام  $\{q_1, q_m, \dots, q_n\}$  برای  $G_D$  بسازید (اگر از قبل موجود نیست).
  - (2-5) به  $G_D$  یک یال از  $\{q_i, q_j, \dots, q_k\}$  به  $\{q_1, q_m, \dots, q_n\}$  اضافه نمایید و آنرا  $a$  نامگذاری کنید.

## فصل دوم

(2-6) هر یک از حالت‌های  $G_D$  که نام آن حاوی  $F$  می‌باشد را به عنوان یک وضعیت نهایی علامتگذاری کنید.

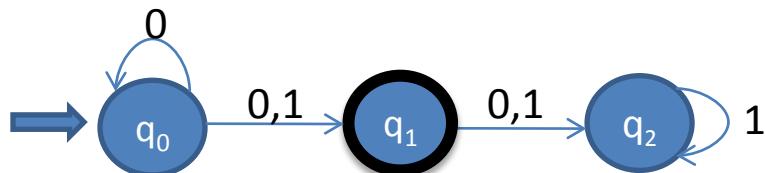
(2-7) اگر  $M_N$  ورودی  $\lambda$  داشته باشد، رأس  $\{q_0\}$  در  $G_D$  هم به عنوان وضعیت نهایی علامت گذاری می‌شود.

• نکته :

حلقه بالا متناهی است و هر بار که تکرار می‌گردد، یک یال به  $G_D$  اضافه می‌گردد.  $G_D$  حداقل دارای  $|\sum|^{Q_N}|^2$  یال است و یک مقدار متناهی است.

## فصل دوم

• مثال : با استفاده از روال،  $NFA$  ای زیر را به  $DFA$  تبدیل نمایید ؟



1)  $\delta_N(q_0, 0) = \{q_0, q_1\} \longrightarrow G_D \{q_0, q_1\}$  در  $\{q_0, q_1\}$

ایجاد یک یال با برچسب 0 بین رئوس  $\{q_0\}, \{q_0, q_1\}$  بین رئوس

2)  $\delta(q_0, 0) = \{q_0, q_1\}, \delta(q_0, 1) = \{q_1\}$

یک یال با برچسب 1 بین  $\{q_1\}, \{q_0\}$  وجود دارد.  $\longrightarrow$

## فصل دوم

$$\delta_N^*(q_0, 0) \cup \delta_N^*(q_1, 0) = \{q_0, q_1, q_2\} \quad \downarrow$$

یک وضعیت جدید بنام  $\{q_0, q_1, q_2\}$  با یال 0 ایجاد می‌گردد یعنی برای  $(a=0, i=0, j=0)$

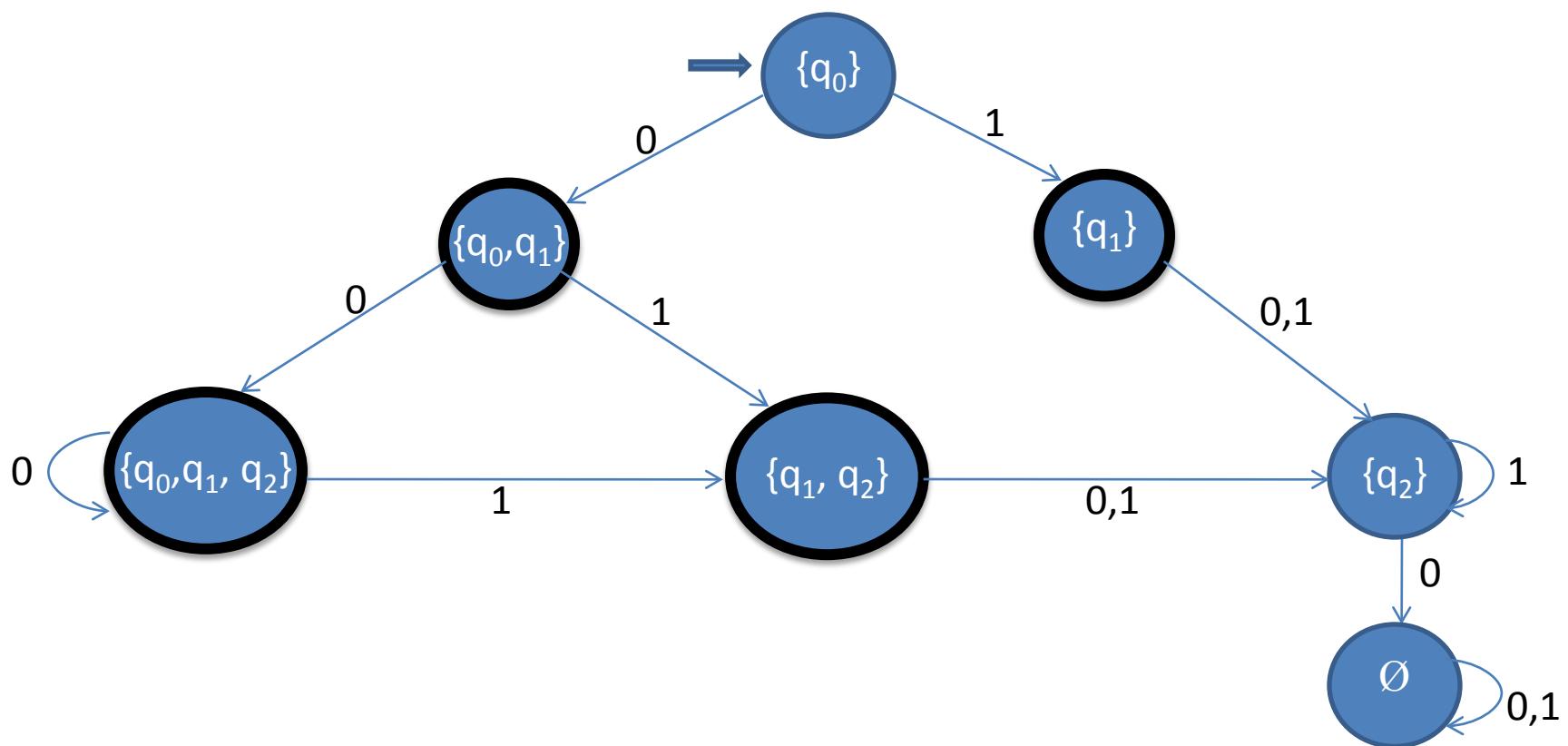
$$\delta_D^*(\{q_0, q_1\}, 0) = \{q_0, q_1, q_2\}$$

$$(\delta_D^*(\{q_0, q_1\}, 1) = \{q_1, q_2\})$$

همچنین برای  $a=1, i=0, j=1, k=2$  داریم :

$$\delta_N^*(q_0, 1) \cup \delta_N^*(q_1, 1) \cup \delta_N^*(q_2, 1) = \{q_1, q_2\} \quad \text{یک وضعیت جدید}$$

# فصل دوم



## فصل دوم

$$\delta^*_D (\{q_0, q_1, q_2\}, 0) = \{q_0, q_1, q_2\}$$

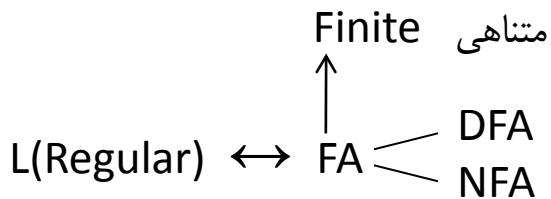
$$\begin{cases} \delta^*_D (\{q_1\}, 0) = \{q_2\} \\ \delta^*_D (\{q_1\}, 1) = \{q_2\} \end{cases} \quad \text{وضعیت جدید}$$

$$\begin{cases} \delta^*_D (\{q_1, q_2\}, 0) = \{q_2\} \\ \delta^*_D (\{q_1, q_2\}, 1) = \{q_2\} \end{cases}$$

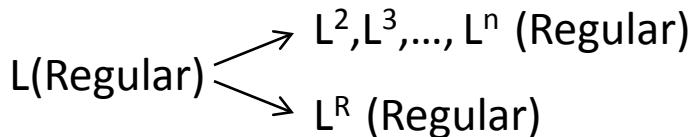
$$\begin{cases} \delta^*_D (\{q_2\}, 0) = \emptyset \\ \delta^*_D (\{q_2\}, 1) = \{q_2\} \end{cases} \quad \text{وضعیت جدید}$$

# فصل دوم

نکته ۱ : تمام زبان های متناهی منظم هستند .



نکته ۲ : اگر  $L$  منظم باشد ، آنگاه  $L^R$  هم منظم است .



## فصل دوم

### • کاهش تعداد وضعیت های FA :

یک DFA زبان منحصر بفردی را می پذیرد، اما عکس آن صحت ندارد و DFA های مختلفی می توانند با تعداد وضعیتهای متفاوت دارای یک زبان باشند. برای استفاده بهینه از حافظه، مطلوبست تا جایی که امکان دارد تعداد وضعیت ها را کاهش دهیم.

### ادغام پذیری وضعیتها :

دو وضعیت  $q$  و  $p$  از یک DFA در صورتی ادغام پذیر (*Indistinguishable*) میباشند که :

$$\forall w \in \Sigma^* \quad \left\{ \begin{array}{l} \delta^*(p, w) \in F \rightarrow \delta^*(q, w) \in F \\ \delta^*(p, w) \notin F \rightarrow \delta^*(q, w) \notin F \end{array} \right.$$

## فصل دوم

- اگر رشته ای مانند  $w \in \Sigma^*$  وجود داشته باشد، بطوریکه :  
     $\delta^*(p, w) \in F$  ،  $\delta^*(q, w) \notin F$  ، یا بر عکس ، آنگاه وضعیتهای  $q$  و  $p$  را ادغام ناپذیر به  
    واسطه رشته  $w$  می نامیم.
  - رابطه ادغام پذیری یک رابطه هم ارزی است (اگر  $q$  و  $p$  ادغام پذیر و  $r$  و  $q$  با هم  
    ادغام پذیر باشند، آنگاه  $r$  و  $p$  نیز ادغام پذیرند).
  - یکی از راههای کاهش وضعیتهای یک DFA یافتن وضعیتهای ادغام پذیر و ترکیب  
    آنهاست. حال به چند روش مختلف ادغام کردن وضعیتها می پردازیم.
- قضیه :** روال علامت گذاری به هر DFA به نام  $M(Q, \Sigma, \delta, q_0, F)$  اعمال شود  
پایان می یابد و تمامی وضعیتهای ادغام ناپذیر را مشخص می کند .

## فصل دوم

### روال علامت گذاری :

- ۱) تمامی وضعیت های غیر قابل دسترس را حذف نمایید. بدین صورت که :  
کلیه مسیرهای ساده گراف  $DFA$  را از وضعیت های شروع مشخص نمایید و  
هر وضعیتی که در یکی از این مسیرها نیست ، غیر قابل دسترسی است.
- ۲) کلیه زوج های وضعیت  $(p, q)$  را بررسی نمایید. اگر  $p \in F$  و  $q \notin F$  و یا  
برعکس باشد، آنها را بعنوان ادغام ناپذیر علامت بزنید.
- ۳) گام زیر را آنقدر تکرار کنید تا زوجی برای علامت گذاری باقی نماند.
  - برای همه زوج های  $(p, q)$  و همه  $a \in \sum$ ،  $\delta(p, a) = p_a$  و  $\delta(q, a) = q_a$  را  
محاسبه نمایید. اگر زوج  $(p_a, q_a)$  بعنوان ادغام ناپذیر علامت گذاری شده  
باشد، آنگاه  $(p, q)$  را بعنوان ادغام ناپذیر علامت بزنید.

## فصل دوم

روال کاهش :

- یک DFA مانند  $M(Q, \Sigma, \delta, q_0, F)$  داده شده است. یک DFA کاهش یافته مانند  $\hat{M}(\hat{Q}, \hat{\Sigma}, \hat{\delta}, \hat{q}_0, \hat{F})$  به شرح زیر می سازیم :

۱) با استفاده از روال علامت گذاری تمامی زوج های وضعیت غیر قابل ادغام را پیدا کنید و سپس تمامی وضعیت های قابل ادغام را پیدا کنید.

۲) به ازای هر مجموعه وضعیت قابل ادغام مانند  $\{q_i, q_j, \dots, q_k\}$  یک وضعیت با برچسب  $k, \dots, j, i$  برای  $\hat{M}$  ایجاد نمایید.

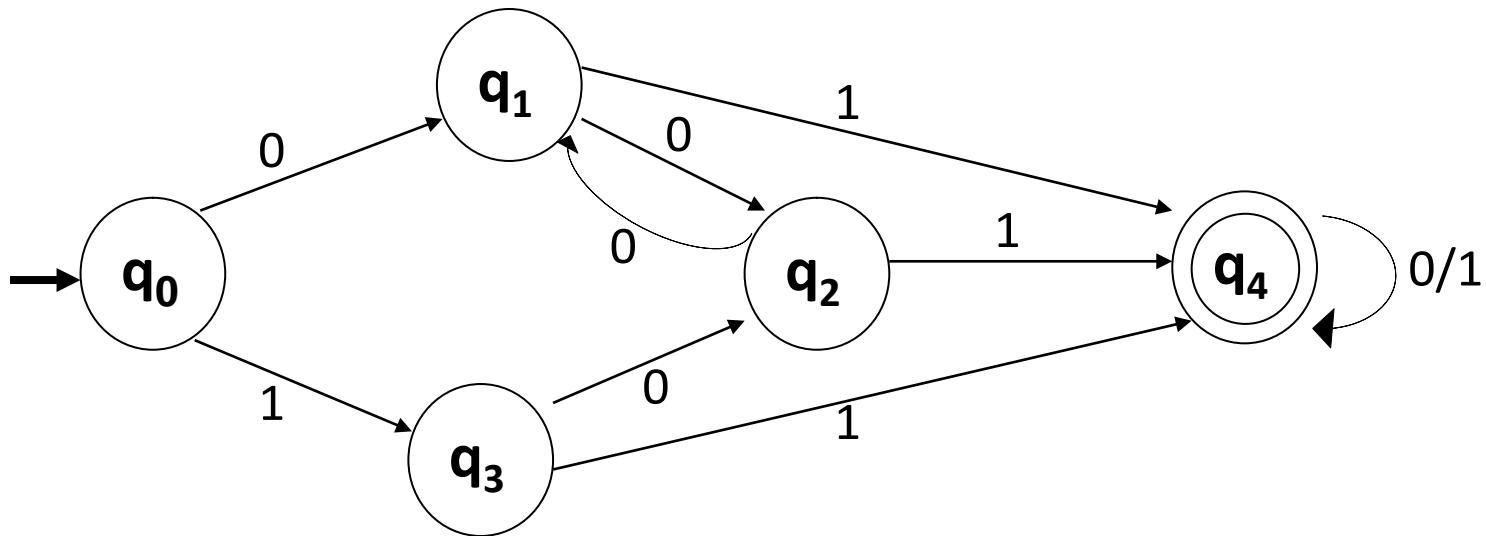
۳) به ازای هر قانون انتقال از  $M$  به شکل  $\delta(q_r, a) = q_p$ ، مجموعه ای را که  $q_r$  و  $q_p$  به آن تعلق دارند، پیدا کنید. اگر  $\{q_i, q_j, \dots, q_k\}$  و  $q_p \in \{q_r, q_m, \dots, q_n\}$  باشد، قانون  $\delta(ij\dots k, a) = lm\dots n$  را به  $\hat{\delta}(ij\dots k, a) = lm\dots n$  اضافه کنید.

۴) وضعیت ابتدایی  $\hat{q}_0$  وضعیتی از  $\hat{M}$  است که برچسب آن شامل ۰ است.

۵)  $F$  مجموعه تمامی وضعیتهايی است که بر چسب آن شامل ۱ است، بطوریکه  $q_i \in F$  است.

## فصل دوم

• مثال : آتماتای زیر را در نظر بگیرید.

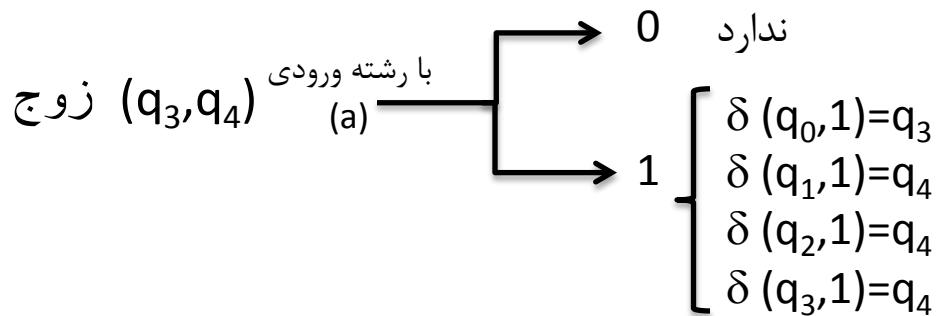


روال علامت گذاری :

- (۱) وضعیت غیر قابل دسترسی وجود ندارد.
- (۲) زوج های ادغام ناپذیر :  $(q_0, q_4)$  ,  $(q_1, q_4)$  ,  $(q_2, q_4)$  ,  $(q_3, q_4)$   
(در این حالت خاص، چون تنها حالت نهایی  $q_4$  می باشد، کلیه رشته هایی که توسط  $q_4$  پذیرفته شده است، توسط هیچکدام دیگر پذیرفته نیست).

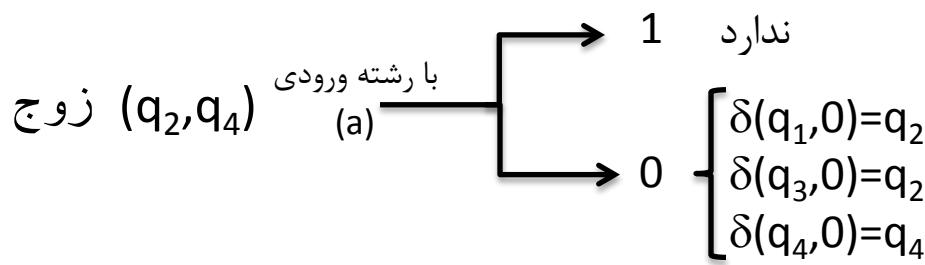
## فصل دوم

(۳)



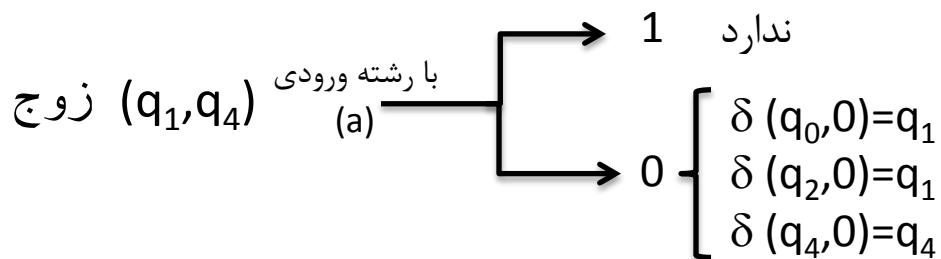
چون  $(q_3, q_4)$  ادغام ناپذیر است

زوج های ادغام ناپذیر  
 $(q_0, q_1), (q_0, q_2), (q_0, q_3)$



چون  $(q_2, q_4)$  ادغام ناپذیر است

زوج های ادغام ناپذیر  
 $(q_1, q_4), (q_3, q_4)$



چون  $(q_1, q_4)$  ادغام ناپذیر است

زوج های ادغام ناپذیر  
 $(q_0, q_4), (q_2, q_4)$

# فصل دوم

روال کاهش :

(گام ۱ روال کاهش)

$$\rightarrow \begin{cases} \text{مجموعه ادغام ناپذیر} = \{(q_0, q_1), (q_0, q_2), (q_0, q_3), (q_0, q_4), (q_1, q_4), (q_2, q_4), (q_3, q_4)\} \\ \text{مجموعه ادغام پذیر} = \{(q_1, q_2), (q_1, q_3), (q_2, q_3)\} \end{cases}$$

$$\rightarrow \text{وضعیتهای ادغام پذیر} = \{q_1, q_2, q_3\} \quad (\text{گام ۲ روال کاهش})$$

$$\rightarrow \begin{matrix} \{q_0\}, \{q_1, q_2, q_3\}, \{q_4\} \\ \text{برچسبها} \quad 0, \quad 123, \quad 4 \end{matrix}$$

$$\left[ \begin{array}{l} \delta(q_0, 0) = q_1 \\ \delta(q_1, 0) = q_2 \\ \delta(q_2, 0) = q_1 \\ \delta(q_3, 0) = q_2 \\ \delta(q_4, 0) = q_4 \end{array} \right] \quad \left[ \begin{array}{l} \delta(q_0, 1) = q_3 \\ \delta(q_1, 1) = q_4 \\ \delta(q_2, 1) = q_4 \\ \delta(q_3, 1) = q_4 \\ \delta(q_4, 1) = q_4 \end{array} \right]$$

```

graph LR
    S(( )) --> 0((0))
    0 -- "0/1" --> 123((123))
    123 -- "1" --> 4(((4)))
    4 -- "0/1" --> E(( ))
  
```

(گام ۳ روال کاهش)

## فصل دوم

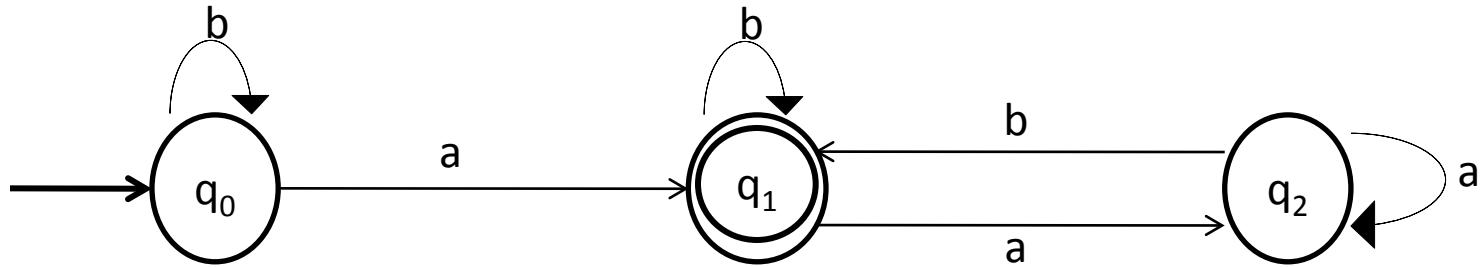
• قضیه :

اگر یک DFA به نام  $M$  داشته باشیم ، روال کاهش یک DFA دیگر به نام  $\hat{M}$  می دهد  
بطوریکه  $L(M) = L(\hat{M})$  علاوه بر آن  $\hat{M}$  مینیمال است بدین معنی که هیچ DFA دیگری  
با تعداد کمتری وضعيت که  $L(M)$  را بپذيرد ، وجود ندارد.

