

جزوه درس

نظريه زبانها و ماشينها

نام استاد : علی اصغر پورحاجی کاظم

منال:

$$G1: L(G1) = \{a^k b^k c^k | k \geq 1\}$$

$$G1: L(G2) = \{a^k b^k | k \geq 1\}$$

$$\begin{aligned} \Sigma &\rightarrow A \\ A &\rightarrow aABC \\ A &\rightarrow abc \\ CB &\rightarrow BC \\ bB &\rightarrow bb \\ bC &\rightarrow bc \\ cC &\rightarrow cc \end{aligned}$$

$$\begin{aligned} \Sigma &\rightarrow A \\ A &\rightarrow aABC \\ A &\rightarrow abc \\ CB &\rightarrow BC \\ bB &\rightarrow bb \\ bC &\rightarrow bc \end{aligned}$$

$$G3: L(G3) = \{a^k b^k | k \geq 1\}$$

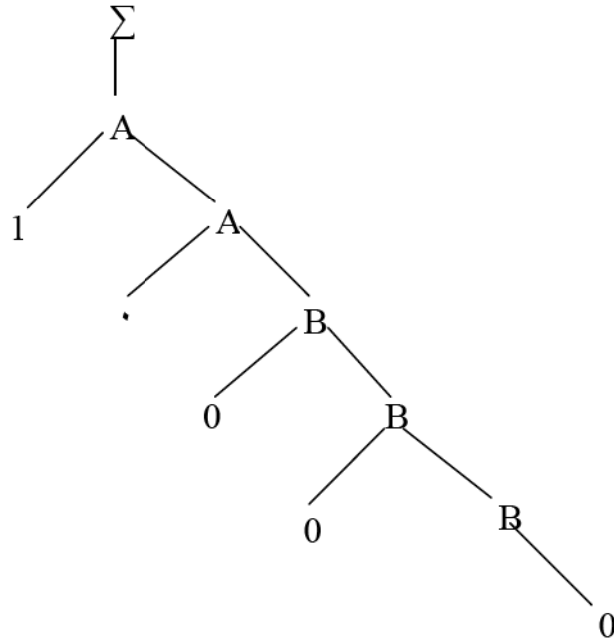
$$\begin{aligned} \Sigma &\rightarrow S \\ S &\rightarrow aSb \\ S &\rightarrow ab \end{aligned}$$

	Type	Protection	
Contracting (منقبض شونده)	0	$\phi A \psi \rightarrow \phi \omega \psi$	گرامر بدون محدودیت
Non-Contract (غیر منقبض شونده)	1	$\phi A \psi \rightarrow \phi \omega \psi, \omega \neq \lambda$ $\Sigma \rightarrow \lambda$	گرامرهای Content Sensitive
	2	$A \rightarrow \omega, \omega \neq \lambda$ $\Sigma \rightarrow \lambda$	گرامرهای Content Free
	3	$\left. \begin{array}{l} A \rightarrow aB \\ A \rightarrow a \\ \Sigma \rightarrow \lambda \end{array} \right\} \text{right linear}$ <p style="text-align: center;">OR</p> $\left. \begin{array}{l} A \rightarrow Ba \\ A \rightarrow a \\ \Sigma \rightarrow \lambda \end{array} \right\} \text{left linear}$	Regular Grammers

درخت اشتقاق (Derivation Tree):

$\Sigma \rightarrow A$
 $A \rightarrow 1A$
 $A \rightarrow 0B$
 $B \rightarrow 0B$
 $B \rightarrow 0$

10000

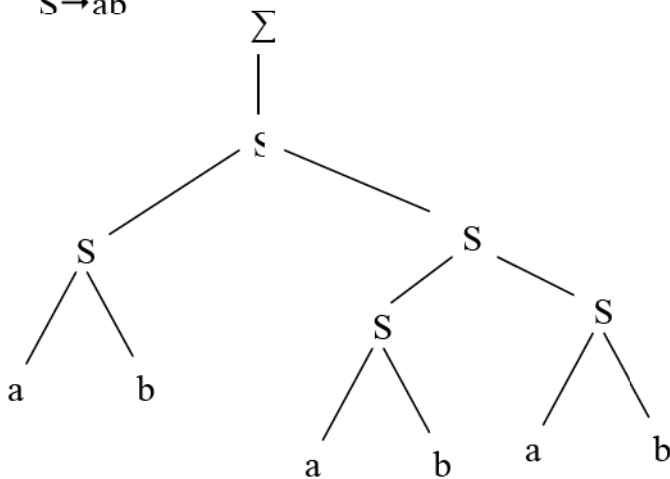


ابهام (Ambiguity):

فرض کنیم G یک گرامر content free باشد و فرض کنیم ω جمله ای در (G) باشد L باشد آنگاه ω مبهم است اگر اشتقاقی از ω وجود داشته باشد که مربوط به درختهای متفاوتی باشند.

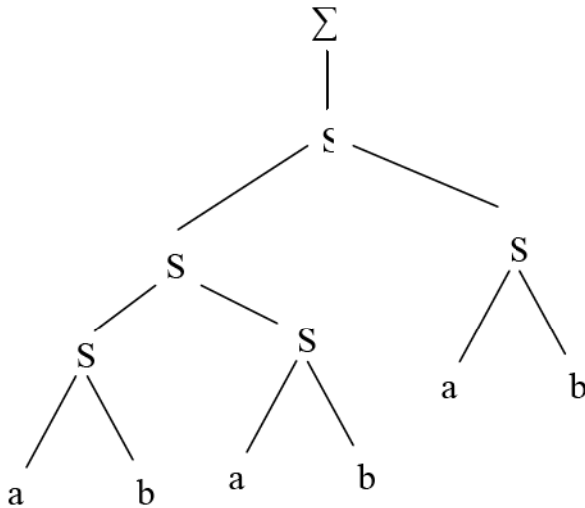
$\Sigma \rightarrow S$
 $S \rightarrow SS$
 $S \rightarrow ab$

ababab



1)

2)



یک اشتقاق leftmost است اگر و فقط اگر چپ ترین سمبل غیر پایانی در ω_i برای حصول به ω_{i+1} جایگزین گردد.

$$\omega_0 \Rightarrow \omega_1 \Rightarrow \dots \Rightarrow \omega_n$$

$$\omega_i = \alpha A \beta \quad \alpha \in T^*$$

$$\omega_{i+1} = \alpha \omega \beta \quad \begin{matrix} A \in N \\ \beta \in (N \cup T)^* \end{matrix}$$

$A \rightarrow \omega$ در صورتیکه قاعده تولیدی بفرم داشته باشیم

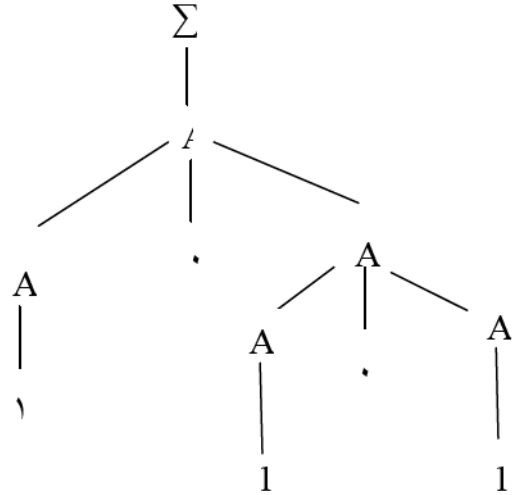
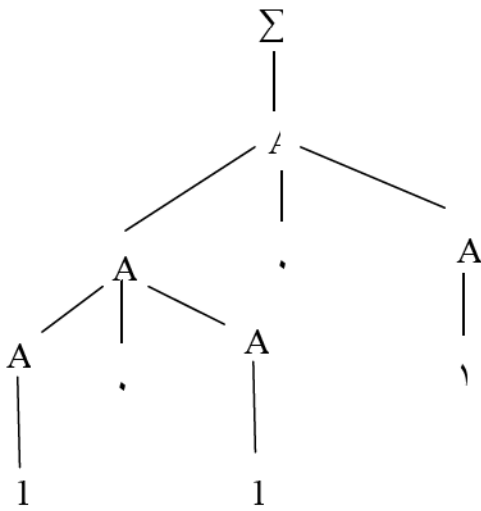
تعریف: یک گرامر Content free مبهم است اگر فقط جملاتی بوسیله دو یا چند اشتراک Left most تولید کند.