## فصل دوم

### • تمرینات مهم فصل دوم:

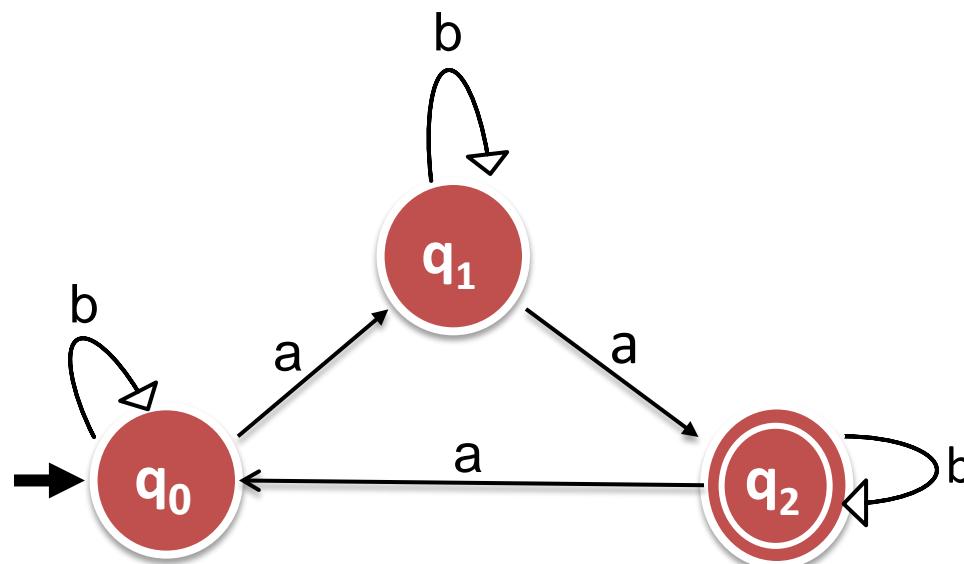
۱) زبانی را که توسط آتاماتای زیر پذیرفته می شود ، بصورت مجموعه ای بنویسید ؟



$$L = \{w : w = b^n a \mid u b : u \in \Sigma^+, n_a(u) \neq 0, n \geq 0\}$$

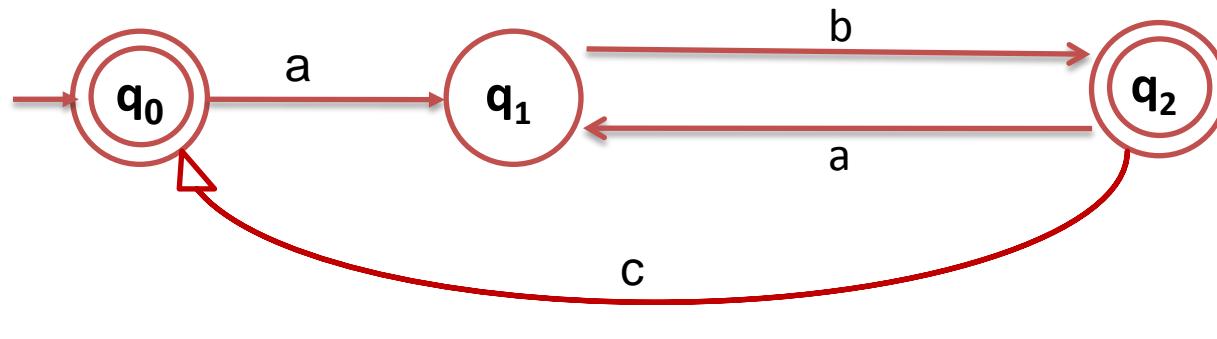
## فصل دوم

۲) یک DFA برای زبان  $L = \{w : n_a(w) \bmod 3 > 1\}$  طراحی نماید.

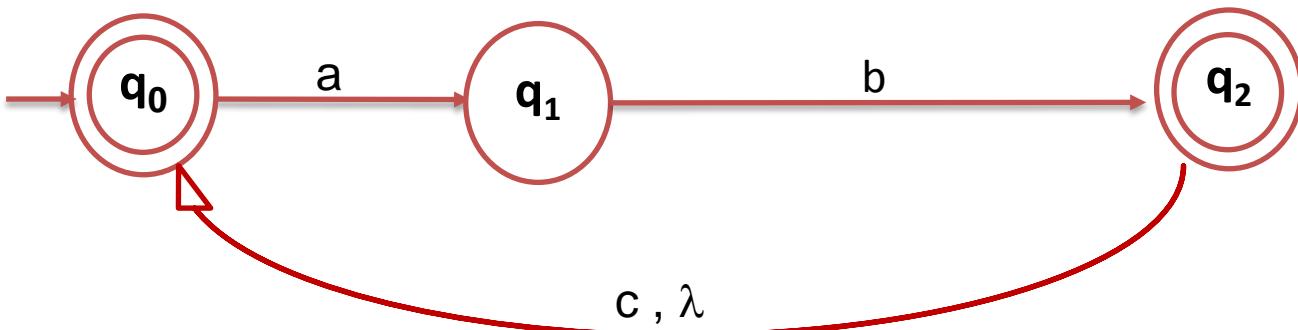


## فصل دوم

۳) یک NFA که زبان  $\{ab, abc\}^*$  را بپذیرید، طراحی کنید؟

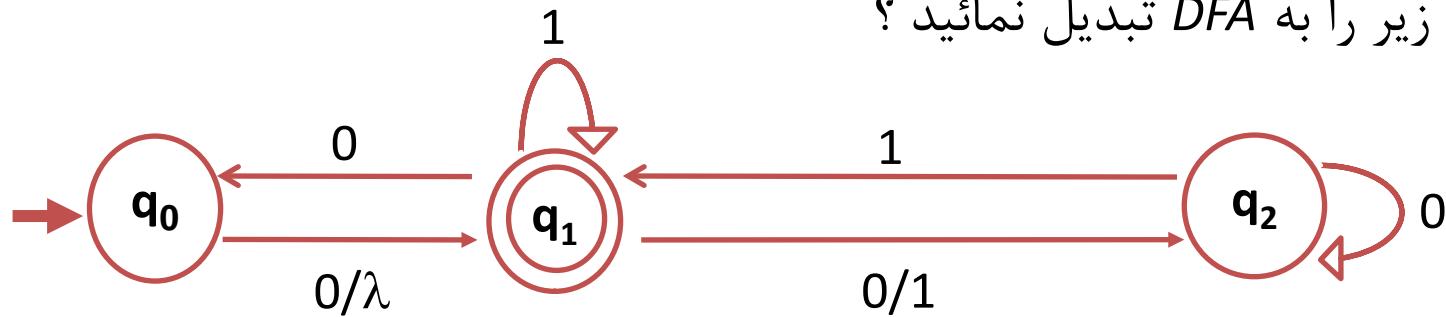


یا



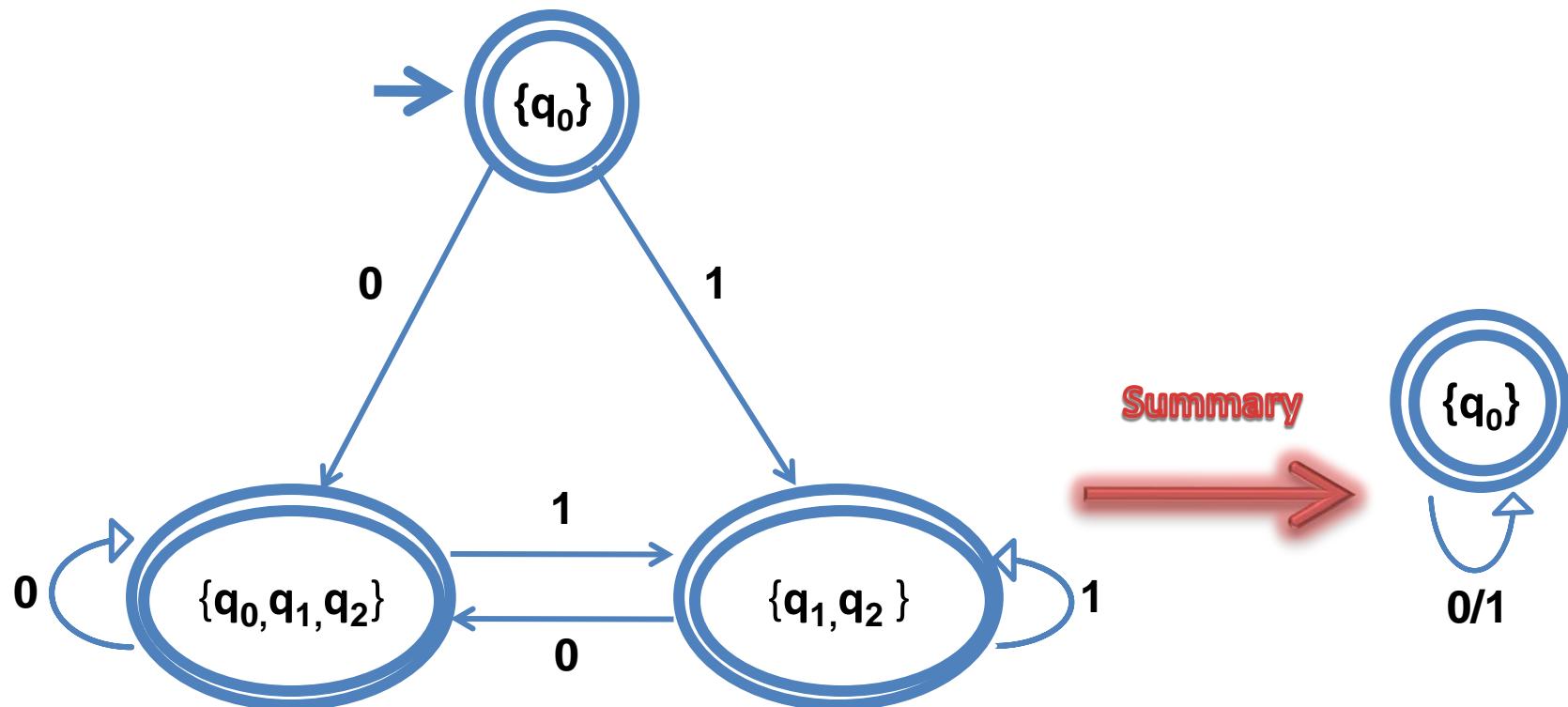
# فصل دوم

زیر را به DFA تبدیل نمایید ؟



$$\begin{aligned}\delta(\{q_0\}, 0) &= \{q_0, q_1, q_2\} \\ \delta(\{q_0\}, 1) &= \{q_1, q_2\} \\ \delta(\{q_1, q_2\}, 0) &= \{q_0, q_1, q_2\} \\ \delta(\{q_1, q_2\}, 1) &= \{q_1, q_2\} \\ \delta(\{q_0, q_1, q_2\}, 0) &= \{q_0, q_1, q_2\} \\ \delta(\{q_0, q_1, q_2\}, 1) &= \{q_1, q_2\}\end{aligned}$$

## فصل دوم



- چون  $NFA$ ،  $\lambda$  را هم می پذیرد پس  $q_0$  وضعیت نهایی خواهد بود.
- $DFA$  بالا همه رشته ها را می پذیرد. بنابراین شکل فوق بطور خلاصه در خواهد آمد.

# فصل سوم

زبانهای منظم و گرامرهاي منظم

# فصل سوم

- یک زبان در صورتی منظم خوانده می شود که یک پذیرنده متناهی (FA) برای آن وجود داشته باشد. بنابراین هر زبان منظم می تواند توسط یک DFA یا NFA توصیف شود.

## • عبارات منظم :

- یکی از راههای توصیف زبانهای منظم ، عبارات منظم میباشد.
- عبارت منظم ، ترکیبی از رشته هایی از حروف یک الفبای  $\Sigma$  و پراتنز و عملگرهای . ، \* ، + می باشد.
- یک زبان مانند  $\{a,b,c\}^* = \sum$  با استفاده از + به عنوان  $\Sigma$  ، از \* به عنوان بستار ستاره ای (Star Closure) ، واژ . برای اتصال استفاده می کنیم.
- به عنوان مثال ، در زبان  $\{bc\} \cup \{a\}^*$  که همان زبان  $\{...,abc,bca,...\}$  است ، دارای بستار ستاره ای  $(a+b.c)^*$  می باشد.

# فصل سوم

## • تعریف رسمی عبارات منظم :

اگر  $\Sigma$  الفبا باشد ، خواهیم داشت :

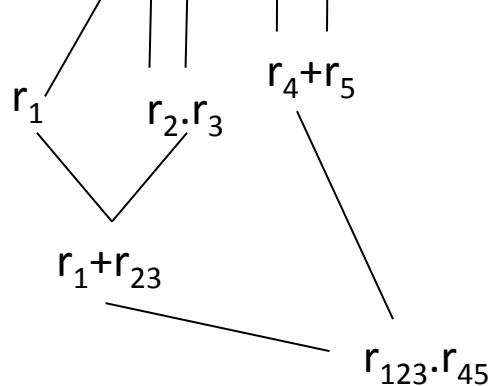
(۱)  $a \in \Sigma, \emptyset, \lambda$  عبارات منظم ابتدایی هستند.

(۲) اگر  $r_1, r_2$  عبارت منظم باشند ،  $r_1^*, r_1 \cdot r_2, r_1 + r_2$  عبارت منظم هستند.

(۳) رشته ای عبارت منظم است، اگر و فقط اگر بتوان آنرا از عبارات منظم ابتدایی با اعمال دفعات متناهی قانون ۲ تولید کرد.

عبارت منظم  $\rightarrow (a+b.c)^* \cdot (c+\emptyset)$  رشته

• مثال :  $\Sigma = \{a,b,c\}$



عبارت منظم نیست  $\rightarrow (a+b+)$

# فصل سوم

- زبانهای مرتبط با عبارات منظم :
- اگر  $r$  یک عبارت منظم باشد ،  $L(r)$  را زبان مرتبط با  $r$  می گوئیم.
- زبان  $L(r)$  که با عبارت منظم  $r$  نشان داده می شود ، با قواعد زیر تعریف می گردد :
  - (۱)  $\emptyset$  یک عبارت منظم است (مجموعه تهی).
  - (۲)  $\lambda$  یک عبارت منظم است ( $\{\lambda\}$ ).
  - (۳) برای هر  $a$  ،  $a \in \Sigma$  یک عبارت منظم است ( $\{a\}$ ).

$$4-1) L(R_1 + R_2) = L(R_1) \cup L(R_2)$$

$$4-2) L(R_1 \cdot R_2) = L(R_1) \cdot L(R_2)$$

$$4-3) L((R_1)) = L(R_1)$$

$$4-4) L(R_1^*) = L((R_1))^*$$

(۴) اگر  $r_1, r_2$  عبارت منظم باشند :

# فصل سوم

مثال : زبان  $L(a^*(a+b))$  را با نماد مجموعه ای نشان دهید؟

$$L(a^*(a+b)) = L(a^*) \cdot L(a+b) = L((a))^* \cdot L(a) \cup L(b)$$

$$\{\lambda, a, aa, aaa, \dots\} \cdot \{a, b\} = \{a, aa, aaa, \dots, b, ab, aab, \dots\}$$

• تقدم عملگرهای عبارات منظم :



مثال : عبارت  $r=(a+b)^*(a+bb)$  منظم است و نمایانگر زبان  $\sum=\{a,b\}$  است

$$L(r)=\{a, bb, aa, abb, ba, bbb, \dots\}$$

$(a+b)^*$  → نمایانگر هر رشته از a ها و b ها است.  
 $(a+bb)$  → نمایانگر رشته a یا bb است.

$L(r)$

مجموعه تمامی زبانهای مرتبط به الفبای  $\sum$  است و به a و یا bb ختم می شود.

## فصل سوم

**مثال :** عبارت  $r = (aa)^*(bb)^*b$  ، مجموعه تمامی رشته هایی با تعداد زوج  $a$  و بدنبال آن تعداد فرد  $b$  است. به عبارت دیگر :

$$L(r) = \{a^{2n} b^{2m+1} : n \geq 0, m \geq 0\}$$

**مثال :** برای  $\Sigma = \{1,0\}$  ، یک عبارت منظم  $r$  را بیابید که:

$$L(r) = \{w \in \Sigma^* \text{ دارای یک جفت صفر باشد :}\}$$

✓ هر رشته ای در  $L(r)$  باید حاوی ۰۰ در جایی باشد، ولی اینکه قبل و یا بعد از آن چه باشد، فرقی نمی کند. لذا خواهیم داشت :

$$(0+1)^* 00 (0+1)^*$$

# فصل سوم

مثال : یک عبارت منظم برای زبان زیر بیابید :

$L = \{w \in \{0,1\}^* \mid w \text{ دارای یک زوج صفر متوالی نباشد}\}$

✓ در این زبان، بعد از هر 0 باید 1 باید. چنین رشته‌ای میتواند میان تعداد دلخواه 1..101..1 باشد ( $1^*011^*$ ), اما این کامل نیست و خواهیم داشت :

$$r = (01)^* + (10)^* \rightarrow$$

$$r = (1+01)^*(0+\lambda) \rightarrow$$

$$r = (1+01)^*(0+1^*)$$

✓ تعداد نا محدودی عبارت منظم برای هر زبان وجود دارد.  
✓ دو عبارت منظم در صورتی مساویند که یک زبان را نمایش دهند.

• روابط مهم در عبارات منظم :

$$(r^*)^* = r^*$$

$$r_1^* (r_1 + r_2)^* = (r_1 + r_2)^*$$

$$(r_1 + r_2)^* = (r_1^* r_2^*)^*$$

# فصل سوم

## • ارتباط بین عبارات منظم و زبانهای منظم :

- ✓ هر زبان منظم ، یک عبارت منظم دارد و برای هر عبارت منظم ، یک زبان منظم وجود دارد.
- ✓ عبارت منظم ، زبانهای منظم را نمایش می دهد.
- ✓ یک زبان در صورتی منظم است که توسط یک DFA پذیرفته شود. باستی نشان دهیم اگر یک عبارت منظم  $r$  داشته باشیم ، میتوانیم یک NFA بسازیم که  $L(r)$  را بپذیرد و ساختن آن بر تعریف بازگشتی  $L(r)$  استوار باشد.

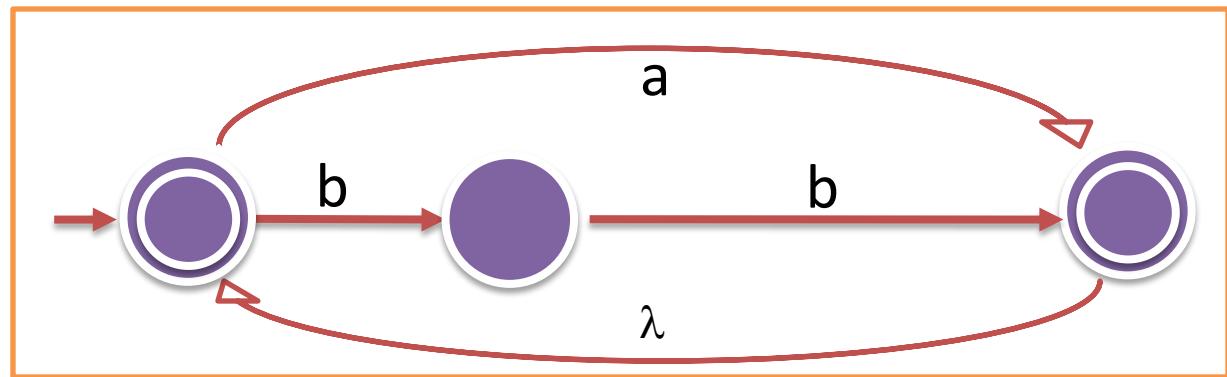
**قضیه :** اگر  $r$  یک عبارت منظم باشد آنگاه یک DFA وجود دارد که  $L(r)$  را می پذیرد و در نتیجه  $L(r)$  یک زبان منظم است.

# فصل سوم

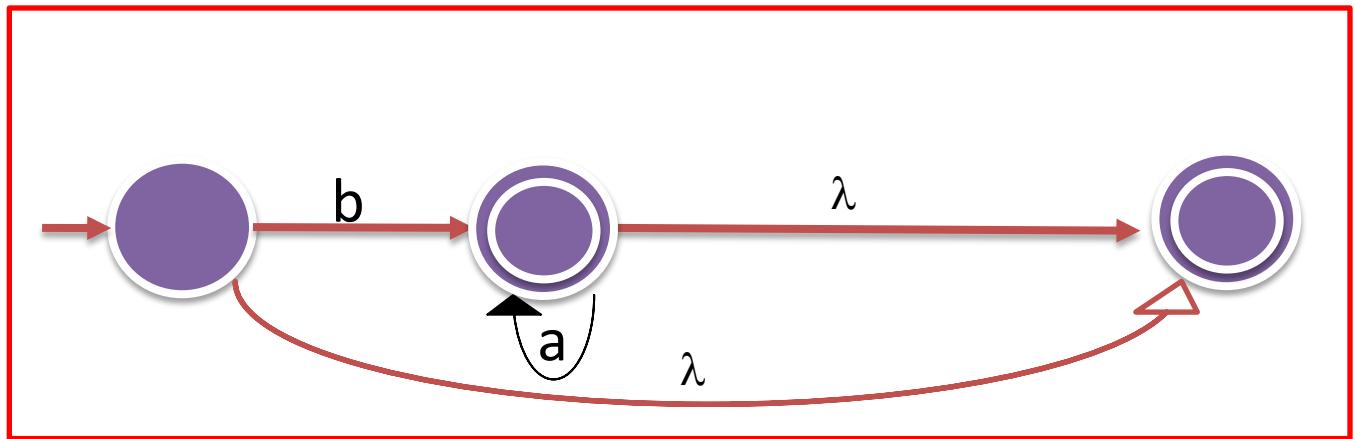
مثال : یک NFA بیابید که  $L(r) = (a + bb)^*(ba^* + \lambda)$  را در جایی که  $L(r)$  باشد ، را بپذیرد ؟

بایستی آتماتای مربوط به  $L(a+bb)^*$  و مربوط به  $L(ba^* + \lambda)$  را بسازیم.