مثال:

$$\begin{matrix} \Sigma \rightarrow A \\ A \rightarrow A0A \\ A \rightarrow 1 \end{matrix} \quad 10101$$

$$\Sigma \Rightarrow A \Rightarrow A0A \Rightarrow A0A0A \Rightarrow 10A0A \Rightarrow 1010A \Rightarrow 10101$$

$$\Sigma \Rightarrow A \Rightarrow A0A \Rightarrow 10A \Rightarrow 10A0A \Rightarrow 1010A \Rightarrow 10101$$



G: $\Sigma \rightarrow A$

100

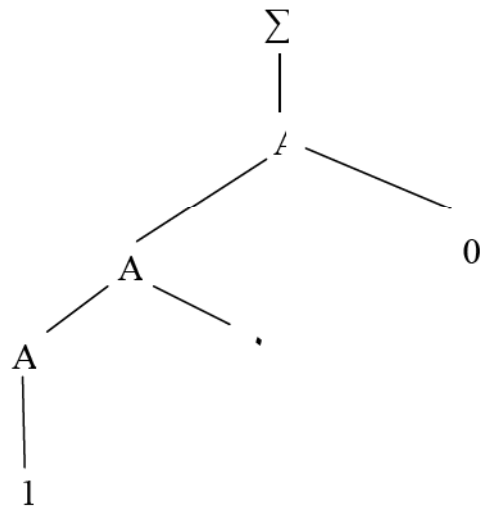
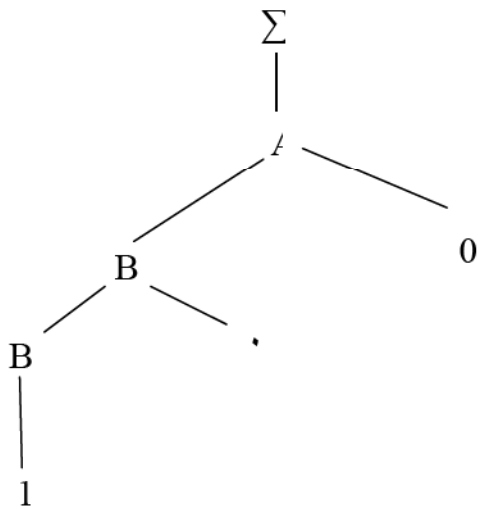
$A \rightarrow B0$

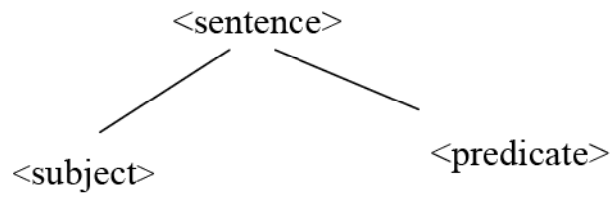
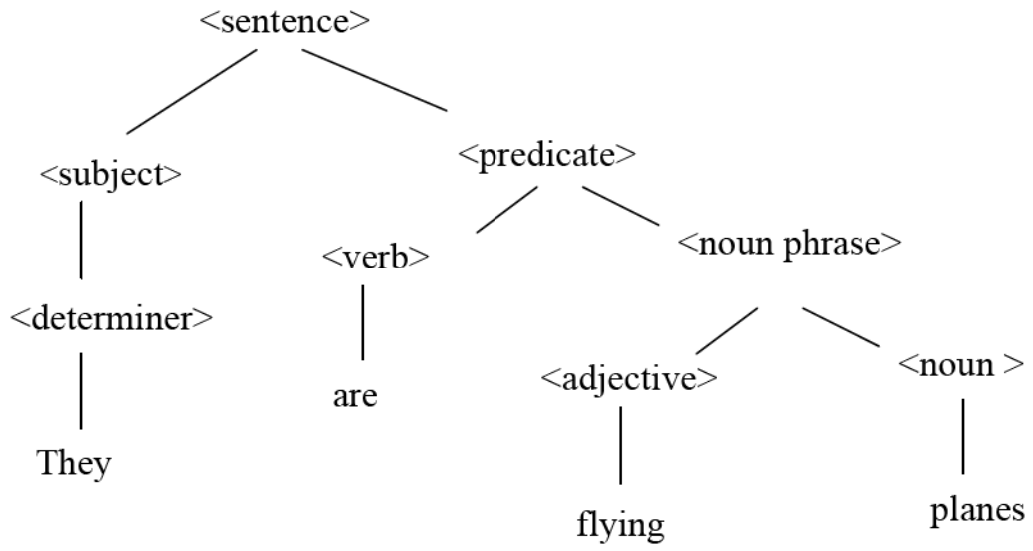
$A \rightarrow A0$