M1: $L(a+bb)^*$

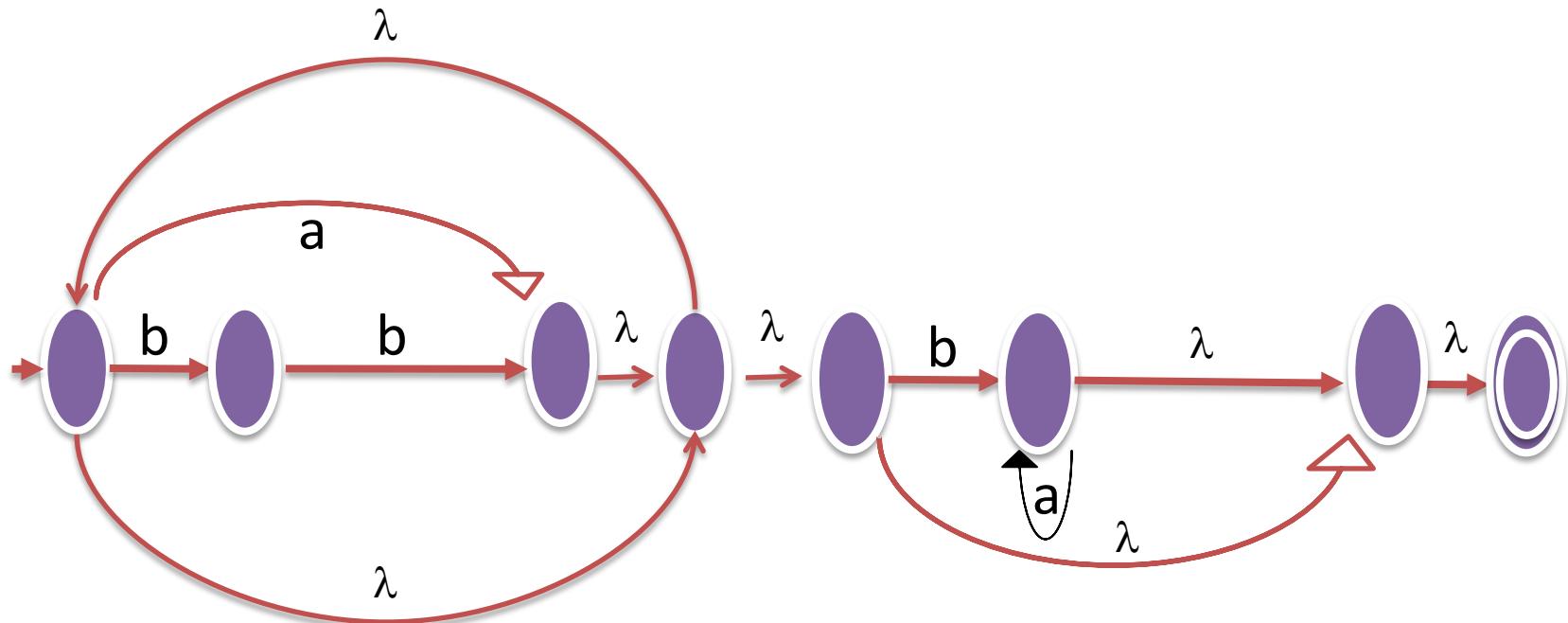


M2: $L(ba^* + \lambda)$



# فصل سوم

و سپس آن دو را ترکیب نمائیم.



$L((a+bb)^*(ba^*+\lambda))$  آتاماتای پذیرنده زبان

# فصل سوم

## • عبارات منظم برای زبان های منظم :

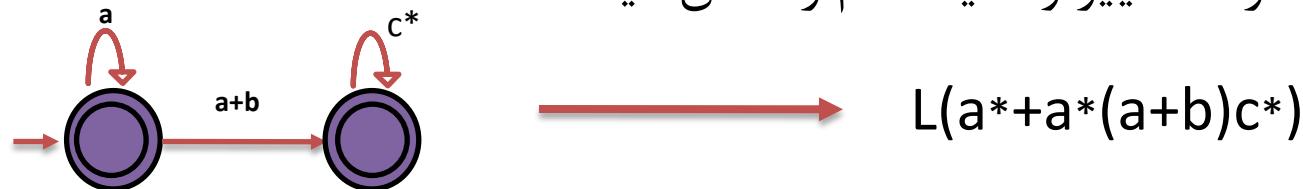
✓ برای هر زبان منظم ، یک عبارت منظم معادل وجود دارد. چون به ازای هر زبان منظم یک NFA وجود دارد که آن را می پذیرد ، بنابراین باید یک عبارت منظم بیابیم که قادر به تولید تمامی یال های کلیه راهها از  $q_0$  به یک وضعیت نهایی باشد. (مشکل چرخه ها در گراف)

## • گراف تغییر وضعیت عام (Generalized Transition Graph) :

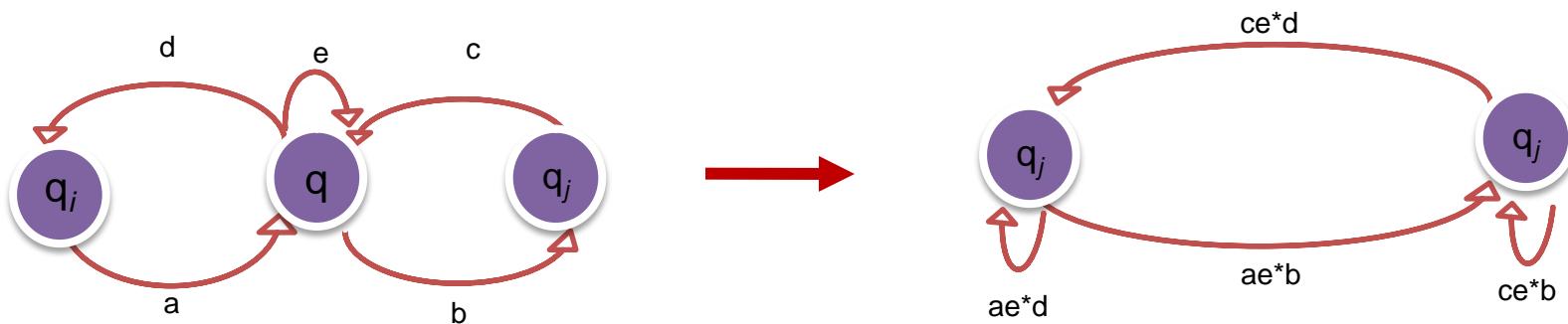
✓ گرافی است که یالهای آن با عبارات منظم نامگذاری می شود. برچسب هر راه از وضعیت اولیه به وضعیت نهایی در یک گراف تغییر وضعیت عام از اتصال چند عبارت منظم بدست می آید و در نتیجه خود یک عبارت منظم می شود. رشتہ هایی که بواسطه این عبارات منظم مشخص می شوند ، زیر مجموعه ای از زبانهای پذیرفته شده توسط گراف تغییر وضعیت عام است.

# فصل سوم

**مثال :** شکل زیر یک گراف تغییر وضعیت عام را نشان میدهد.



- ✓ می توان برچسب  $a$  را به  $a^*$  تبدیل کرد و تفسیرهای متفاوتی را ارائه داد.
- ✓ گراف تغییر وضعیتی به شکل زیر را در نظر گرفته که وضعیت  $q$  نه وضعیت اولیه است نه وضعیت نهایی.
- ✓ میخواهیم گراف معادلی با یک وضعیت کمتر که با حذف  $q$  بدست می آید ، پیدا کنیم.



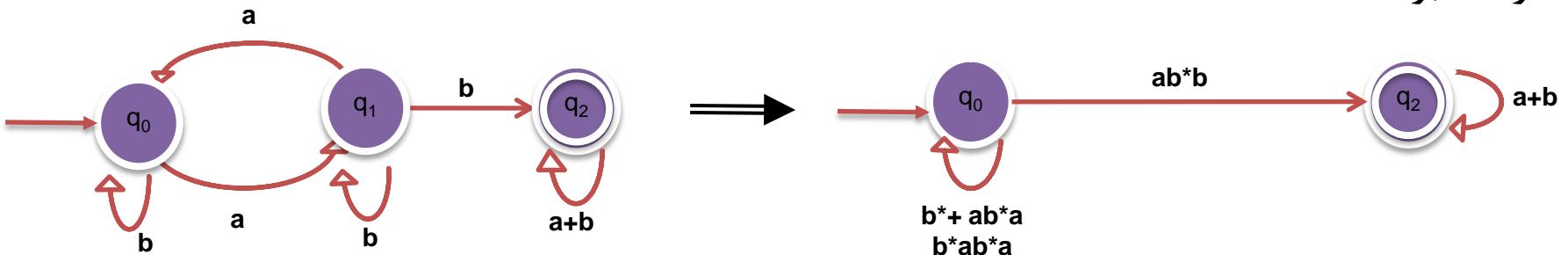
# فصل سوم

**قضیه :** اگر  $L$  یک زبان منظم باشد ، آنگاه یک عبارت منظم  $r$  وجود دارد ، بطوریکه

$$L = L(r)$$

**مثال :**  $NFA$  شکل زیر را در نظر بگیرید. گراف تغییر وضعیت عام پس از حذف  $q_1$

خواهد بود :



✓ با شناسایی  $r_4 = a+b$  ،  $r_3 = \emptyset$  ،  $r_2 = ab^*b$  ،  $r_1 = b+ab^*a$  عبارت منظم بدست می آید .

# فصل سوم

## • عبارت منظم برای توصیف الگوهای ساده :

- ✓ میتوان اجزا زبانهای برنامه سازی مانند متغیرها و اعداد صحیح و اعشاری را با پذیرنده های متناهی ارتباط داد و از عبارات منظم برای توصیف این اجزا استفاده کرد.
- ✓ بعنوان مثال : مجموعه کلیه اعداد صحیح در زبان پاسکال را میتوان با عبارت منظم  $Sdd^*$ ، که در آن علامت  $S$  (با مقادیر  $\{\lambda, -, +\}$ )، و  $d$  یک رقم از میان ارقام ۰ الی ۹ می باشد.
- ✓ الگو (*Pattern*) بمعنای مجموعه ای از عناصر است که دارای صفات مشترک باشند.  
مانند الگوی اعداد صحیح در پاسکال.
- ✓ تطبیق الگوها (*Pattern Matching*) به معنای مقدار دهی به عناصر است.

# فصل سوم

**مثال :** یکی از کاربردهای تطبیق الگو، ویرایش متون است. کلیه ویرایشگرها امکان جستجوی یک فایل برای یافتن یک رشته از حروف را فراهم می‌کند. بعنوان مثال، در سیستم عامل Unix دستور  $/aba^*c$ ، بعنوان دستور جستجوی فایل برای یافتن اولین مورد از رشته  $ab$  و بدنیال آن هر تعداد  $a$  و بدنیال آن یک  $c$  تفسیر می‌شود.

■ در کاربردهای دیگری که الگوهای پیچیده تر و نامحدود هستند، الگوهای از قبل معین نبوده و در زمان اجرا ایجاد می‌شوند.

اگر الگوها بواسطه عبارات منظم مشخص شوند، آنگاه برنامه تطبیق الگو می‌تواند این عبارت را به NFA تبدیل کند و سپس می‌توان طبق الگوریتم‌های بیان شده، DFA را به NFA تبدیل و کاهش داد. از یک DFA در قالب یک جدول تغییر وضعیت می‌توان یک الگوریتم تطبیق الگو ساخت.

# فصل سوم

## • گرامرهای منظم :

✓ گرامرها یکی از راههای توصیف زبانها می باشند.

✓ گرامر خطی راست و خطی چپ (*Right & Left Linear Grammar*)

✓ اگر در گرامر  $(RLG)$  همگی قواعد از نوع  $\begin{cases} A \rightarrow XB \\ A \rightarrow X \end{cases}$  باشند، گرامر خطی راست  $G(V, T, S, P)$  می نامند.

و اگر همگی قواعد از نوع  $\begin{cases} A \rightarrow BX \\ A \rightarrow X \end{cases}$ ، گرامر خطی چپ (*LLG*) می نامند.

$$A, B \in V, X \in T^*$$

✓ گرامر منظم ، گرامری است که یا خطی راست و یا خطی چپ باشد. در گرامر منظم، حداکثر یک متغیر ظاهر می شود و باید همواره آخرین نشانه سمت راست یا چپ، همان متغیر باشد.

# فصل سوم

$G_1 = (\{S, S_1, S_2\}, \{a, b\}, S, P_1)$

$$\begin{aligned}P_1 : \quad S &\rightarrow S_1ab \\&S_1 \rightarrow S_1ab | S_2 \\&S_2 \rightarrow a\end{aligned}$$

: مثال

✓ گرامر  $G_1$  یک گرامر خطی چپ و بنابراین یک گرامر منظم با زبان  $L(aab(ab)^*)$  می باشد.

$G_2 = (\{S, A, B\}, \{a, b\}, S, P_2)$

$$\begin{aligned}P_2 : \quad S &\rightarrow A \\&A \rightarrow aB \mid \lambda \\&B \rightarrow Ab\end{aligned}$$

: مثال

✓ گرامر  $G_2$  منظم نیست، زیرا نه خطی چپ و نه خطی راست است، ولی یک گرامر خطی است.

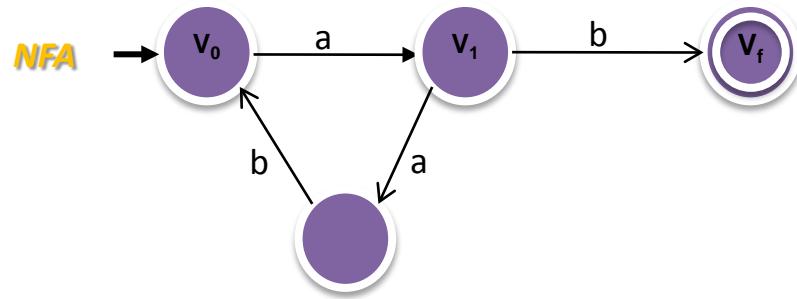
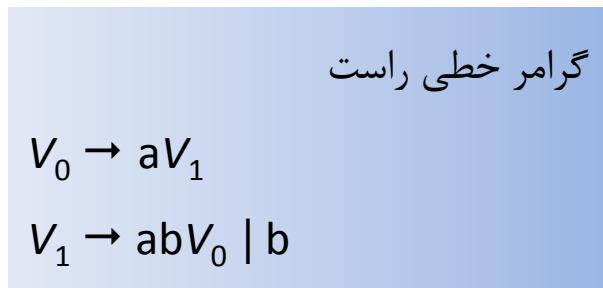
✓ گرامر خطی (*Linear Grammar*) گرامری است که در آن حداقل یک متغیر می تواند در سمت راست یک قانون یافت شود. بنابراین هر گرامر خطی ، منظم نیست، ولی هر گرامر منظم خطی است.

# فصل سوم

**نتیجه :** برای هر زبان منظم ، یک گرامر منظم وجود دارد.

**قضیه :** اگر  $G = (V, T, S, P)$  یک گرامر خطی راست باشد ، آنگاه  $L(G)$  یک زبان منظم است.

**مثال :** یک آتماتای متناهی بسازید که زبانی را که توسط گرامر زیر تولید میشود ، بپذیرد؟



$L((aab)^*ab)$  : زبان منظم

• **گرامر خطی راست برای زبانهای منظم :**

✓ از DFA استفاده کرده ، وضعيتهای گرامر و علائمی که باعث تغيير وضعیت می شوند ، ترمینال های گرامر می شوند.

# فصل سوم

**قضیه:** اگر  $L$  یک زبان منظم بر روی الفبا  $\Sigma$  باشد، آنگاه یک گرامر خطی راست، مانند  $G = (V, \Sigma, S, P)$  وجود دارد، بطوریکه  $L = L(G)$ .

**اثبات:** فرض کنید  $DFA$ ، ماشین  $M = (Q, \Sigma, S, q_0, F)$  پذیرنده  $L$  باشد. فرض می کنیم :

$$Q = \{q_0, q_1, \dots, q_n\}, \quad \Sigma = \{a_1, a_2, \dots, a_m\}$$

۱) گرامر خطی راست  $G(V, \Sigma, S, P)$  را با  $S = \{q_0\}$ ،  $V = \{q_0, q_1, \dots, q_n\}$  بسازید.

۲) برای هر تغییر وضعیتی مانند  $M$  از  $\delta(q_i, a_j) = q_k$ ، قانون زیر را به  $P$  اضافه کنید.

$$q_i \rightarrow a_j q_k$$

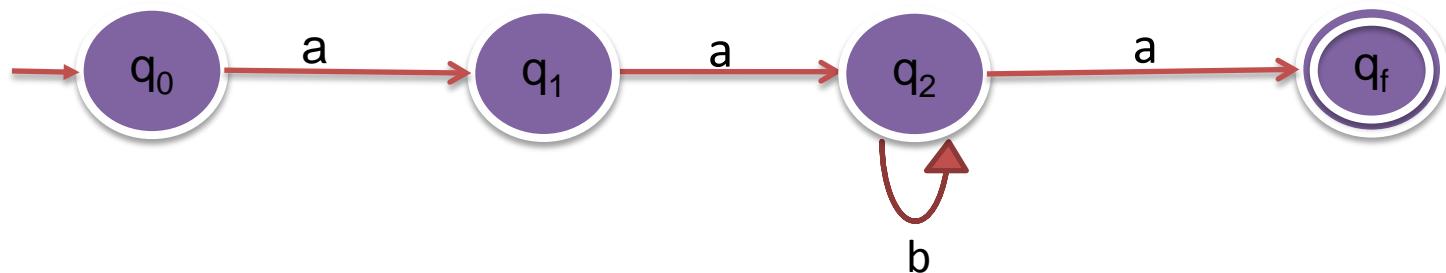
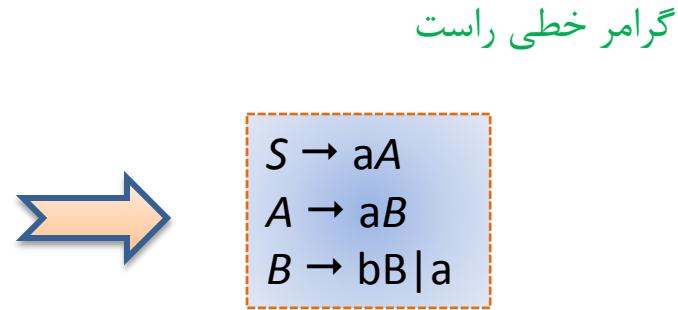
۳) بعلاوه اگر  $q_k \in F$  باشد، قانون زیر را نیز به  $P$  اضافه کنید.

$$q_k \rightarrow \lambda$$

# فصل سوم

• مثال : برای  $L(aab^*a)$ ، یک گرامر خطی راست بسازید؟

$\delta(q_0, a) = [q_1]$	$q_0 \rightarrow aq_1$
$\delta(q_1, a) = [q_2]$	$q_1 \rightarrow aq_2$
$\delta(q_2, b) = [q_2]$	$q_2 \rightarrow bq_2$
$\delta(q_2, a) = [q_f]$	$q_2 \rightarrow aq_f$
$q_f \in F$	$q_f \rightarrow \lambda$



# فصل سوم

- **قضیه :** زبان  $L$  منظم است، اگر و فقط اگر یک گرامر خطی چپ  $G$  باشد بطوریکه  $L=L(G)$ .
- **قضیه :** زبان  $L$  منظم است، اگر و فقط اگر گرامر منظمی مانند  $G$  داشته باشیم بطوریکه  $L=L(G)$ .
- برای توصیف زبان های منظم از راههای مختلف که عبارتنداز:  $DFA$  ها،  $NFA$  ها، عبارات منظم و گرامرهای منظم، استفاده می کنیم. این روش ها دارای ارتباط زیر هستند :

Regular  
Grammars



DFA  
NFA



Regular  
Expressions

# فصل سوم

## • تمرینات مهم فصل سوم:

(۱) برای مجموعه های زیر یک عبارت منظم بیابید؟

- $\{a^n b^m : n \geq 4, m \leq 3\} \rightarrow aaaaa^* (\lambda + b + bb + bbb)$
- $\{a^n b^m : (n + m) \text{ زوج}\} \rightarrow (aa)^* (ab + \lambda) (bb)^*$
- $\{a^n b^m : n \geq 1, m \geq 1, nm \geq 3\} \rightarrow a^+ (bbb)^+ + (aa)^+ (bb)^+ + (aaa)^+ b^+ = aa^* bbbb^* + aa(aa)^* bb(bb)^* + aaaa^* bb^*$

## فصل سوم

۲) برای زبان های زیر روی الفبای  $\Sigma = \{a, b, c\}$  عبارات منظم بنویسید ؟

الف) تمامی رشته هایی که حداکثر سه **a** دارند:

$$(b+c)^*(a+\lambda)(b+c)^*(a+\lambda)(b+c)^*(a+\lambda)(b+c)^*$$

ب ) رشته هایی که حداقل از هر یک از حروف الفبا یکی را داشته باشد:

If  $r=(a+b+c)^*$   $\rightarrow$  rarbrcr + rarcrcr + rbrarcr + rbrccrar + rcrarbr + rcrbrar