$B \rightarrow B0$

$A \rightarrow 1$

$B \rightarrow 1$





G: $A \rightarrow ABA$

$A \rightarrow a$

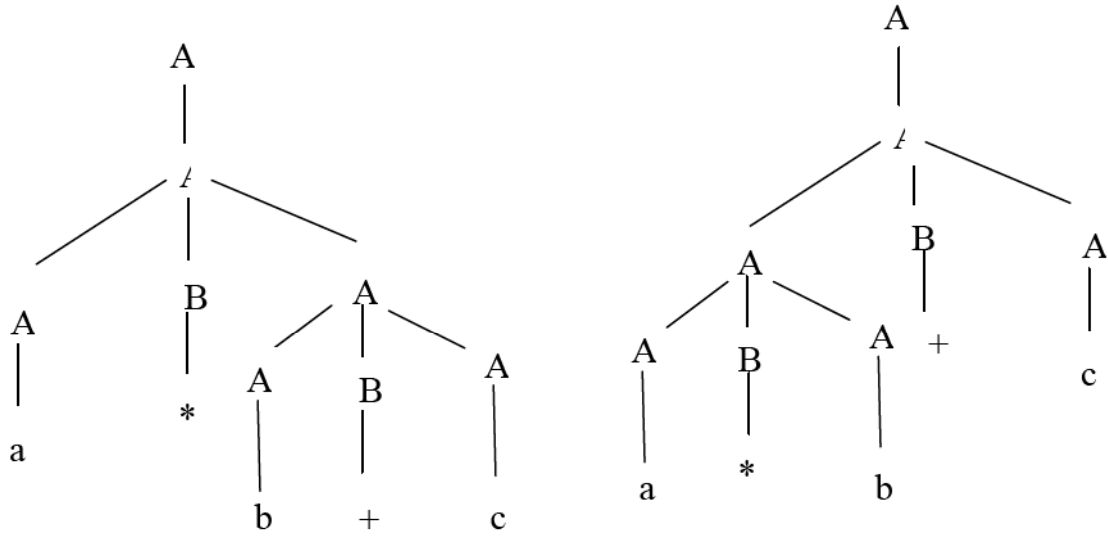
$A \rightarrow b$

$A \rightarrow C$

$B \rightarrow +$

$B \rightarrow *$

$a*b+c$



مسائل فصل سوم: 3.1, 3.2, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9

فصل چهارم

Finite State Machine (ماشین حالت محدود):

- ❖ Finiteness
- ❖ Discreteness
- ❖ (deterministic) Sequential Action



N part

 $q^i(t)$ حالت part i ام در لحظه t باشدحالت کلی ماشینی M در لحظه t هر part k , وضعیت دارد

$$q(t) = (q^{(1)}(t), q^{(2)}(t), \dots, q^{(n)}(t))$$

ماشینی M , k^n حالت می تواند داشته باشد. q_i وضعیت اولیه ماشینی را نشان می دهد. Q در لحظه صفر $q(0) = q_i$

$$q(0), q(1), \dots, q(n)$$

$$Q \in q(t)$$

تابع انتقال حالت $q(t+1) = f(q(t), s(t+1))$

$$F: Q \times S \rightarrow Q$$

تابع خروجی $r(t+1) = g(q(t), s(t+1))$

$$g: Q \times S \rightarrow R$$

ویژگیهای یک ماشین حالت محدود:

۱- رفتار M در زمانهای $t=0,1,2,\dots$ تعریف می شود

۲- سمبل $s(t)$ از مجموعه ای به نام مجموعه سمبل های ورودی یعنی S انتخاب میشود

۳- سمبل های خروجی $r(t)$ از مجموعه ای محدود به نام مجموعه سمبل های خروجی یعنی R انتخاب

می شوند

۴- رفتار M برحسب دنباله سمبل های ورودی تعیین می شود

۵- رفتار M بصورت دنباله ای از حالات که هر یک عضوی از Q است نشان داده میشود

۶- یک حالت ابتدایی از M وجود دارد (q_i) که تشریح کننده وضعیت ابتدایی $part$ های ماشینی M قبل از

ورود تحریکی به آن است.

Element های ریاضی یک ماشین محدود:

۱-مجموعه های محدود S, R, Q

۲-تابع انتقال f که تعریف کننده حالت بعدی بر حسب حالت فعلی و سمبلهای ورودی بعدی است.

۳- تابع خروجی g که تعریف کننده سمبل خروجی بر حسب حالتهاى فعلی و سمبل ورودی بعدی است.

۴- $q_i \in Q$

ماشینهای Transition Assigned output

خروجی بر اساس انتحالات ماشینی حاصل میشود.

تعریف:

یک Transition Assigned Output یک شش تائی:

$$M = (Q, S, R, F, G, QI)$$

Q: مجموعه حالات M

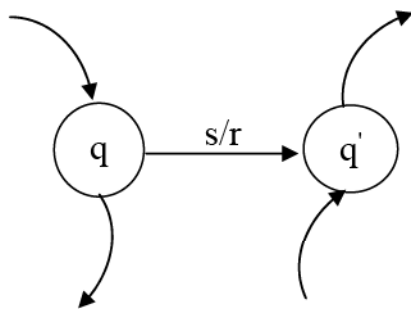
S: مجموعه سمبلهای ورودی

R: مجموعه سمبلهای خروجی

f: تابع انتقال $f: Q \times S \rightarrow Q$

g: تابع خروجی $g: Q \times S \rightarrow R$

State Diagram

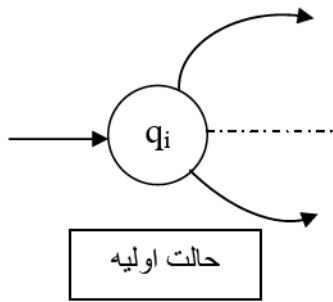


$$q' = f(q, s)$$

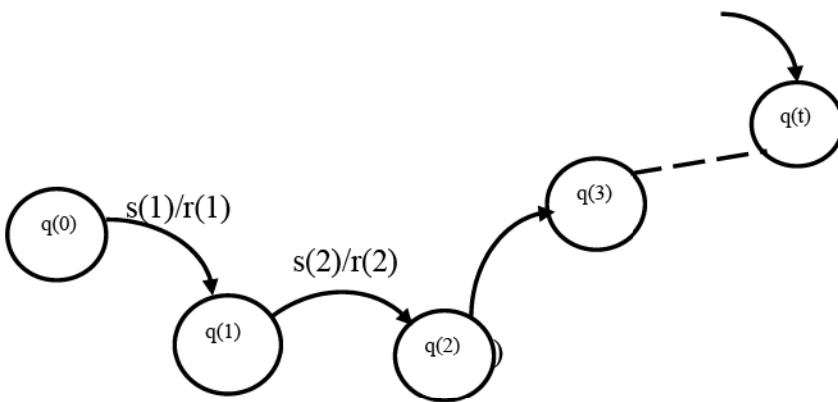
$$r = g(q, s)$$

State Table

$q_i \rightarrow 1$.	.	S		
۲					
.					
.		
q			q', r		
.		