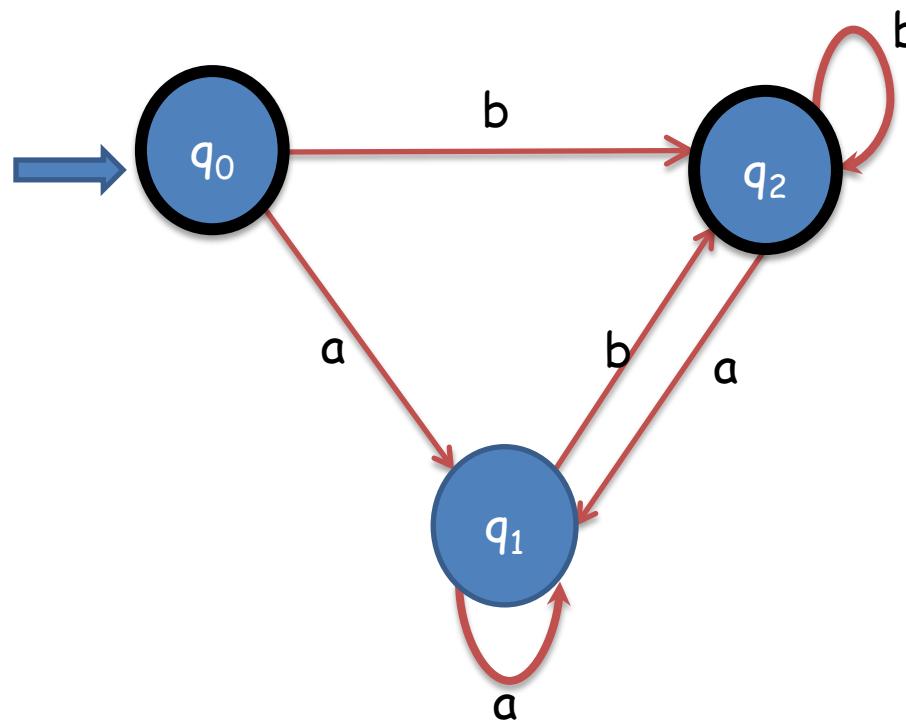
ج) رشته هایی که طول تمام دنباله های **a** آنها مضارب ۳ باشند:

$$(b+c+aaa)^*$$

## فصل سوم

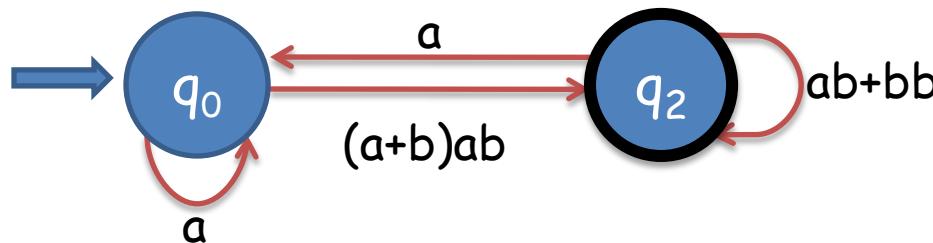
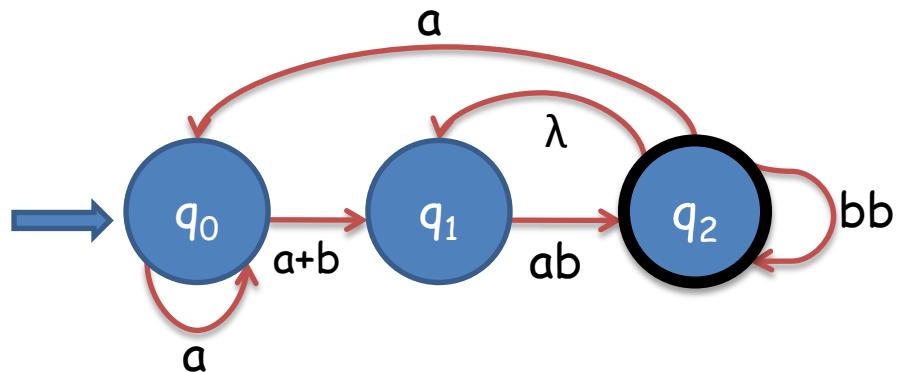
۳) برای زبان زیر یک DFA بیابید ؟

$$L(((aa^*)^*b)^*)$$



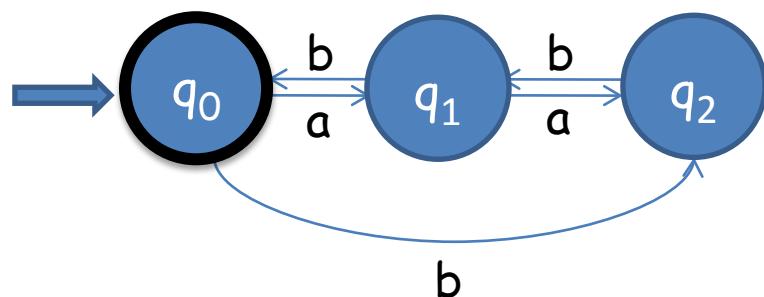
## فصل سوم

۴) تغییر وضعیت عام زیر را در نظر گرفته و سپس گراف تغییر وضعیت عام معادل آن را با کاهش یک وضعیت بیابید؟

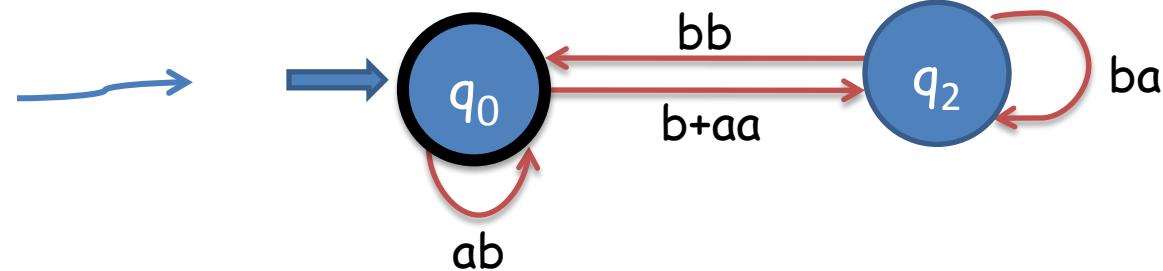


# فصل سوم

۵) برای زبان پذیرفته شده توسط آتماتون زیر، عبارت منظمی بیابید؟



گراف تغییر وضعیت  
عام با دو وضعیت



عبارت منظم

$((ab)^*((aa+b)(ba)^*(bb)^*)^*)^*$   
 $(ab+(b+aa)(ba)^*(bb))^*$

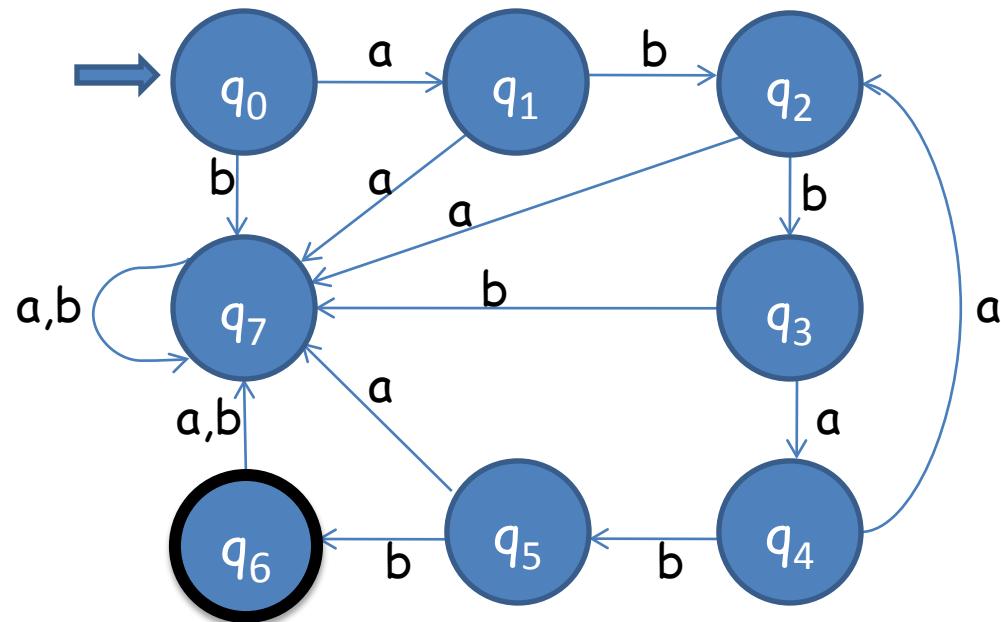
# فصل سوم

۶) یک DFA طراحی کنید که زبان پذیرفته شده توسط گرامر زیر را تولید کند ؟

$$\left\{ \begin{array}{l} S \rightarrow abA \\ A \rightarrow baB \\ B \rightarrow aA|bb \end{array} \right.$$

گرامر خطی

$$\left\{ \begin{array}{l} S \rightarrow Abb \\ A \rightarrow Aaba|abba \end{array} \right.$$



## فصل سوم

۷) گرامرهاي خطى راست و خطى چپ را برای زبان زير بنويسيد ؟

$$L = \{a^n b^m \mid n \geq 2, m \geq 3\}$$

: خطى چپ

$$\begin{array}{l} S \rightarrow Abbb \\ A \rightarrow Ab \mid Baa \\ B \rightarrow Ba \mid \lambda \end{array}$$

: خطى راست

$$\begin{array}{l} S \rightarrow aaA \\ A \rightarrow aA \mid bbbB \\ B \rightarrow bB \mid \lambda \end{array}$$

# فصل سوم

۸) یگ گرامر منظم برای زبان زیر بنویسید ؟

$$L = \{a^n b^m \mid m + n : \text{زوج}\}$$

$$\begin{array}{lcl} S & \xrightarrow{\quad} & Ab \mid D \\ A & \xrightarrow{\quad} & Abb \mid Ba \\ B & \xrightarrow{\quad} & Baa \mid \lambda \\ D & \xrightarrow{\quad} & Db \mid E \mid \lambda \\ E & \xrightarrow{\quad} & Eaa \mid \lambda \end{array}$$

# فصل سوم

۹) یک گرامر منظم بنویسید که تمام اعداد حقیقی پاسکال را تولید کند ؟

$$S \longrightarrow IdA \mid IdB \mid I.D$$

"ارقام بین "0 - 9"

$$A \longrightarrow dA \mid \lambda$$

$\pm 6$

$$B \longrightarrow dB \mid \cdot D$$

$\pm 6.$  خ

$$D \longrightarrow dA$$

$\pm 6.2$

$$I \longrightarrow + \mid - \mid \lambda$$

$\pm .6$

$$d \longrightarrow 0 \mid \dots \mid 9$$

# فصل چهارم

خواص زبانهای منظم

# فصل چهارم

## • خواص زبانهای منظم :

- وقتی عملیاتی برروی زبانهای منظم انجام می گیرد چه اتفاقی می افتد ؟ آیا زبان بدست آمده باز هم منظم است ؟
- عملیات  $\leftarrow$  عملیات ساده روی مجموعه ها (اتصال) و همه عملیاتی که رشته های زبان را تغییر دهد.
- خاصیت بستار (Closure)
- برای اینکه بدانیم زبانی منظم است، (زبان منظم را می توان با DFA طراحی و سپس پیاده سازی نمود). باید بدانیم زبانهای منظم از چه صفاتی برخوردارند.

## • خواص بستاری زبان های منظم :

- با وجود زبانهای منظم  $L_1$  و  $L_2$ ، آیا عملیاتی همچون اجتماع  $L_2 \cup L_1$ ، باز هم منظم است. (خانواده زبان های منظم تحت اجتماع بسته است).

## فصل چهارم

• قضیه :

- اگر  $L_1$  و  $L_2$  زبان های منظم باشند ،  $L_1 \cap L_2$  ،  $L_1 \cup L_2$  ،  $L_1^*$  ،  $\bar{L}_1$  نیز منظم هستند.

• مثال :

- آیا خانواده زبانهای منظم تحت تفاصل بسته است ؟ **بلی**

$L_1, L_2$  منظم  $\rightarrow L_1 - L_2$  منظم

$L_1 - L_2 = L_1 \cap \bar{L}_2$  ،  $L_2$  منظم  $\rightarrow \bar{L}_2$  منظم  $L_1, L_2$  منظم

• قضیه :

- خانواده زبان های منظم تحت معکوس کردن بسته است.

( منظم :  $L^R \rightarrow$  منظم :  $L$ )

## فصل چهارم

### • هم‌ریختی یا *Homomorphism* •

○ با فرض اینکه  $\Sigma$ ,  $\Gamma$ , الفبا باشند، تابعی مانند  $\Sigma \xrightarrow{*} \Gamma : h$  را هم‌ریختی گوییم.  
هم‌ریختی یک جایگزینی یک حرف با یک رشته است.  
می‌توان دامنه تابع  $h$  را به رشته‌ها بسط داد و خواهیم داشت:

$$w = a_1 a_2 \dots a_n$$

$$h(w) = h(a_1) h(a_2) \dots h(a_n)$$

- اگر  $L$  زبانی روی  $\Sigma$  باشد، تصویر هم‌ریختی (*Homomorphism Image*) عبارتست از:

$$h(L) = \{ h(w) \mid w \in L \}$$

## فصل چهارم

• مثال :

فرض کنید :

$$\Sigma = \{a, b\}, \Gamma = \{b, c, d\}, \begin{cases} h(a) = dbcc \\ h(b) = bdc \end{cases}$$

همریختی  $h(aba) = h(a)h(b)h(a) = dbccbdcbcc$

تصویر همریختی  $L\{aa, ba\} = \{dbccdbcc, bdcdcbcc\}$

$$r = (a+b^*)(aa)^*$$

عبارت منظم مربوط به  $h(L)$  عبارت منظم  $\rightarrow r_1 = (dbcc + (bdc)^*)(dbccdbcc)^*$

# فصل چهارم

• قضیه :

□ هرگاه  $L$  یک زبان منظم باشد ، هم‌ریختی  $h$  زبان  $L$  هم منظم می‌باشد.  
به عبارت دیگر ، خانواده زبان‌های منظم تحت هر هم‌ریختی دلخواه بسته است.

• خارج قسمت زبان‌ها :

□ اگر  $L_1$  و  $L_2$  زبان‌هایی برروی یک الفبای مشترک باشند ، خارج قسمت راست  $L_1$  به  $L_2$  عبارتست از :

$$L_1 / L_2 = \{ x \mid xy \in L_1, y \in L_2 \}$$

## فصل چهارم

• مثال : مطلوبست  $L_1 / L_2$  ؟

$$L_1 = \{ a^n b^m \mid n \geq 1, m \geq 0 \} \cup \{ ba \}$$

$$L_2 = \{ b^m \mid m \geq 1 \}$$

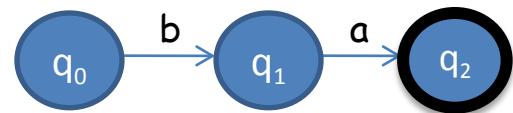
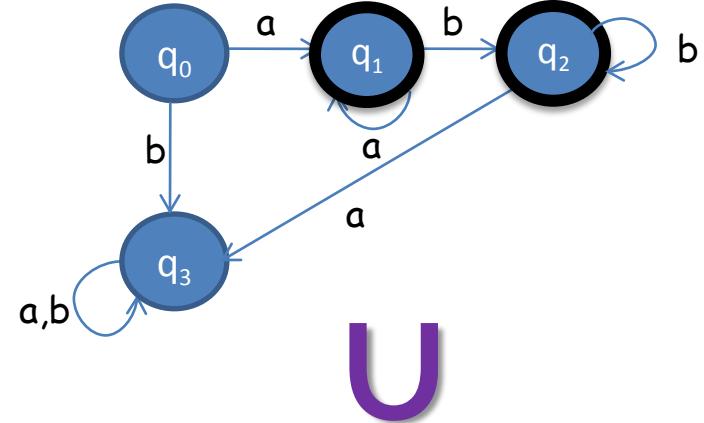
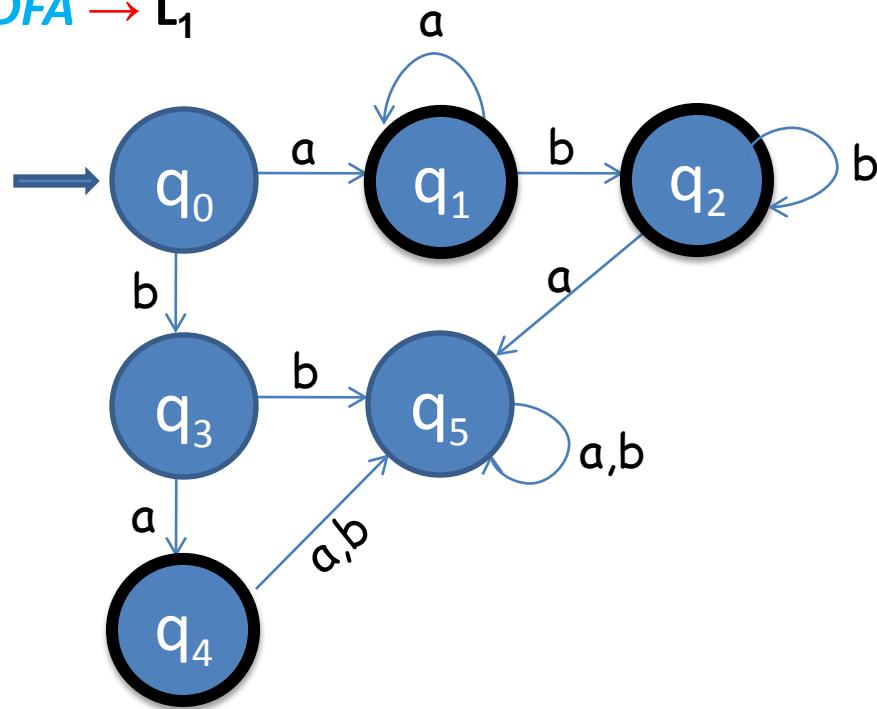
$$\xrightarrow{\quad\quad\quad} L_1 / L_2 = \{ a^n b^m \mid n \geq 1, m \geq 0 \}$$

به ازای هر  $q \in Q$  ، معین می کنیم که آیا راهی با برچسب  $v$  بگونه ای که باشد، به یک وضعیت نهایی وجوددارد یا خیر . اگر وجودداشته باشد، هر رشتہ  $X$  اگر برابر باشد، در  $L_2 / L_1$  خواهدبود.

# فصل چهارم

- لذا هریک از وضعیت های  $q_0$  الی  $q_5$  را تست می کنیم تا راهی با برچسب  $\{q_1, q_2, q_4\}$  به یکی از وضعیت های  $\{q_1, q_2, q_4\}$  (به خوبی  $(b^+)$  یا  $bb^*$ )  $\{b^m \mid m \geq 1\}$  وجود دارد یا خیر، که فقط  $q_2$  و  $q_1$  واجد شرایط هستند.

DFA  $\rightarrow L_1$

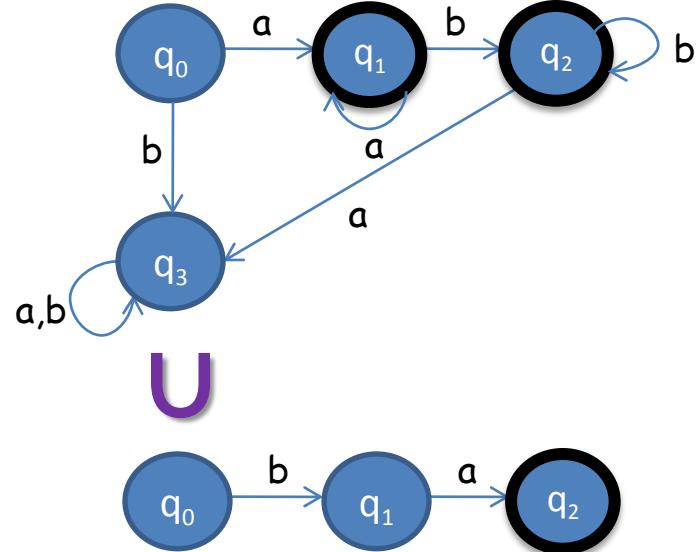
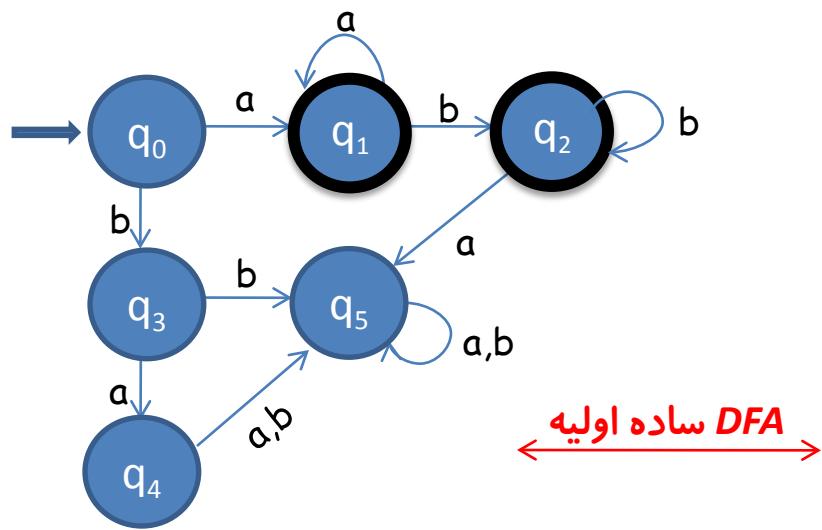


# فصل چهارم

آتماتای بدست آمده برای  $L_1 / L_2$  به شکل زیر خواهد بود :

$q_0 : bq_3bq_5 \dots , q_1 : bq_2bq_2 \dots$  (Acceptor) ,  $q_2 : bq_2bq_2 \dots$  (Acceptor)  
 $q_3 : bq_5bq_5 \dots , q_4 : bq_5bq_5 \dots$

DFA  $\rightarrow L_1 \setminus L_2$



## فصل چهارم

درمثال فوق  $L_1$  و  $L_2$  زبانهای منظم می باشند ولذا طبق قضیه زیر ،  $L_2 / L_1$  نیز منظم خواهد بود.