رفتار ماشین: با دنباله ای از اتصالات نشان می دهیم.



$$q(0) \xrightarrow{\omega / \varphi} q(t)$$

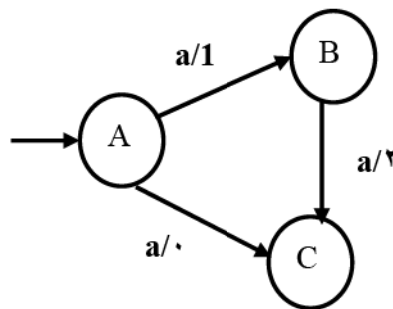
$$\omega = s(1)s(2)\dots s(t)$$

$$\varphi = r(1)r(2)\dots r(t)$$

مثال: ماشین که MOD 3 تعداد سمبلهای ورودی را به ما بدهد.

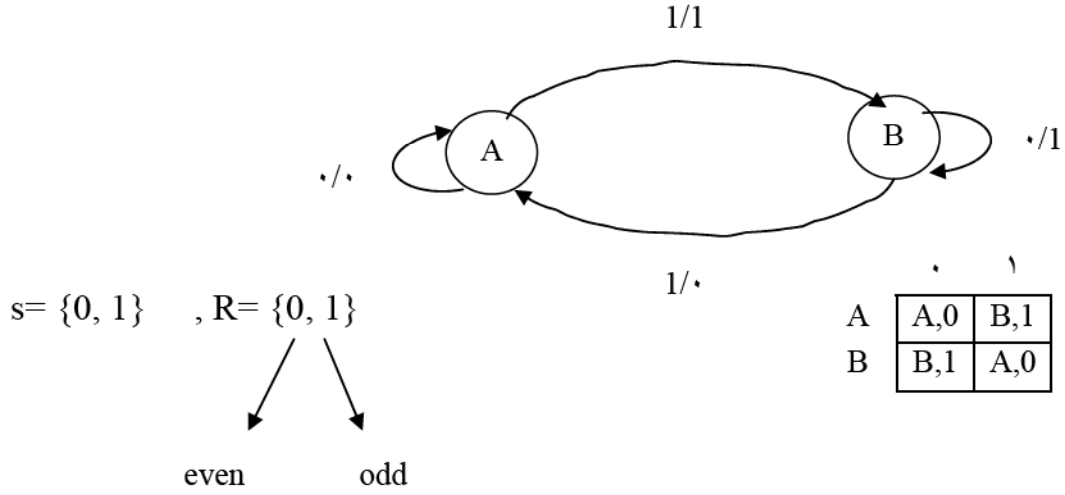
q(t) A means $|\omega| \bmod 3 = 0$
 B means $|\omega| \bmod 3 = 1$
 C means $|\omega| \bmod 3 = 2$