• **قضیه :**

اگر  $L_1$  و  $L_2$  زبان های منظم باشند ، آنگاه  $L_2 / L_1$  نیز منظم می باشد. بعبارت دیگر :  
«خانواده زبان های منظم تحت خارج قسمت راست نسبت به یک زبان منظم دیگر بسته است.»

• **مثال :** مطلوبست  $L_1 / L_2$  ؟

$$L_1 = L(a^*baa^*)$$

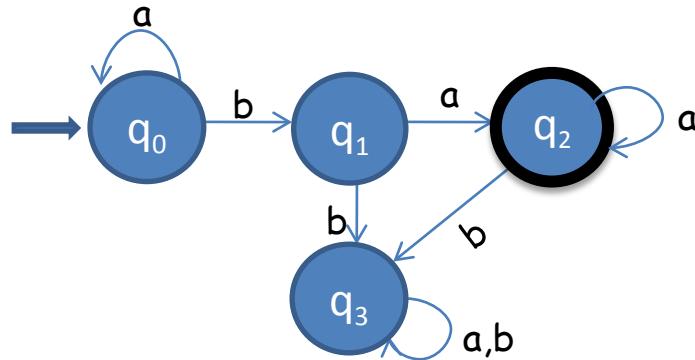
$$L_2 = L(ab^*)$$

• در زبان  $L_2$  ، حتماً یک  $a$  می آید، ولی می تواند  $b$  نیاید، بنابراین باقیستی  $a$  را از  $L_1$  حذف نمود.

# فصل چهارم

ابتدا  $DFA$  ای مربوط به  $L_1$  را مشخص می کنیم :

$DFA (L_1)$

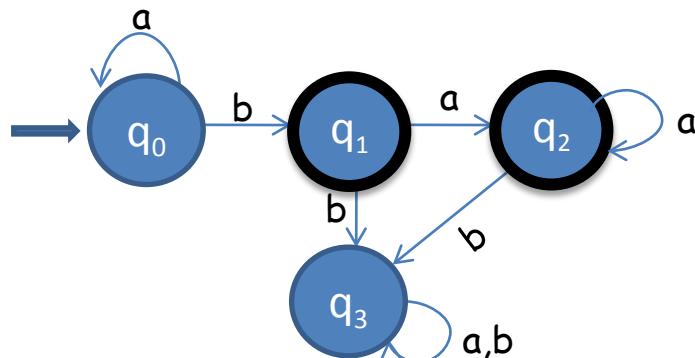


- $(a^*) : L(M_0) \cap L_2 = \emptyset$
- $(a^*b) : L(M_1) \cap L_2 = \{a\}$
- $(a^*baa^*) : L(M_2) \cap L_2 = \{a\}$
- $(a^*bb(a+b)^*) : L(M_3) \cap L_2 = \emptyset$

# فصل چهارم

بنابراین زبان پذیرفته شده توسط  $L_1 / L_2$  عبارت منظم زیر خواهد بود :

$$a^*b + a^*baa^* = a^*baa^* \rightarrow L_1 / L_2 = L(a^*baa^*)$$



$$\left. \begin{array}{l} \forall q \in Q = \{q_0, q_1, q_2, q_3\} \\ \text{برچسب } ab^* (L_2) \\ F = \{q_2\} \end{array} \right\} \rightarrow F = \{q_1, q_2\}$$

# فصل چهارم

## • قضیه :

اگر زبان منظم  $L$  بر روی الفبای  $\Sigma$  بصورت نمایش استاندارد و  $W \in \Sigma^*$  داده شده باشد، الگوریتمی برای تعیین اینکه  $W$  در  $L$  است یا خیر، وجود دارد که بدان **الگوریتم عضویت** گویند.

## • قضیه :

برای تعیین تهی بودن ، متناهی یا غیر متناهی بودن یک زبان منظم که با نمایش استاندارد داده شده است ، الگوریتمی وجود دارد.

## • اثبات :

۱) اگر مسیر ساده ای از رأس ابتدایی  $DFA$  به هریک از رأس های نهایی وجود داشته باشد، زبان غیر تهی است .

۲) تمامی رئوسی را که بر پایه یک چرخه هستند را پیدا نموده، اگر این رئوس بر روی یکی از مسیر های رأس مبدأ به یک رأس نهایی باشند، زبان متناهی است و در غیر این صورت غیر متناهی است .

# فصل چهارم

- **قضیه :**
- اگر دوزبان  $L_1$  و  $L_2$  را به صورت نمایش استاندارد داشته باشیم، آنگاه الگوریتمی برای تعیین اینکه آیا  $L_2 = L_1$  است یا خیر ، وجود دارد.
- **تشخیص زبانهای غیر منظم :**
- **اصل لانه کبوتر :**  $n$  شی را در  $m$  جعبه قرار دهیم به طوریکه  $m > n$ . (حداقل یک جعبه باید بیش از یک شی را در خود جای داده باشد.)
- **مثال :** آیا زبان  $L = \{a^n b^n \mid n \geq 0\}$  منظم است ؟ **خیر**
- **برهان خلف :** فرض کنیم  $L$  منظم است، بنابراین یک DFA بنام  $M = (Q, \{a, b\}, \delta, q_0, F)$  وجود دارد.

## فصل چهارم

تعداد نامحدودی  $\delta^*(q_0, a^i)$  وجود دارد، اما تعداد معینی وضعیت *state* در  $M$  وجود دارد: به ازای  $(i=1, 2, \dots)$

$$\delta^*(q_0, a^n) = q$$

$$\delta^*(q_0, a^m) = q \quad (m \neq n)$$

بنابر اصل لانه کبوتر ←

اما چون  $M$  عبارت  $a^n b^n$  را می‌پذیرد باید  $\delta^*(q, b^n) = q_f \in F$ ، لذا خواهیم داشت:

$$\delta^*(q_0, a^m b^n) = \delta^*(\delta^*(q_0, a^m), b^n) = \delta^*(q, b^n) = q_f$$

که این خلاف فرض اولیه است.  $M$  در صورتی  $a^n b^n$  را می‌پذیرد که  $n = m$  باشد.

**عبارتی دیگر:** یک آنماقی متناهی که دارای حافظه‌ای محدود است، و برای اینکه  $a^n b^n$  را بپذیرد، نیاز به حافظه نامتناهی دارد.

## فصل چهارم

### • لِم تزریق :

► دریک گراف تغییر وضعیت با  $n$  رأس ، هر راهی با طول  $n$  یا طولانی تر باید یک رأس را تکرار کند. (دارای چرخه)

### • قضیه :

► اگر  $L$  یک زبان نامتناهی باشد، آنگاه  $m \geq 0$  وجوددارد به طوریکه هر رشته  $w \in L$  ( $|w| \geq m$ ) ، میتواند به صورت زیر تجزیه گردد :

$$W = xyz \rightarrow |xy| \leq m \rightarrow |y| \geq 1$$
$$W_i = xy^i z \quad (i = 0, 1, \dots)$$

می گوییم رشته وسطی ( $y$ ) تزریق شده است .

## فصل چهارم

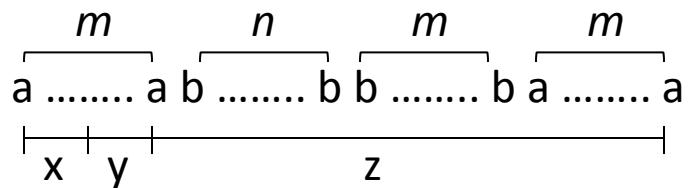
مثال : نشان دهید که  $L = \{a^n b^n \mid n \geq 0\}$  منظم نمی باشد ؟

فرض می کنیم  $L$  منظم است. مقدار  $m$  مجھول است اما همواره می توانیم  $n = m$  انتخاب نمائیم.  
بنابراین زیر رشته  $\gamma$  بایستی فقط از  $a$  تشکیل شده باشد.

فرض کنیم  $k = |y| = k$  ، آنگاه رشته ای که به ازای  $0 = i$  در معادله  $(W_i = xy^i z)$  بدل است  
بدست می آید که قطعاً در  $L$  نیست و مغایر با لم تزریق است و  $L$  منظم نیست.

# فصل چهارم

مثال : با فرض  $\Sigma = \{a,b\}$  نشان دهید که  $L = \{ww^R \mid w \in \Sigma^*\}$  منظم نیست؟



فقط از a تشکیل شده باشد : γ → حالت خاص

در رشته  $w_i = xyz$  به ازای  $(i=0)$  بنا براین در سمت چپ رشته تعداد کمتری a از سمت راست وجود دارد و نمی تواند به صورت  $ww^R$  باشد، بنابراین منظم نیست .

# فصل چهارم

**مثال :** نشان دهید که زبان  $L = \{w \in \Sigma \mid n_a(w) < n_b(w)\}$ ، با فرض  $\Sigma = \{a, b\}$ ، منظم نیست؟

$$\left. \begin{array}{l}
 \text{معلوم} \quad m \rightarrow w = a^m b^{m+1} \\
 \text{تعداد} \quad a, b
 \end{array} \right\} \longrightarrow \text{«فرض»} \quad y : a \text{ انتخاب حاوی فقط} \rightarrow \left[ \begin{array}{l}
 y = a^k, \quad 1 \leq k \leq m \\
 |y| \geq 1, \quad |xy| \leq m
 \end{array} \right]$$

$$w_i = xy^i z$$

$$\text{«فرض»} \quad i = 2 \rightarrow w_2 = \underbrace{a^{m+k}}_{a^m} \underbrace{b^{m+1}}_{a^k} \notin L \rightarrow L \text{ منظم نیست}$$

$\downarrow$   $\downarrow$   
 $xy$   
 $a^m \cdot a^k = a^{m+k}$

## فصل چهارم

• مثال : نشان دهید که زبان  $L = \{a^{n!} \mid n \geq 0\}$  منظم نیست ?

فرض  $m \rightarrow w = a^{m!}$  معلوم :

$$\left. \begin{array}{l} w_i = xy^iz \\ \end{array} \right\} \xrightarrow{\text{---}} |y| = k \leq m$$

طول رشته  $y$  / کل طول رشته

$$|xz| = m! - k \rightarrow j! = m! - k$$

غیر ممکن  $\leftarrow m! - k > (m-1)! : L$  داریم  $k \leq m, m > 2 \leftarrow$  زیرا

# فصل پنجم

زبانهای مستقل از متن

## فصل پنجم

### ❖ زبان های مستقل از متن : *Context Free Language (CFL)* :

مثال : زبان غیر منظم  $\{a^n b^n \mid n \geq 0\}$  صحیح ، ولی  $(( ))$  صحیح نیست.  
تعریف متغیر  $a, b$  زوج مرتب  $(, )$

زبان های مستقل از متن در زبان های برنامه سازی و کامپایلرها کاربرد فراوانی دارد.

### • گرامرهای مستقل از متن :

برای تولید گرامرهای قوی تر باید محدودیت ها را حذف نماییم. با حفظ محدودیت سمت چپ مولد (وبا مجاز ساختن هر چیزی در سمت راست مولد) به گرامرهای مستقل از متن می رسیم .

# فصل پنجم

▪ گرامر  $G(V, T, S, P)$  را در صورتی مستقل از متن می نامیم که همه قوانین  $P$  بصورت :

$$\left[ \begin{array}{l} A \rightarrow X \\ X \in (V \cup T)^*, A \in V \end{array} \right]$$

. $L = L(G)$  باشند و زبان  $L$  مستقل از متن است، اگر و فقط اگر گرامر مستقل از متن  $G$  موجود باشد بطوریکه

► هر گرامر منظم ، مستقل از متن است و در نتیجه زبان منظم نیز، زبان مستقل از متن است. ولی زبان  $\{a^n b^n \mid n \geq 0\}$  منظم نیست، ولی با گرامر مستقل از متنی بوجود می آید.

بنابر این : زبان های منظم زیرمجموعه ای از زبان های مستقل از متن است.

► در گرامرهای مستقل از متن می توان جایگزینی متغیرهای سمت چپ یک قانون را در هر زمانی که این متغیر در یک شکل جمله ای دیده می شود، انجام داد، و این بستگی به بقیه شکل جمله ندارد. (مجاز به انتخاب فقط یک متغیر در سمت چپ قانون هستیم.)

## فصل پنجم

مثال : گرامر  $G$  مستقل از متن است .

$$G = (\{S\}, \{a,b\}, S, P)$$

$$P \left\{ \begin{array}{l} S \rightarrow aSa \\ S \rightarrow bSb \\ S \rightarrow \lambda \end{array} \right.$$

$S \rightarrow aSa \rightarrow aaSaa \rightarrow aabbaa$  (منظم نیست) : نمونه اشتقاق

$$L(G) = \{ ww^R \mid w \in \{a,b\}^* \}$$

# فصل پنجم

مثال : گرامر  $G$  مستقل از متن است .

$$G = (\{S, A, B\}, \{a, b\}, S, P)$$
$$P \left\{ \begin{array}{l} S \rightarrow abB \\ A \rightarrow aaBb \mid \lambda \\ B \rightarrow bbAa \end{array} \right.$$

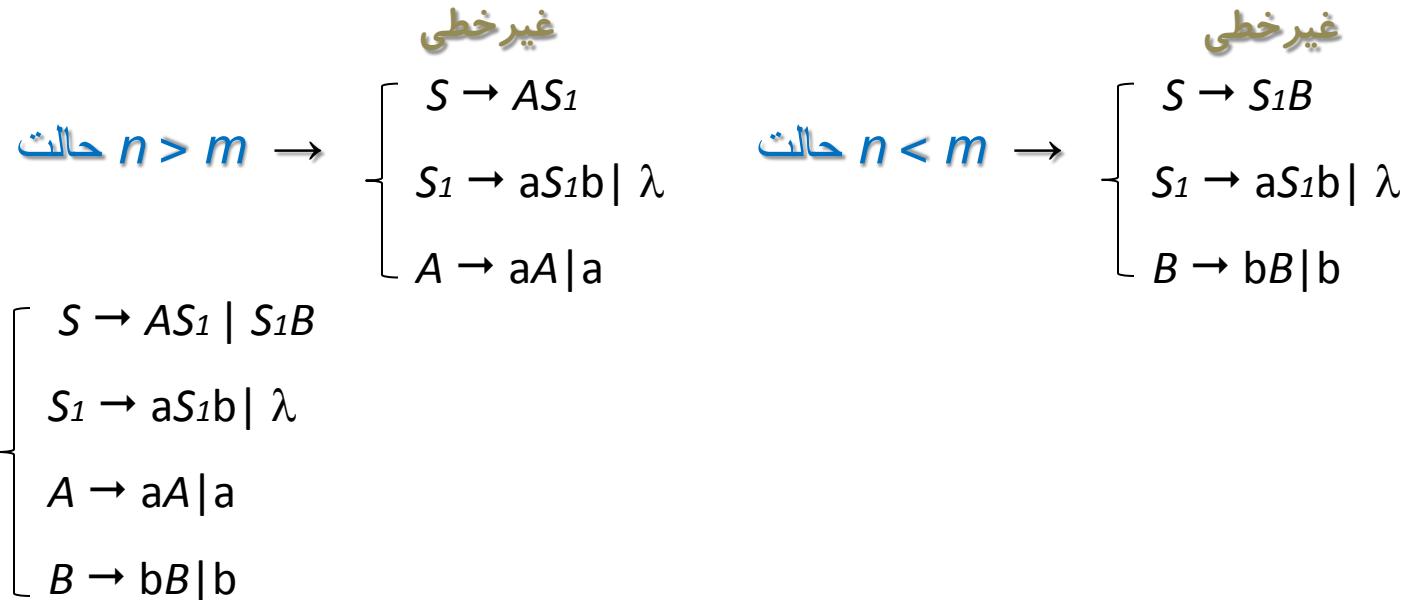
$$L(G) = \{ ab(bbba)^n bba(ba)^n \mid n \geq 0 \}$$

► دو گرامر فوق، گرامرهای **مستقل از متن و خطی** اند : گرامرهای منظم و خطی، مستقل از متن هستند، ولی یک گرامر مستقل از متن، لزوما خطی و یا منظم نیست.

# فصل پنجم

**مثال :** زبان  $L$  مستقل از متن است.

$$L(G) = \{ a^n b^m \mid n \neq m \}$$



**مثال :** گرامر  $G$  مستقل از متن است.

$$G = (\{S\}, \{a, b\}, S, P)$$

$$P : S \rightarrow SS \mid aSb \mid bSa \mid \lambda \rightarrow$$

$$L(G) = \{ w \in \{a, b\}^* \mid n_a(w) = n_b(w) \}$$

ارتباط این گرامر در زبانهای برنامه سازی است، هنگامی که در عبارات مانند:  $e^{*((a+b)-c)/d}$ ، تعداد  $( )$  با یستی یکسان باشد.

# فصل پنجم

## • اشتقاق های چپ و راست :

زبان های مستقل از متن که خطی نیستند  $\leftarrow$  اشتقاق ها، حاوی جمله هایی با بیش از یک متغیر هستند.

مثال :

$$G = (\{S, A, B\}, \{a, b\}, S, P)$$

$$P = \begin{cases} S \rightarrow AB \\ A \rightarrow aaA \mid \lambda \\ B \rightarrow Bb \mid \lambda \end{cases}$$

$$L(G) = \{ a^{2n}b^m \mid n \geq 0 \text{ \& } m \geq 0 \}$$

اشتقاق چپ 1)  $S \rightarrow AB \rightarrow aaAB \rightarrow aaB \rightarrow aaBb \rightarrow aab$

هر دو رشته های بالارشته های پکسان هستند.

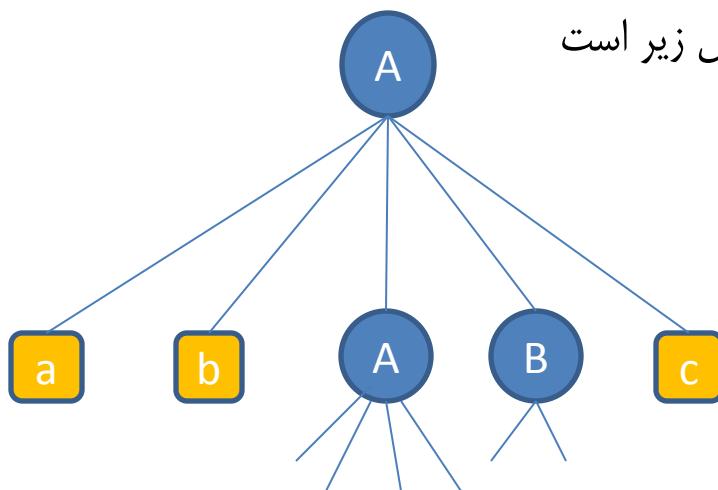
اشتقاق راست 2)  $S \rightarrow AB \rightarrow ABb \rightarrow aaABb \rightarrow aaAb \rightarrow aab$

# فصل پنجم

## • درخت اشتقاق :

یک درخت مرتب است که در آن گره ها با سمت چپ قوانین علامت گذاری می شوند و بچه های یک گره ، سمت راست آن قانون هستند.

مثال : درخت اشتقاق قانون  $A \rightarrow abABc$  به شکل زیر است



# فصل پنجم

## • تعریف درخت اشتقاق :

با فرض مستقل از متن بودن گرامر  $G = (V, T, S, P)$  ، یک درخت مرتب اشتقاق برای  $G$  وجود دارد، اگر و فقط اگر دارای خواص زیر باشد :

- ✓ ریشه ، برچسب  $S$  داشته باشد.
  - ✓ هر یک از برگ ها ، برچسبی از  $\{\lambda\} \cup T$  باشد.
  - ✓ هریک از گره های درونی(غیربرگ) ، دارای برچسبی از  $V$  باشد.
  - ✓ اگر گرهی با برچسب  $A \in V$  باشد و فرزندان آن از چپ به راست بصورت  $a_1, a_2, a_3, \dots, a_n$  باشند. برچسب گذاری شوند، آنگاه  $P$  دارای قانون  $A \rightarrow a_1, a_2, a_3, \dots, a_n$  می باشد.
  - ✓ برگی که برچسب  $\lambda$  داشته باشد ، هیچ خانواده ای ندارد ، یعنی گرهی که دارای فرزندی با برچسب  $\lambda$  باشد ، فرزند دیگری ندارد.
- درخت اشتقاق ، ذاتا مبهم است.

# فصل پنجم

## • درخت اشتقاق جزئی :

درختی که دارای خاصیت ۱, ۲, ۳, ۴ باشد و خاصیت ۵ لزوماً صدق نکند و خاصیت ۲ به شکل زیر باشد:  
هر برگ دارای برچسب  $\{\lambda\} \cup T \cup V$  باشد.

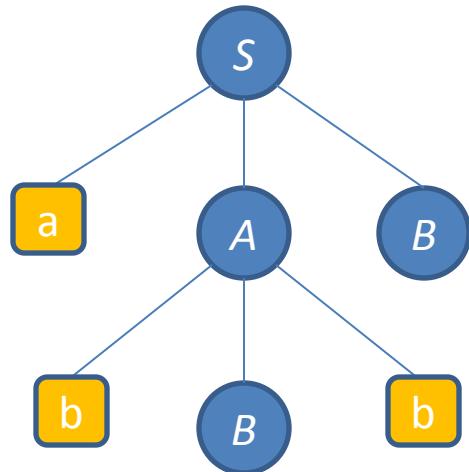
رشته ای که از خواندن برگ های درخت از چپ به راست ، پس از حذف ها بدست می آید ، **حاصل درخت** گویند.

مثال : درگرامی با قوانین زیر :

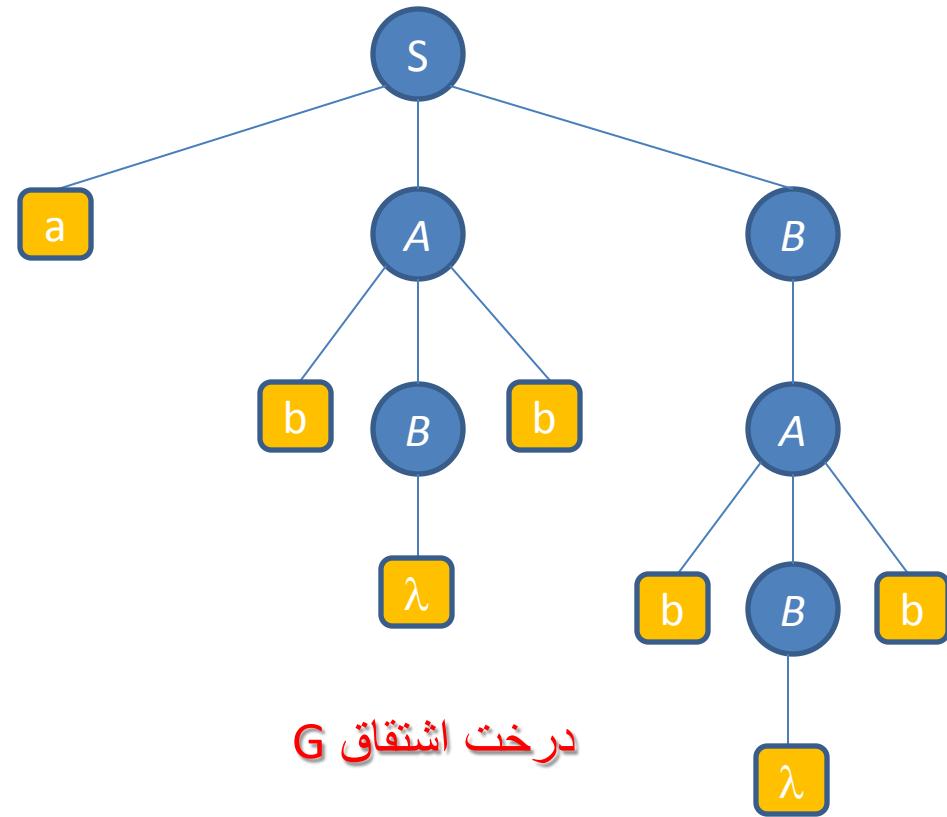
$$G = (\{S, A, B\}, \{a, b\}, S, P)$$

$$P \left\{ \begin{array}{l} S \rightarrow aAB \\ A \rightarrow bBb \\ B \rightarrow A|\lambda \end{array} \right.$$

# فصل پنجم



درخت اشتقاق جزئی



درخت اشتقاق  $G$

# فصل پنجم

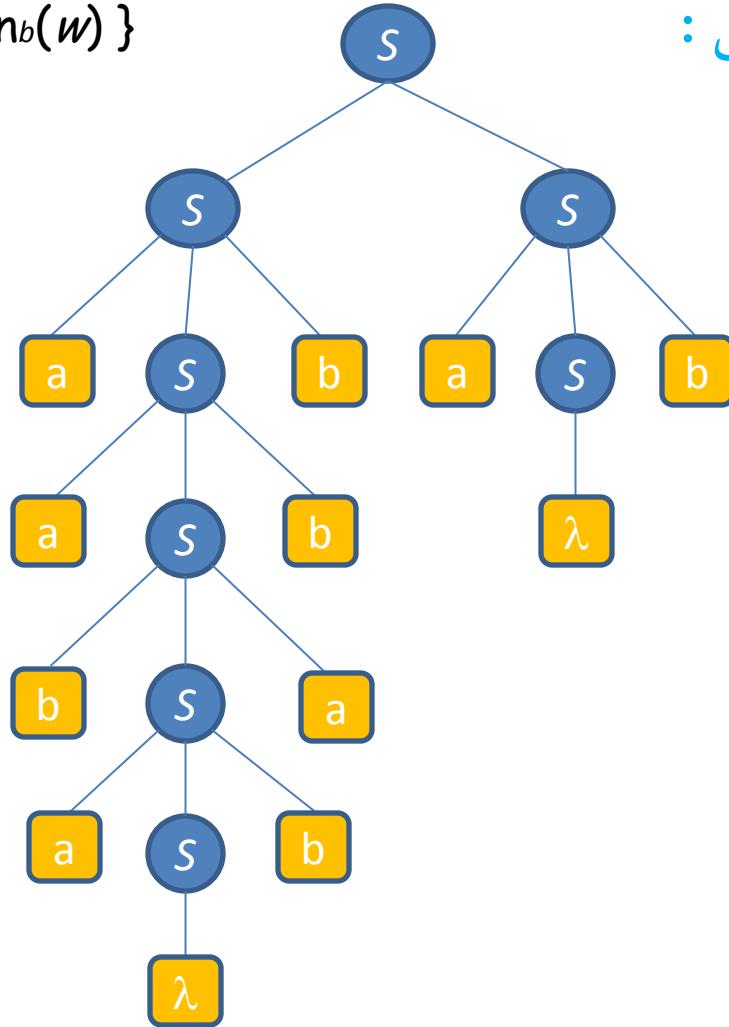
$$L(G) = \{ w \in \{a,b\}^* \mid n_a(w) = n_b(w) \}$$

مثال :

$$S \rightarrow aSb \mid bSa \mid SS \mid \lambda$$

رشته فرضی : aabababbab

اشتقاق چپ



# فصل پنجم

❖ قضیه :

با فرض مستقل از متن بودن گرامر  $\forall w \in L(G)$  ، به ازای هر  $G = (V, T, S, P)$  ، یک درخت اشتقاق برای  $G$  وجود دارد که حاصل آن  $W$  است و بالعکس.

❖ حاصل درخت اشتقاق در  $L(G)$  است.

❖ اگر  $t_G$  یک درخت اشتقاق جزئی برای  $G$  باشد ، آنگاه حاصل  $t_G$  یک شکل جمله ای از  $G$  می باشد.

# فصل پنجم

## پویش و گنگی (Parsing) :

پویش یعنی یافتن یک سری قانون که با استفاده از آن ها  $L(G) \subseteq W$  مشتق می شود.

### الگوریتم عضویت :

کلیه اشتقاق های چپ ممکن را میسازیم و ملاحظه میکنیم که آیا یکی از آن ها با  $W$  منطبق میشود یا خیر.

در دور اول ، همه ی قوانینی که به شکل  $X \rightarrow S$  هستند ، میابیم. اگر هیچ یک با  $W$  انطباق نداشت ، آنگاه :

در دور دوم ، همه ی قوانینی که قابل اعمال بر روی متغیر چپ همه ی  $X$  ها است ، میابیم.

این روش ، یک اشتقاق چپ از  $W$  ارائه می کند. این روش ، روش **پویش جستجوی کامل** است و نوعی پویش بالا به پایین (ریشه به برگ ها) می باشد .

# فصل پنجم

مثال : گرامر زیر را در نظر بگیرید.

$$S \rightarrow \begin{matrix} 1 & 2 & 3 & 4 \\ SS & | aSb & | bSa & | \lambda \end{matrix}$$

$w = aabb$

حذف دستور 3 و 4 (عدم انطباق با  $w$ )

دور اول  $\rightarrow P$

$$\left[ \begin{array}{l} 1. S \rightarrow SS \\ 2. S \rightarrow aSb \\ 3. S \rightarrow bSa \\ 4. S \rightarrow \lambda \end{array} \right]$$

# فصل پنجم

این قوانین حاصل دستور 1 بوده و از استقاق  $S$  درسمت چپ دستور 1 بدست آمده است. (حذف دستور 1.3)

$$P \rightarrow \text{دور دوم} \left[ \begin{array}{l} 1.1. S \rightarrow SS \rightarrow SSS \\ 1.2. S \rightarrow SS \rightarrow aSbS \\ 1.3. S \rightarrow SS \rightarrow bSaS \\ 1.4. S \rightarrow SS \rightarrow S \end{array} \right]$$

$$P \rightarrow \text{دور سوم} \left[ \begin{array}{l} 2.1. S \rightarrow aSb \rightarrow aSSb \\ 2.2. S \rightarrow aSb \rightarrow aaSbb \\ 2.3. S \rightarrow aSb \rightarrow abSab \\ 2.4. S \rightarrow aSb \rightarrow ab \end{array} \right]$$

حذف قوانین 2.3 و 2.4

$$S \rightarrow aSb \rightarrow aaSbb \rightarrow aabb \quad \text{رشته نهایی}$$

➤ همانطور که مشاهده می شود، این روش دارای مشکل است. مشکل نامعین بودن و عدم توقف آن است.

مشکل قانون  $\lambda \rightarrow S$  میباشد. از این قانون برای کاهش طول شکل های جمله ای استفاده میشود، ولی به سادگی نمیتوان تشخیص داد که چه زمانی باید توقف کرد. بنابراین قوانین  $A \rightarrow B$  ،  $A \rightarrow \lambda$  را نباید داشته باشیم.

# فصل پنجم

▪ **قضیه:** با فرض گرامر مستقل از متن بودن  $G = (V, T, S, P)$  که هیچ قانونی به شکل  $A \rightarrow \lambda$  و یا  $A \rightarrow B$ ، بطوریکه  $A, B \in V$ ، ندارد. روش پویش جستجوی کامل بصورت الگوریتمی وجود دارد که به ازای هر  $w \in \Sigma^*$ ، یا یک پویش از  $w$  تولید میکند و یا  $w \notin L(G)$  نخواهد بود. این الگوریتم هر رشته  $w \in L(G)$  را در تعداد مراحلی که متناسب با  $|w|^3$  است، پویش می کند.

## • گرامرساده (*Simple Grammar*) :

به گرامر مستقل از متن  $G = (V, T, S, P)$ ، گرامرساده می گویند که تمامی قوانین آن به شکل  $(A, a)$  حداکثر یکبار در  $P$  وجود داشته باشد. و هر زوج  $(A, a)$  که در آن  $A \in V$ ,  $a \in T$ ,  $X \in V^*$  وجود داشته باشد.

: مثال

$$1) S \rightarrow aSbSSc$$

*S-Grammar*

$$2) S \rightarrow aS | bSS | aSS | c$$

تکرار زوج  $(S, a)$

## فصل پنجم

✓ گرامر مستقل از متن  $G$  را در صورتی گنج می‌گوئیم که یک  $W \in L(G)$  وجود داشته باشد، که حداقل دو درخت استقاق متفاوت داشته باشد.

$$S \rightarrow aSb | SS | \lambda \quad \text{گنج است} \rightarrow (aaabbb)$$

مثال : گرامر  $G = (V, T, S, P)$  را با  $V = \{ E, I \}$  و  $T = \{ a, b, c, +, *, (, ) \}$  و  $P$  با قوانین :

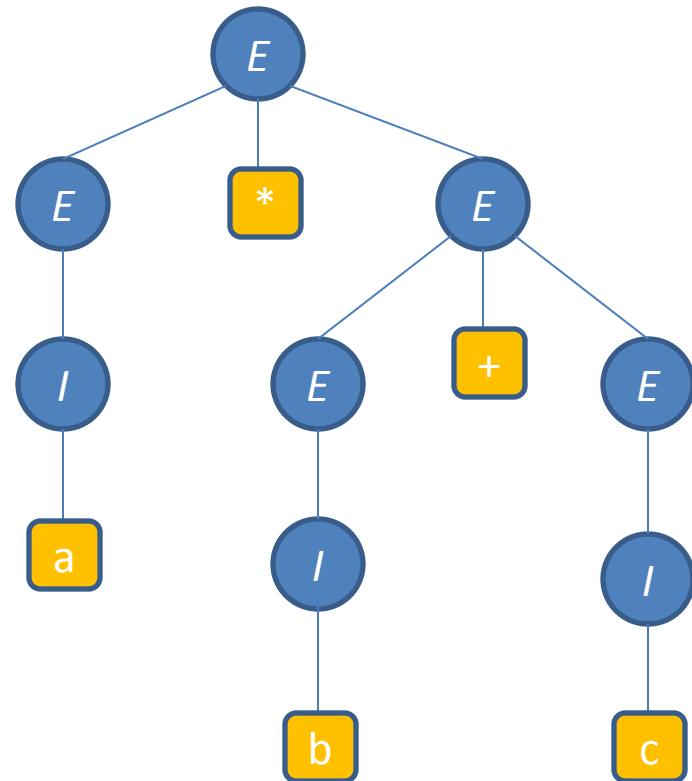
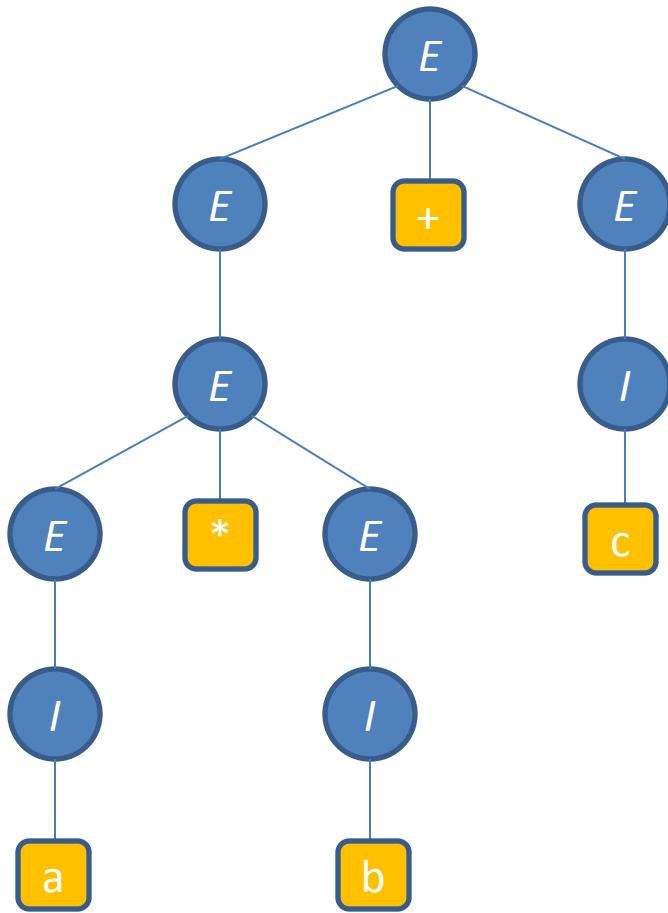
$$P \left\{ \begin{array}{l} E \rightarrow I \mid E+E \mid E^*E \mid (E) \\ I \rightarrow a \mid b \mid c \end{array} \right.$$

پیماش عبارتهای  $\rightarrow$  گنج است.

$$\left[ \begin{array}{l} a^*b+c \\ a+b^*c \end{array} \right]$$

# فصل پنجم

پیمایش  $a * b + c$



# فصل پنجم

✓ راه حل : اولویت و تقدم عملگرها

گرامر زیر غیرگنج است.

$$E \rightarrow T | E + T$$

$$T \rightarrow F | T^* F$$

$$F \rightarrow (E) | I$$

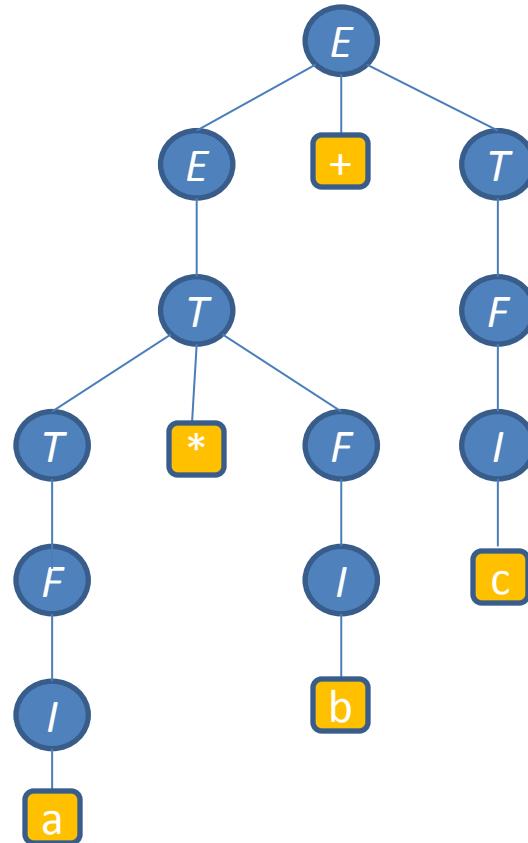
$$I \rightarrow a | b | c$$

$$\text{expr} \rightarrow \text{expr} + \text{term} | \text{expr} - \text{term} | \text{term}$$

$$\text{term} \rightarrow \text{term} * \text{factor} | \text{term} / \text{factor} | \text{factor}$$

$$\text{factor} \rightarrow (\text{expr}) | \text{digit}$$

$$\text{digit} \rightarrow 0 | 1 | \dots | 9$$



# فصل پنجم

► بعضی از زبانها ذاتا گنگ هستند و گرامر غیرگنگی برای آنها وجود ندارد.

مثال : زبان  $\{a^n b^n c^m\} \cup \{a^n b^m c^m\}$  مستقل ازمن و ذاتا گنگ است .

$L \rightarrow L_1 \cup L_2 ; S \rightarrow S_1 | S_2$  گرامر مستقل ازمن و گنگ است.

$$L_1 \left\{ \begin{array}{l} S_1 \rightarrow S_1c | A \\ A \rightarrow aAb | \lambda \end{array} \right. \quad \begin{array}{l} \text{تعداد } a, b \text{ را محدود می کند .} \\ \text{تعداد } S_1 : S \rightarrow S_1 \end{array}$$
$$L_2 \left\{ \begin{array}{l} S_2 \rightarrow aS_2 | B \\ B \rightarrow bAc | \lambda \end{array} \right. \quad \begin{array}{l} \text{تعداد } c \text{ را محدود می کند .} \\ \text{تعداد } S_2 : S \rightarrow S_2 \end{array}$$

# فصل پنجم

▪ زبان های مستقل از متن و زبان های برنامه سازی :

▪ نماد *BNF* :

$<expression> ::= <term> \mid <expression> + <term>$        $T = \{+, *\}$   
 $<term> ::= <factor> \mid <term> * <factor>$

استفاده از کلمات کلیدی  $T = \{\text{if}\}$

var  $x, y : \text{real} ;$       var  $x : \text{integer} ;$   
                 $x, z : \text{integer} ;$        $x := 3.2 ;$

$<\text{if-statement}> ::= \text{if } <\text{expression}> <\text{then-clause}> <\text{else-clause}>$

# فصل پنجم

## • تمرینات فصل پنجم :

✓ برای هریک از زبان‌های زیر یک گرامر مستقل از متن بنویسید؟

$$1) L = \{ a^n b^m \mid n \neq m - 1 \}$$

$$\begin{cases} S \rightarrow aSb | aA | bB \\ A \rightarrow aA | \lambda \\ B \rightarrow bB | \lambda \end{cases}$$

$$2) L = \{ w \in (a,b)^* \mid n_a(w) >= n_b(w) \}$$

راه حل اول :  $n > m-1 \rightarrow ab, aab, \dots$

راه حل دوم :  $n < m-1 \rightarrow b, abb, abbb, \dots$

$$\begin{cases} S \rightarrow aSb | aS | SS | \lambda \end{cases}$$

$$3) L = \{ a^n b^m c^k \mid k = m + n \}$$

$$\begin{cases} S \rightarrow aSc | A \\ A \rightarrow bAc | \lambda \end{cases}$$

# فصل پنجم

4)  $L = \{ a^n b^m c^k \mid n = m \text{ or } m \neq k \}$

حالات	حالات	حالات
$n=m$	$m>k$	$m< k$
$S \rightarrow MD   ABE   AHD$		
$M \rightarrow aMb   \lambda$		
$D \rightarrow cD   \lambda$		
$A \rightarrow aA   \lambda$		
$B \rightarrow bB   \lambda$		
$E \rightarrow bEc   b$		
$H \rightarrow bHc   c$		

5)  $L = \{ a^n w w^R b^n \mid w \in (a,b)^* \text{ & } n \geq 1 \}$

$S \rightarrow aSb   aAb$
$A \rightarrow aAa   bAb   \lambda$

# فصل پنجم

۶) نشان دهید که زبان زیر مستقل از متن است؟

$$L = \{ uvwv^R \mid u, v, w \in (a, b)^+ \text{ and } |u| = |w| = 2 \} \quad \left\{ \begin{array}{l} S \rightarrow AB \\ A \rightarrow aa \mid bb \mid ab \mid ba \\ B \rightarrow aBa \mid bBb \mid aAa \mid bAb \end{array} \right.$$

۷) یک گرامر ساده برای زبان  $L = \{ a^n b^n \mid n \geq 1 \}$  بیابید؟

$$\left\{ \begin{array}{l} S \rightarrow aA \\ A \rightarrow aAB \mid b \\ B \rightarrow b \end{array} \right. \quad \left\{ \begin{array}{l} S \rightarrow aSb \mid ab \end{array} \right.$$

۸) یک گرامر غیرگنگ ارائه دهید که مجموعه تمام عبارات منظم روی برای زبان  $\Sigma = \{a, b\}$  را تولید کند؟

$$\left\{ \begin{array}{l} S \rightarrow T \mid S+T \\ T \rightarrow F \mid T.F \\ F \rightarrow I \mid (S) \mid S^* \mid S^+ \\ I \rightarrow a \mid b \mid \lambda \end{array} \right.$$

# فصل پنجم

(۹) یک عبارت BNF برای while در پاسکال ارائه دهید؟

```
<while exp> ::= while <compare exp> do begin <expression> end ;  
<compare exp> ::= <compare exp> and <compare exp> | <compare exp> or <compare exp>  
<compare exp> ::= not <compare exp>  
<compare exp> ::= ( <compare exp> )  
<compare exp> ::= <expression> <compare sign> <expression>  
<compare sign> ::= > | >= | < | <= | = | <>  
<expression> ::= <expression> <expression> | <variable> := <expression> ;  
<expression> ::= <expression> <sign> <expression>  
<sign> ::= + | - | * | / | div | mod  
<expression> ::= <variable> | <numeric>  
<expression> ::= ( <expression> )  
<variable> ::= <alphabet> <var>  
<alphabet> ::= A | ... | Z | a | ... | z  
<var> ::= <digit> <var>  
<var> ::= _ <var> | λ  
<numeric> ::= <digit> <numeric> | <digit>  
<digit> ::= 0 | 1 | 2 | ... | 9
```

# فصل ششم

ساده سازی گرامرهاي مستقل از متن  
و فرم های نرمال

# فصل ششم

## ■ هدف :

تبديل گرامرهای مستقل از متن به گرامرهای معادلی که محدودیت های خاصی را رعایت نمایند.

## ■ روش های تبدیل گرامرها :

(۱) حذف  $\lambda$  از گرامرها : فرض کنید  $L$  یک زبان مستقل از متن و  $G = (V, T, S, P)$  یک گرامر مستقل از متن برای  $\{\lambda\} - L$  باشد. آنگاه گرامری که با افزودن  $S_0$  به  $V$  بعنوان نشانه شروع در قانون  $S_0 \rightarrow S|\lambda$  بدست می آید، زبان  $L$  را تولید می کند. بعلاوه، اگر گرامر مستقل از متن  $G$  را داشته باشیم، می توان  $\hat{G}$  را طوری بیابیم که  $L(\hat{G}) = L(G) - \lambda$ .

- ✓ در یک گرامر مستقل از متن، هر قانونی که به شکل  $A \rightarrow \lambda$  باشد، قانون  $\lambda$  گویند.
- ✓ هر متغیری مانند  $A$  که برای آن اشتقاء  $A \rightarrow \lambda \rightarrow \dots$  ممکن باشد، *Nullable* گویند.
- ✓ یک گرامر میتواند زبانی را که حاوی  $\lambda$  نباشد، تولید کند و در عین حال دارای قوانین  $\lambda$  یا متغیرهای *Nullable* باشد.

## فصل ششم

مثال : در گرامرها زیر، متغیر  $\lambda$  را حذف کنید؟

$$P_1 \left[ \begin{array}{l} S \rightarrow ABb|\lambda \\ A \rightarrow aA|\lambda \\ B \rightarrow bB|\lambda \end{array} \right] \quad \xrightarrow{\hspace{1cm}} \quad P'_1 \left[ \begin{array}{l} S \rightarrow Abb|Ab|Bb|b|AB|A|B|\lambda \\ A \rightarrow aA|a \\ B \rightarrow bB|b \end{array} \right]$$

$$P_2 \left[ \begin{array}{l} S \rightarrow aS_1b \\ S_1 \rightarrow aS_1b|\lambda \end{array} \right] \quad \xrightarrow{\hspace{1cm}} \quad P'_2 \left[ \begin{array}{l} S \rightarrow aS_1b|ab \\ S_1 \rightarrow aS_1b|ab \end{array} \right]$$

# فصل ششم

۲) **قانون جایگزینی سودمند** : حذف بعضی از قوانین نامطلوب که تنها بخاطر کاهش تعداد قوانین صورت نمی پذیرد.

**قضیه** : گرامر مستقل ازمن  $G = (V, T, S, P)$  با قوانین  $x_1 B x_2 \rightarrow A$  را درنظر گرفته وفرض نمایید که مجموعه کلیه قوانین  $P$  که  $B$  را درسمت چپ خود دارد ، باشند.

✓ اگر  $\hat{G} = (V, T, S, \hat{P})$  گرامری باشد که درآن  $\hat{P}$  با حذف  $A$  از  $P$  و افزودن قانون  $. L(\hat{G}) = L(G)$ .

**قضیه** : با فرض مستقل ازمن بودن گرامر  $G = (V, T, S, P)$  بدون قوانین  $\lambda$ ، آنگاه گرامر مستقل ازمن دیگری  $\hat{G} = (V, T, S, \hat{P})$  وجود دارد که معادل گرامر  $G$  است و هیچ قانون یکه ای ندارد.

## ۲.۱) حذف قانون Chain

شامل گرامرهایی است که متغیرها درآن بصورت  $A, B \in V$  و  $A \rightarrow B$  که درآنها  $A \rightarrow B$  و  $B \rightarrow A$  بتعارض باشند.

**۲.۲) حذف قانون یکه** : هر قانونی که به شکل  $A \rightarrow B$  که درآن  $A, B \in V$  باشد، یک قانون یکه است.

# فصل ششم

**مثال :** در گرامر زیر، متغیر  $B$  را بطور سودمندی جایگزین کنید؟

$$P \left[ \begin{array}{l} A \rightarrow a | aaA | abBc \\ B \rightarrow abbA | b \end{array} \right] \longrightarrow P' \left[ \begin{array}{l} A \rightarrow a | aaA | ababbAc | abbc \end{array} \right]$$

$$\xrightarrow{\text{aaabbc رشته}} P \left[ \begin{array}{l} A \rightarrow aaA \rightarrow aaabBc \rightarrow aaabbc \\ A \rightarrow aaA \rightarrow aaabbc \end{array} \right]$$

**مثال :** در گرامر زیر، قوانین یکه را حذف نمایید؟

$$P \left[ \begin{array}{l} S \rightarrow Aa | B \\ A \rightarrow a | bc | B \\ B \rightarrow A | bb \end{array} \right] \xrightarrow{\text{قوانین یکه}} S \rightarrow B \rightarrow A \xrightarrow{\text{قوانین یکه}} \left[ \begin{array}{l} S \rightarrow A \\ S \rightarrow B \\ B \rightarrow A \\ A \rightarrow B \end{array} \right]$$

$$\begin{array}{l} \text{قوانین غیریکه} \\ \left[ \begin{array}{l} S \rightarrow Aa \\ A \rightarrow a | bc \\ B \rightarrow bb \end{array} \right] + \text{قوانین جدید} \left[ \begin{array}{l} S \rightarrow a | bc | bb \\ A \rightarrow bb \\ B \rightarrow a | bc \end{array} \right] \xrightarrow{\text{قوانین جدید}} \left[ \begin{array}{l} S \rightarrow a | bc | bb | Aa \\ A \rightarrow a | bc | bb \\ B \rightarrow a | bc | bb \end{array} \right] \end{array}$$

## فصل ششم

۳) حذف قوانین بی فایده: برای بی فایده بودن یک قانون به دو دلیل نیازداریم. اولا ، منجره تولید رشته ای در زبان نشود. ثانیا ، از وضعیت شروع قابل دستیابی نباشد.

■ **قضیه :** بافرض مستقل بودن از متن گرامر  $G=(V,T,S,P)$  ، متغیر  $A \in V$  را مفید گوییم ، اگر و فقط اگر حداقل یک  $w \in L(G)$  باشد ، بطوریکه :  $x,y \in (V \cup T)^*$  ،  $S \rightarrow xAy \rightarrow w$

✓ متغیری که قابل استفاده نباشد ، بی فایده (*Useless*) ، و قوانینی که دارای متغیر بی فایده باشد ، قانون بی فایده نامیده می شود.

**مثال :** در گرامر  $\begin{cases} S \rightarrow aSb|A|\lambda \\ A \rightarrow aA \end{cases}$  با حرکت از  $S$  به طرف  $A$  و با استفاده از قانون  $S \rightarrow A$  هیچ رشته ای تولید نخواهد شد. یعنی، بودن یا نبودن متغیر  $A$  و قانون مربوطه اش ، هیچ تاثیری روی زبان ندارد و می توان آن را حذف کرد.

## فصل ششم

قانون  $B \rightarrow bA$  ، هیچ مسیری از وضعیت شروع  $S$  به متغیر  $B$  وجود ندارد.

$$\text{مثال : در گرامر} \quad \left\{ \begin{array}{l} S \rightarrow A \\ A \rightarrow aA | \lambda \\ B \rightarrow bA \end{array} \right.$$

لذا بودن و نبودن قانون مربوطه اش، هیچ تاثیری روی زبان ندارد و می توان آن را حذف نمود. متغیر  $B$  بی فایده و قانون  $B \rightarrow bA$  ، قانون بی فایده می باشد.

**مثال :** قوانین بی فایده را از گرامر  $G = (\{S,A,B,C\}, \{a,b\}, S, P)$  حذف نمایید؟

$$P \left\{ \begin{array}{l} S \rightarrow aS | A | C \\ A \rightarrow a \\ B \rightarrow aa \\ C \rightarrow aCb \end{array} \right. \quad \begin{array}{l} \text{(1) شناسائی متغیرهایی که به رشته های ترمینال منجر می شوند.} \\ \text{متغیر } C \text{ بی فایده است.} \end{array}$$

$$G_1 = (\{S,A,B\}, \{a\}, S, P_1)$$

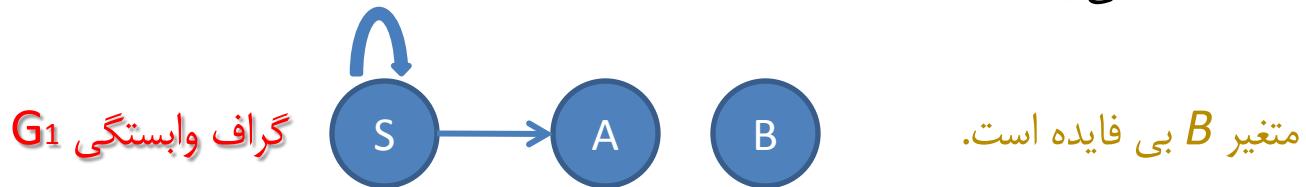
$$P_1 \left\{ \begin{array}{l} S \rightarrow aS | A \\ A \rightarrow a \\ B \rightarrow aa \end{array} \right.$$

# فصل ششم

(2) شناسایی متغیرهایی که از وضعیت شروع قابل دستیابی نباشند.

## استفاده از گراف وابستگی (Dependency Graph)

در گراف وابستگی برای یک زبان مستقل از متن، رئوس، متغیرهای گرامر، و یال ها، قوانینی به شکل  $C \rightarrow xDy$  می باشند.



$$G_2 = (\{S, A\}, \{a\}, S, P_2) \quad P_2 \left[ \begin{array}{l} S \rightarrow aS|A \\ A \rightarrow a \end{array} \right] \xrightarrow{\text{حایگزینی سودمند}} \left[ \begin{array}{l} S \rightarrow aS|a \end{array} \right]$$

■ **قضیه** : با فرض مستقل از متن بودن گرامر  $G = (V, T, S, P)$  ، گرامری معادل گرامر  $G$  بنام  $G^\wedge$  وجود دارد که دارای هیچ متغیر یا قانون بی فایده نیست.

# فصل ششم

## ▪ فرم های نرمال (Normal Forms)

### ۱) فرم نرمال چامسکی :

گرامری در این فرم است که همه قواعد آن به یکی از دو صورت زیر یا ترکیبی از هردو باشد :

$$\left\{ \begin{array}{l} A \rightarrow BC \\ A \rightarrow a \end{array} \right. \quad A, B, C \in V, a \in T$$

مثال : گرامر  $\left[ \begin{array}{l} S \rightarrow AS|AAS \\ A \rightarrow SA|aa \end{array} \right]$  در فرم نرمال چامسکی است و گرامر  $\left[ \begin{array}{l} S \rightarrow AS|a \\ A \rightarrow SA|b \end{array} \right]$

در فرم نرمال چامسکی نیست. (در این مثال، گرامر دارای قوانین  $\lambda$  و یکه نیست.)

▪ قضیه : هر گرامر مستقل از متن  $G = (V, T, S, P)$  که در آن  $\lambda \notin L(G)$  باشد ، دارای گرامر معادل  $\hat{G} = (V, T, S, P)$  در فرم نرمال چامسکی است.

# فصل ششم

► ساختن گرامر  $\hat{G}$  در طی دو مرحله به شرح زیر انجام می‌گیرد :

## ○ گام اول :

ساختن گرامر  $(G_1 = (V_1, T, S, P_1))$  با قوانینی به شکل  $(A \rightarrow X_1X_2\dots X_n)$  (  $X_i \in VUT$  ) ، از  $G$  با قوانینی به شکل  $($

قانون در  $P_1$  قرار می‌گیرد اگر  $n=1$   $\longrightarrow$   $X_1$  (چون قانون یکه وجود ندارد) ترمینال

اگر  $n>1$   $\longrightarrow$   $B_a$  ،  $\forall a \in T$  تعریف متغیر جدید

قانون در  $P_1$  قرار می‌گیرد  $\forall P_1$   $\longrightarrow A \rightarrow X_1X_2\dots X_n \longrightarrow A \rightarrow C_1C_2\dots C_n$

متغیر  $\forall B_a \longrightarrow B_a \rightarrow a$  اگر  $C_i = X_i$   $X_i \in V$  اگر  $C_i = B_a$   $X_i \in a$

این قسمت از الگوریتم ، تمامی ترمینالها را از قوانینی که سمت آنها طول بزرگتر از یک دارد ، با جایگزینی آنها با متغیرهای جدید ، حذف می‌کند .

$G_1: A \rightarrow a$   $A \rightarrow C_1C_2\dots C_n$  ( $C_i \in V_i$ )

$L(G) = L(G_1)$

# فصل ششم

## ○ گام دوم :

ابتدا همه قوانین به شکل  $A \rightarrow a$  و سپس همه قوانین به شکل  $A \rightarrow C_1C_2...C_n$  را با  $n=2$  در

قرار می دهیم. (اعمال قانون  $A \rightarrow BC$ )

$$P \left\{ \begin{array}{l} A \rightarrow C_1D_1 \\ D_1 \rightarrow C_2D_2 \\ \dots \\ D_{n-2} \rightarrow C_{n-1}C_n \end{array} \right.$$

**مثال :** گرامری با قوانین زیر را به فرم نرمال چامسکی تبدیل نمایید؟

(در این مثال، گرامر دارای قوانین  $\lambda$  و یکه نیست.)

$G: < \{S, A, B\}, \{a, b, c\}, \{S\}, P >$

$$P \left\{ \begin{array}{l} S \rightarrow ABa | B \\ A \rightarrow aab \\ B \rightarrow Ac \end{array} \right.$$

# فصل ششم

$$\left\{ \begin{array}{l} S \rightarrow ABB_a \\ A \rightarrow BaB_aB_b \\ B \rightarrow Ab_c \\ B_a \rightarrow a \\ B_b \rightarrow b \\ B_c \rightarrow c \end{array} \right.$$

○ گام اول : اعمال قانون  $A \rightarrow a$  متغیرهای  $B_a, B_b, B_c$

$$\left\{ \begin{array}{l} S \rightarrow AD_1 \\ D_1 \rightarrow BB_a \\ A \rightarrow B_aD_2 \\ D_2 \rightarrow B_aB_b \\ B \rightarrow AB_c \\ B_a \rightarrow a \quad \& \quad B_b \rightarrow b \quad \& \quad B_c \rightarrow c \end{array} \right.$$

○ گام دوم : اعمال قانون  $A \rightarrow BC$

## فصل ششم

### ۲) فرم نرمال گریباخ :

یک گرامر مستقل از متن است که در آن تمامی قوانین بصورت  $a \in T, X \in V^*$ ، بطوریکه  $A \rightarrow aX$  باشد.

- ✓ محدودیت روی محل قرارگیری متغیرها و ترمینال‌ها می‌باشد.
- ✓ لزومی ندارد زوج  $(A, a)$  فقط یکبار تکرار شده باشد (گرامرهای ساده) و این محدودیت را از بین می‌برد.
- ✓ مورد کاربرد آن در کامپایلرها، نیاز به متغیر  $\text{First}(A)$  برای انتخاب مولد مورد نظر، می‌باشد.

**مثال :** گرامر  $G_1$  در فرم نرمال گریباخ نیست، ولی گرامر  $G_2$  در فرم نرمال گریباخ می‌باشد.

$$G_1 \left\{ \begin{array}{l} S \rightarrow AB \\ A \rightarrow aA | bB | b \\ B \rightarrow b \end{array} \right.$$

$$G_2 \left\{ \begin{array}{l} S \rightarrow aAB | bBB | bB \\ A \rightarrow aA | bB | b \\ B \rightarrow b \end{array} \right.$$

## فصل ششم

**مثال :** گرامر  $S \rightarrow abSb|aa$  را به فرم نرمال گریباخ تبدیل نمایید?  
(همانند فرم نرمال چامسکی عمل می نماییم.)

$$\left\{ \begin{array}{l} S \rightarrow aBSB|aA \\ A \rightarrow a \\ B \rightarrow b \end{array} \right.$$

متغیرهای  $A, B$  را که متراffد با  $a, b$  هستند، جایگزین می کنیم.

**قضیه :** به ازای هر گرامر مستقل از متن  $G=(V,T,S,P)$  که در آن  $\lambda \notin L(G)$  باشد، دارای گرامر معادل  $\hat{G}=(V,T,S,P)$  در فرم نرمال گریباخ است. ■

# فصل ششم

## الگوریتم عضویت برای گرامرهای مستقل از متن :

- ✓ شرط اصلی : الگوریتم CYK در صورتی کار می کند که گرامر در فرم نرمال چامسکی باشد.
- ✓ فرض کنید گرامر  $G = (V, T, S, P)$  در فرم نرمال چامسکی بوده و رشته  $w = a_1 a_2 \dots a_n$  را داریم:

زیررشته  $w = a_i \dots a_j$

زیرمجموعه از  $V$   $V_{ij} = \{A \in V \mid A \xrightarrow{*} w_{ij}\}$

نکته ۱ : واضح است که  $w \in L(G)$  است ، اگر و فقط اگر  $S \in V_{1n}$  باشد. ○

نکته ۲ : است ، اگر و فقط اگر  $G$  حاوی قانونی مانند  $A \rightarrow a_i$  باشد. ○

$$V_{ij} = \bigcup_{k \in \{i, i+1, \dots, j-1\}} \{A \mid A \xrightarrow{*} BC ; B \in V_{i,k}, C \in V_{k+1,j}\}$$

$V_{1,1}, V_{2,2}, \dots, V_{n,n}$  .1 را محاسبه کنید.

$V_{1,2}, V_{2,3}, \dots, V_{n-1,n}$  .2 را محاسبه کنید.

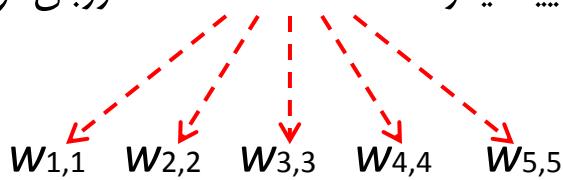
$V_{1,3}, V_{2,4}, \dots, V_{n-2,n}$  .3 را محاسبه کنید.

... .4

# فصل ششم

**مثال :** تعیین نمایید آیا رشته  $w = aabbba$  در زبان تولید شده توسط گرامر زیر وجود دارد یا خیر ؟

$$\left\{ \begin{array}{l} S \rightarrow AB \\ A \rightarrow BB | a \\ B \rightarrow AB | b \end{array} \right.$$



( محاسبه  $V_{1,1}, V_{2,2}, \dots, V_{n,n}$  )

$w_{1,1} = a \rightarrow V_{1,1} = \{A\}$  → مجموعه متغیرهایی است که بطور بلا فصل a را می دهد :

$w_{2,2} = a \rightarrow V_{2,2} = \{A\}$  → مجموعه متغیرهایی است که بطور بلا فصل a را می دهد :

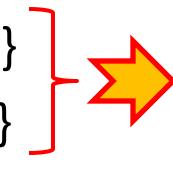
$w_{3,3} = b \rightarrow V_{3,3} = \{B\}$  → مجموعه متغیرهایی است که بطور بلا فصل b را می دهد :

$w_{4,4} = b \rightarrow V_{4,4} = \{B\}$  → مجموعه متغیرهایی است که بطور بلا فصل b را می دهد :

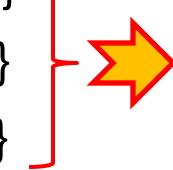
$w_{5,5} = b \rightarrow V_{5,5} = \{B\}$  → مجموعه متغیرهایی است که بطور بلا فصل b را می دهد :

# فصل ششم

:  $V_{1,2}, V_{2,3}, \dots, V_{n-1,n}$  ( محاسبه ۲ )

$$V_{1,2} = \{ A \mid A \rightarrow BC ; B \in V_{1,1}, C \in V_{2,2} \} \\ V_{1,1} = V_{2,2} = \{A\}$$


مجموعه حاوی تمامی متغیرهایی است  
که در سمت چپ قوانینی هستند که  
سمت راست آنها  $AA$  است. چون چنین  
موردی را نداریم، بنابراین :  $V_{12} = \emptyset$

$$V_{2,3} = \{ A \mid A \rightarrow BC ; B \in V_{2,2}, C \in V_{3,3} \} \\ V_{2,2} = \{A\} \\ V_{3,3} = \{B\}$$


مجموعه حاوی تمامی متغیرهایی است  
که در سمت چپ قوانینی هستند که  
سمت راست آنها  $AB$  است. بنابراین :  
 $V_{23} = \{S, B\}$

$$V_{3,4} = \{A\} ; V_{4,5} = \{A\}$$

$$V_{1,3} = \{S, B\} ; V_{2,4} = \{A\} ; V_{3,5} = \{S, B\}$$

$$V_{1,4} = \{A\} ; V_{2,5} = \{S, B\}$$

$$V_{1,5} = \{S, B\}$$

❖ نکته : دقیقا  $n(n+1)/2$  مجموعه از  $V_{ij}$  محاسبه شده اند.

# فصل ششم

## • تمرینات فصل ششم:

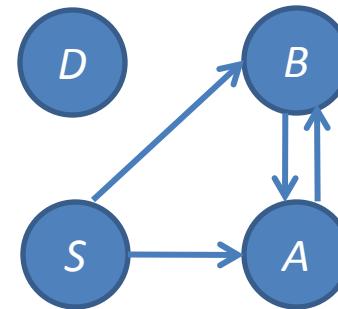
$$\left\{ \begin{array}{l} S \rightarrow aA|a|B|C \\ A \rightarrow aB|\lambda \\ B \rightarrow Aa \\ C \rightarrow cCD \\ D \rightarrow ddd \end{array} \right.$$

۱) قوانین بی فایده را از گرامر مقابل حذف نمایید؟

### گام اول

- $V_1 = \emptyset$
- i.  $P_1 = \{ S \rightarrow a, D \rightarrow ddd, A \rightarrow \lambda \}$
- $V_1 = \{S, D, A\}$
- ii.  $P_1 = P_1 \cup \{ S \rightarrow aA, B \rightarrow Aa \}$
- $V_1 = V_1 \cup \{B\}$
- iii.  $P_1 = P_1 \cup \{ A \rightarrow aB, S \rightarrow B \}$
- $V_1 = V_1$

### گام دوم



$$\left\{ \begin{array}{l} S \rightarrow aA|a|B \\ A \rightarrow aB|\lambda \\ B \rightarrow Aa \end{array} \right.$$

## فصل ششم

۲) قوانین  $\lambda$  و بی فایده را از گرامر مقابل حذف نمایید؟

$$\left\{ \begin{array}{l} S \rightarrow AaB | aaB \\ A \rightarrow \lambda \\ B \rightarrow bbA | \lambda \end{array} \right.$$

ابتدا به ترتیب قواعد  $B \rightarrow \lambda$  و  $A \rightarrow \lambda$  را حذف می کنیم. ✓

•  $A \rightarrow \lambda$  : پس از حذف قاعده  $S \rightarrow AaB | aaB | aB$

$$B \rightarrow bbA | bb | \lambda$$

•  $B \rightarrow \lambda$  : پس از حذف قاعده  $S \rightarrow AaB | aaB | Aa | aa$

$$B \rightarrow bbA$$

$$\left\{ \begin{array}{l} S \rightarrow AaB | aaB | aB | Aa | a | aa \\ B \rightarrow bbA | bb \end{array} \right.$$

سپس روال حذف قوانین بی فایده را فراخوانی می کنیم که حاصل بصورت زیر خواهد بود: ✓

$$\left\{ \begin{array}{l} S \rightarrow aaB | aB | aa | a \\ B \rightarrow bb \end{array} \right.$$

# فصل ششم

(۳) گرامر مقابل را به فرم نرمال چامسکی تبدیل نمایید؟

$$\begin{cases} S \rightarrow aSaA | A \\ A \rightarrow abA | b \end{cases}$$

ابتدا قاعده  $\lambda$  و قاعده یکه را حذف می کنیم. ✓

$$\begin{cases} S \rightarrow aSaA | abA | b \\ A \rightarrow abA | b \end{cases}$$

سپس گرامر را به فرم نرمال چامسکی تبدیل می نماییم. ✓

قواعد  $a \rightarrow A_1$  و  $b \rightarrow A_2$  به  $P$  اضافه می شوند و همچنین  $A_1, A_2$  هم به  $V$  اضافه می شوند. .1

$$\begin{cases} S \rightarrow A_1SA_1A | A_1A_2A | b \\ A \rightarrow A_1A_2A | b \\ A_1 \rightarrow a \\ A_2 \rightarrow b \end{cases}$$
.2

# فصل ششم

$$\begin{cases} S \rightarrow A_1A_3 | A_1A_5 | b \\ A_3 \rightarrow SA_4 \\ A_4 \rightarrow A_1A \\ A_5 \rightarrow A_2A \\ A \rightarrow A_1A_5 | b \\ A_1 \rightarrow a \\ A_2 \rightarrow b \end{cases}$$

.3

متغیر های  $A_3, A_4, A_5$  هم به  $V$  اضافه می شود. ✓

# فصل ششم

۴) گرامر مقابل را به فرم نرمال گریباخ تبدیل نمایید?

$$\left\{ \begin{array}{l} S \rightarrow ABb | a \\ A \rightarrow aaA | B \\ B \rightarrow bAb \end{array} \right.$$

ابتدا قاعده  $\lambda$  و قاعده یکه را حذف می کنیم. ✓

$$\left\{ \begin{array}{l} S \rightarrow ABb | a \\ A \rightarrow aaA | bAb \\ B \rightarrow bAb \end{array} \right.$$

سپس گرامر را به فرم نرمال گریباخ تبدیل می نماییم. ✓  
متغیر  $A$  را در قاعده  $S \rightarrow ABb$  جایگزین می کنیم و خواهیم داشت: .1

$$\left\{ \begin{array}{l} S \rightarrow aaABb | bAbBb | a \\ A \rightarrow aaA | bAb \\ B \rightarrow bAb \end{array} \right.$$

سپس آنرا به فرم نرمال گریباخ تبدیل می نماییم. .2

$$\left\{ \begin{array}{l} S \rightarrow aA_1ABA_2 | bAA_2BA_2 | a \\ A \rightarrow aA_1A | bAA_2 \\ B \rightarrow bAA_2 \\ A_1 \rightarrow a \\ A_2 \rightarrow b \end{array} \right.$$

# فصل هفتم

آتاماتای پشتہ ای

# فصل هفتم

## ▪ آتاماتای پشته ای پا (Push Down Automata) PDA

✓ آتامای متناهی (FA) نمی تواند بعلت محدودیت حافظه، کلیه زبانهای مستقل از متن را تشخیص دهد. لذا، باستی از ماشین دیگری برای تشخیص زبان های مستقل از متن بهره جست که همان PDA می باشد.

$$L_1 = \{ a^n b^n \mid n \geq 0 \} \rightarrow$$

نامحدود ، شمارش تعداد  $a$ ها ، کنترل اینکه  $a$ ها قبل از  $b$ ها باید  
حافظه نامحدود ← نیاز به ماشینی با شمارش نامحدود ←

$$L_2 = \{ ww^R \mid w \in (a,b)^* \} \rightarrow$$

کنترل ترتیب معکوس حروف

✓ نیاز به پشته (دارای ظرفیت نامحدود) ← آتاماتای پشته ای (PDA) ← تعريف یک خانواده جدید از زبان ها بنام زبان مستقل از متن معین یا (Deterministic Context Free Language). DCFL

# فصل هفتم

## ■ آتماتای پشته ای نامعین یا **(Nondeterministic PDA) NPDA**

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$$

$\Gamma$ : مجموعه متناهی از علائم الفبای پشته

$q_0 \in Q$ : وضعیت اولیه

$F$ : مجموعه وضعیت نهایی

$Q$ : مجموعه متناهی از وضعیتها

$\Sigma$ : الفبای ورودی

$Z$ : علامت شروع پشته

$$\delta : Q * (\Sigma \cup \{\lambda\}) * \Gamma \rightarrow Q * \Gamma^*$$

**مثال :**  $NDPA = (\{q_0, q_1, q_2, q_3\}, \{a,b\}, \{0,1\}, \delta, \{q_0\}, \{0\}, \{q_3\})$

$$\begin{cases} \delta(q_0, a, 0) = \{(q_1, 10), (q_3, \lambda)\} \\ \delta(q_0, \lambda, 0) = \{(q_3, \lambda)\} \end{cases} \quad \begin{cases} \delta(q_1, a, 1) = \{(q_1, 11)\} \\ \delta(q_1, b, 1) = \{(q_2, \lambda)\} \end{cases} \quad \begin{cases} \delta(q_2, b, 0) = \{(q_2, \lambda)\} \\ \delta(q_2, \lambda, 0) = \{(q_3, \lambda)\} \end{cases}$$

$$\begin{cases} \delta(q_1, a, 1) = \{(q_1, 11)\} \\ \delta(q_2, b, 1) = \{(q_2, \lambda)\} \end{cases}$$

این دو قدم باعث شمارش تعداد  $a$ ها و تطبیق آن با تعداد  $b$ ها می شود. واحد کنترل تا زمانیکه اولین  $b$  خوانده شود، در وضعیت  $q_1$  باقی می ماند و با اولین  $b$  به  $q_2$  می رود. این عمل باعث می گردد، هیچ  $b$  قبل از آخرین  $a$  نباشد.



$$L = \{ a^n b^n \mid n \geq 0 \} \cup \{a\}$$

# فصل هفتم

## ▪ توصیف بلافصل آتماتاتی پشته ای پا PDA :

✓ عوامل موثر : قسمت باقیمانده از رشته ورودی، محتوای فعلی پشته  $\leftarrow$  بطوریکه  $q$  وضعیت واحد کنترل،  $w$  باقیمانده رشته ورودی، و  $u$  محتوای پشته می باشند.

$(q_2,,y) \in \delta(q_1,a,b)$  ممکن است، اگر و فقط اگر  $(q_1,aw,bx) \vdash (q_2,w,yx)$  ✓

✓  $\vdash^*$  به معنای حرکت در چندین گام توسط واحد کنترل و همچنین،  $M$  به معنای حرکت توسط  $M$  گام می باشد.

✓ اگر  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$  یک آتماتاتی پشته ای نامعین باشد، زبان پذیرفته شده توسط ماشین  $M$ ، عبارت است از مجموعه  $L(M)$

$$L(M) = \{W \in \Sigma^*: (q_0, W, Z) \vdash^* M(p, \lambda, u), p \in F, u \in \Gamma^*\}$$

# فصل هفتم

مثال : یک NPDA برای زبان  $L$  طراحی نمایید؟

$$M = (\{q_0, q_f\}, \{a, b\} \cup \{0, 1, 2\}, \delta, q_0, Z, \{q_f\})$$

$$\begin{array}{lll} \delta(q_0, \lambda, z) = \{(q_f, z)\} & \delta(q_0, a, 0) = \{(q_f, 00)\} & \delta(q_0, a, 1) = \{(q_0, \lambda)\} \\ \delta(q_0, a, z) = \{(q_0, 0z)\} & \delta(q_0, b, 0) = \{(q_0, \lambda)\} & \delta(q_0, b, 1) = \{(q_0, 11)\} \\ \delta(q_0, \lambda, z) = \{(q_f, z)\} & & \end{array}$$

رشته  $baab(q_0, baab, z) \vdash (q_0, aab, 1z) \vdash (q_0, ab, z) \vdash (q_0, b, 0z) \vdash (q_0, \lambda, z)$   
 $\vdash (q_{0f}, \lambda, z)$

# فصل هفتم

$$L = \{WW^R : W \in \{a,b\}^+\}$$

**مثال :** یک NPDA برای زبان  $L$  طراحی نمایید؟

$$M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, z\}, \delta, q_0, z, \{q_2\})$$

قرار دادن $W$ در پشته	حدس زدن وسط رشته	طبیق $W^R$
$\delta(q_0, a, a) = \{(q_0, aa)\}$	$\delta(q_0, \lambda, a) = \{(q_1, a)\}$	$\delta(q_1, a, a) = \{(q_1, \lambda)\}$
$\delta(q_0, b, a) = \{(q_0, ba)\}$	$\delta(q_0, \lambda, b) = \{(q_1, b)\}$	$\delta(q_1, b, b) = \{(q_1, \lambda)\}$
$\delta(q_0, a, b) = \{(q_0, ab)\}$		$\delta(q_1, \lambda, z) = \{(q_2, z)\}$
$\delta(q_0, b, b) = \{(q_0, bb)\}$		
$\delta(q_0, a, z) = \{(q_0, az)\}$		
$\delta(q_0, b, z) = \{(q_0, bz)\}$		

رشته  $abba((q_0, abba, z) \vdash (q_0, bba, az) \vdash (q_0, ba, baz) \vdash (q_1, a, az) \vdash (q_1, \lambda, z) \vdash (q_2, z))$

# فصل هفتم

## ■ آتاماتای پشته ای (PDA) و زبان های مستقل از متن :

- برای هر زبان مستقل از متن، یک  $NPDA$  وجود دارد که آن زبان را می پذیرد و بالعکس. هر زبانی که توسط یک  $NPDA$  پذیرفته می شود، مستقل از متن می باشد. ✓
- بنابراین، می توان یک  $NPDA$  طراحی نمود که قادر باشد یک اشتقاق چپ، برای کلیه رشته های زبان را ایجاد نماید. ✓
- فرض می کنیم که زبان، توسط یک گرامر نرمال گریباخ، تولید شده باشد. ✓

**مثال :** یک  $PDA$  برای پذیرش زبانی با گرامر  $S \rightarrow aSbb \mid a$  طراحی نمایید؟

$$\xrightarrow{\text{نرمال گریباخ نیست}} \left\{ \begin{array}{l} S \rightarrow aSA \mid a \\ A \rightarrow bB \\ B \rightarrow b \end{array} \right.$$

$M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, \{q_0\}, \{q_2\})$

# فصل هفتم

$$\delta(q_0, \lambda, z) = \{(q_1, Sz)\}$$

۱. ابتدا، از وضعیت شروع آغاز می نماییم.

۲. با حذف  $S$  از پشته و جایگزینی آن با  $SA$  توسط  $PDA$  ، در حالیکه ورودی  $a$  است، بایستی  
 $S \rightarrow aSA \mid a$  باعث خوانده شدن  $a$  از ورودی و حذف  $S$  از پشته گردد.

$$\delta(q_1, b, A) = \{(q_1, B)\}$$

و

$$\delta(q_1, b, B) = \{(q_1, \lambda)\}$$

۳. داریم :

۴. ظاهر شدن علامت شروع پشته، به معنای پایان اشتقاق بوده و  $PDA$  در وضعیت نهایی قرار می گیرد.

$$\delta(q_1, \lambda, z) = \{(q_2, \lambda)\}$$

• **قضیه :** به ازای هر زبان مستقل از متن  $L$ ، یک  $NDPA$  بنام  $M$  وجود دارد بطوریکه  $L = L(M)$

# فصل هفتم

$$\begin{cases} S \rightarrow aA \\ A \rightarrow aABC \mid bB \mid a \\ B \rightarrow b \\ C \rightarrow c \end{cases}$$

**مثال :** ساختار قضیه مذکور

گرامر موجود در فرم نرمال گریباخ می باشد، لذا یک NPDA برای  $M$  وجود دارد. ✓

$$\begin{cases} \delta(q_0, \lambda, z) = \{(q_1, Sz)\} \\ \delta(q_1, \lambda, z) = \{(q_f, z)\} \end{cases}$$

$$\begin{cases} \delta(q_1, a, S) = \{(q_1, A)\} \\ \delta(q_1, a, A) = \{(q_1, ABC), (q_1, \lambda)\} \\ \delta(q_1, b, A) = \{(q_1, B)\} \\ \delta(q_1, b, B) = \{(q_1, \lambda)\} \\ \delta(q_1, c, C) = \{(q_1, \lambda)\} \end{cases}$$

پردازش رشته □

$$(q_1, aaabc, z) \vdash (q_1, aaabc, Sz) \vdash (q_1, aaabc, Az) \vdash (q_1, abc, ABCz) \vdash (q_1, bc, BCz) \vdash (q_1, c, Cz) \vdash (q_1, \lambda, z) \vdash (q_f, \lambda, z)$$

$$S \rightarrow aA \rightarrow aaABC \rightarrow aaaBC \rightarrow aaabC \rightarrow aaabc$$

اشتقاق رشته □

ضروری نیست که گرامر در فرم نرمال گریباخ باشد. به عنوان مثال، برای قوانین زیر: ✓

I.  $A \rightarrow Bx$  را از پشته برداشته و بدون مصرف ورودی، آن را با  $Bx$  جایگزین می نماییم.

II. ابتدا، رشته  $ab$  در ورودی را با رشته مشابه در پشته، تطبیق داده و سپس،  $A$  را با  $Cx$  جایگزین می نماییم.

$$A \rightarrow abCx$$

# فصل هفتم

## ■ آتاماتای پشته ای معین (DPDA) و زبان های مستقل از متن :

آتاماتای پشته ای معین  $DPDA$  ، به صورت  $M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$  ، مطابق تعریف می گردد. بعلاوه :

$$\forall q \in Q, a, c \in \Sigma \cup \{\lambda\}, b \in \Gamma \rightarrow \begin{cases} 1) \delta(q, a, b) \\ 2) \text{اگر } \delta(q, \lambda, b) \neq \phi \Rightarrow \delta(q, c, b) = \phi \end{cases}$$

به ازای هر ورودی و هر چه در بالای پشته باشد، فقط یک حرکت می تواند انجام گردد.  
حداکثر دارای یک عنصر باشد.

هرگاه یک حرکت  $\lambda$  امکان پذیر گردد، آنگاه برای آن پیکربندی، هیچ حرکت همراه با مصرف ورودی، امکان پذیر نخواهد بود.

تنها شرطی که برای معین بودن وجود دارد، این است که در همه حال فقط یک حرکت امکان پذیر می باشد.

# فصل هفتم

## ▪ زبان مستقل از متن معین یا : DCFL

✓ زبان  $L$  را مستقل از متن معین و یا DCFL گویند، اگر و فقط اگر یک DPDA مانند  $M$  وجود داشته باشد، بطوریکه داشته باشیم .  $L = L(M)$

**مثال :** زبان  $L = \{a^n b^n : n \geq 0\}$  مستقل از متن معین می باشد، چرا که یک DPDA مانند  $M$  به شرح زیر وجود دارد :

$$M = (\{q_0, q_1, q_2\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_0\})$$

• به ازای هر  $a$ ,  $1$  را در پشته Push کرده و با هر  $b$ ,  $1$  را از پشته Pop می کند.

$$\delta(q_0, a, 0) = \{(q_1, 10)\}$$

$$\delta(q_1, a, 1) = \{(q_1, 11)\}$$

$$\delta(q_1, b, 1) = \{(q_2, \lambda)\}$$

$$\delta(q_2, b, 1) = \{(q_2, \lambda)\}$$

$$\delta(q_2, \lambda, 0) = \{(q_0, \lambda)\}$$

**مثال :** کدامیک از زبان های زیر، مستقل از متن می باشند؟

$$L_1 = \{a^n b^n : n \geq 0\}$$

مستقل از متن معین

$$L_2 = \{a^n b^{2n} : n \geq 0\}$$

مستقل از متن



مستقل از متن معین :  $L_1 \cup L_2$

# فصل هفتم

## ■ گرامر های مربوط به زبان های مستقل از متن معین :

- ✓ در گرامر های ساده، در هر مرحله از پویش، مشخص می باشد که کدامیک از قوانین بایستی استفاده گردند. (گرامرهای منظم چپ یا LLG)

**مثال :** گرامر  $S \rightarrow aSb \mid ab$  می باشد.

- برای اینکه قادر باشیم که تعیین نماییم، کدام قانون از بین قوانین  $S \rightarrow aSb$  و یا  $S \rightarrow ab$ ، بایستی استفاده گردد، دو حرف آلفابت پشت سر هم در رشته ورودی را خوانده و در نظر می گیریم. اگر اولین آلفابت، یک حرف  $a$  و دومین آلفابت، یک حرف  $b$  باشد، بایستی از قانون  $S \rightarrow ab$  استفاده نماییم، و در غیر اینصورت از قانون  $S \rightarrow aSb$ .

- ✓ می گوییم گرامر  $LL(K)$  است، اگر قادر باشیم با در دست داشتن آلفابت فعلی ورودی و همچنین خواندن و در نظر گرفتن  $K-1$  آلفابت بعدی، قانونی را که بایستی استفاده گردد، تشخیص دهیم. گرامر موجود در مثال فوق،  $LL(2)$  می باشد.

# فصل هفتم

**مثال :** گرامر  $S \rightarrow SS | aSb | ab$

گرامر فوق، برای هیچ مقداری از  $K$ ، یک گرامر  $LL(K)$  نمی باشد. به عنوان نمونه، دو قانون  $S \rightarrow SS$  و  $S \rightarrow aSb$  را در نظر بگیرید. آلفابت فعلی ورودی قادر نیست که قانون صحیح را برای پردازش ورودی تشخیص دهد. اگر دو آلفابت ورودی با همدیگر خوانده شوند و همچنین، اگر دو آلفابت ورودی  $aa$  باشند، باز هم نمی توان تشخیص داد که کدام قانون صحیح بعدی برای پردازش ورودی خواهد بود.

مشکل اینجاست که نمی توان پیشگویی نمود که چند بار تکرار الگوی  $a^n b^n$  تا انتهای رشته مورد نیاز خواهد بود. بنابراین، باقیتی گرامر فوق را به صورت زیر باز نویسی نمود. توجه داشته باشید که این گرامر  $LL$  می باشد.

$$S \rightarrow aSbS | \lambda$$

به عنوان نمونه، پردازش رشته  $abab$  به شرح زیر می باشد.

$$S \rightarrow aSbS \rightarrow abS \rightarrow abaSbS \rightarrow ababS \rightarrow abab$$

مشاهده می گردد که این گرامر نیاز به انتخاب ندارد، بطوریکه اگر آلفابت ورودی فعلی حرف  $a$  باشد، از دو قاعده گرامر، قاعده  $S \rightarrow aSbS$  استفاده می گردد و همچنین، اگر آلفابت ورودی فعلی حرف  $b$  باشد، یا در انتهای رشته قرار داشته باشد، باقیتی قاعده  $\lambda \rightarrow S$  استفاده گردد. لذا، گرامر جدید قادر خواهد بود که رشته تهی را نیز تولید نماید. بنابراین خواهیم داشت :

$$\begin{cases} S_1 \rightarrow aSbS \\ S \rightarrow aSbS | \lambda \end{cases}$$

# فصل هفتم

## ▪ گرامر $LL(K)$

فرض نمایید که  $G = (V, T, S, P)$  ، یک گرامر مستقل از متن می باشد، اگر به ازای هر

زوج اشتقاق چپ :

$$\begin{cases} S \xrightarrow{*} W_1 A X_1 \Rightarrow W_1 y_1 X_1 \xrightarrow{*} W_1 W_2 \\ S \xrightarrow{*} W_1 A X_2 \Rightarrow W_1 y_2 X_2 \xrightarrow{*} W_1 W_3 \end{cases} \quad W_1, W_2, W_3 \in T^*$$

اگر تساوی  $K$  آلفابت ورودی چپ در رشته  $W_3 W_2$  ، به معنای  $y_1 = y_2$  باشد، آنگاه گرامر  $G$  یک گرامر  $LL(K)$  خواهد بود.

اگر در هر مرحله از اشتقاق چپ  $(W_1 A X)$  ، آلفابت بعدی ورودی مشخص باشد، گام بعدی در اشتقاق دقیقاً معین می شود.

# فصل هشتم

ماشین تورینگ

پایان