$s = \{a\}$
 $Q = \{A, B, C\}$
 $R = \{0, 1, 2\}$



	a
A	B/1
B	C/2
C	A/0

Parity checker



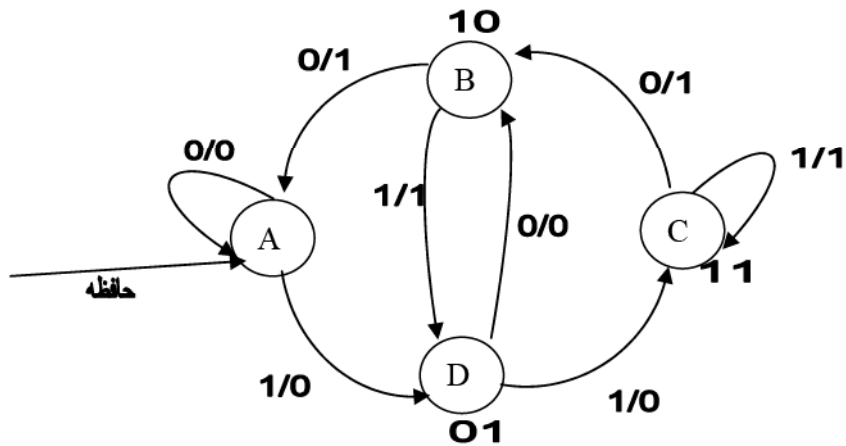
مثال ۳: Two unit delay

$$S(t) \quad r(t+s) = s(t)$$

$$S=R = \{0, 1\}$$

$$r(1) = 0, r(2) = 0$$

S(t-1)	S(t)	State
0	0	A
0	1	B
1	1	C
1	0	D



$$\omega = 0010110$$

$$\varphi = 0000101$$

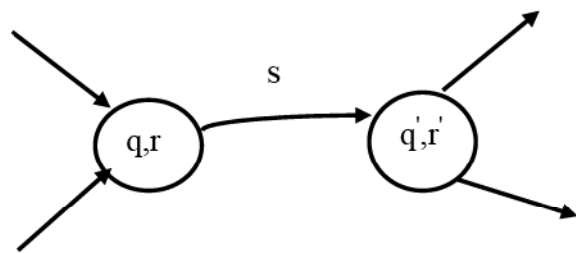
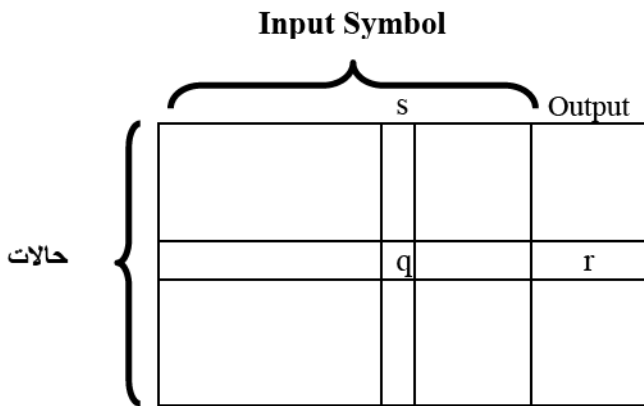
ماشینهای State Assigned Output :

تعریف: یک finite state machine State assigned output شش تائی بفرم

$$M=(Q,S,R,F,H,q_i) \text{ است که:}$$

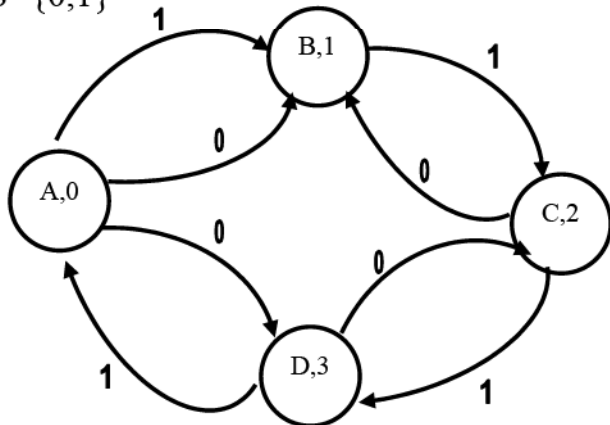
$$F: Q \times S \rightarrow Q \text{ تابع انتقال حالت}$$

$$H: Q \rightarrow R \text{ تابع خروجی}$$



مثال: می خواهیم یک شمارنده بالا به پایین پیمانه \sum بسازیم

$$S=\{0,1\}$$



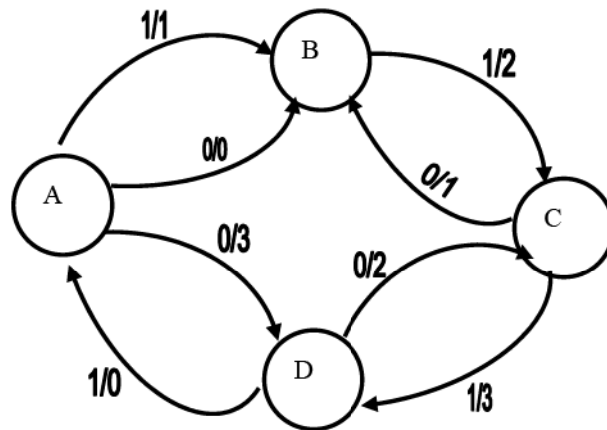
$$\omega = s(1) s(I) \dots s(I)$$

$$N_0(\omega) = \omega \text{ تعداد صفرهای}$$

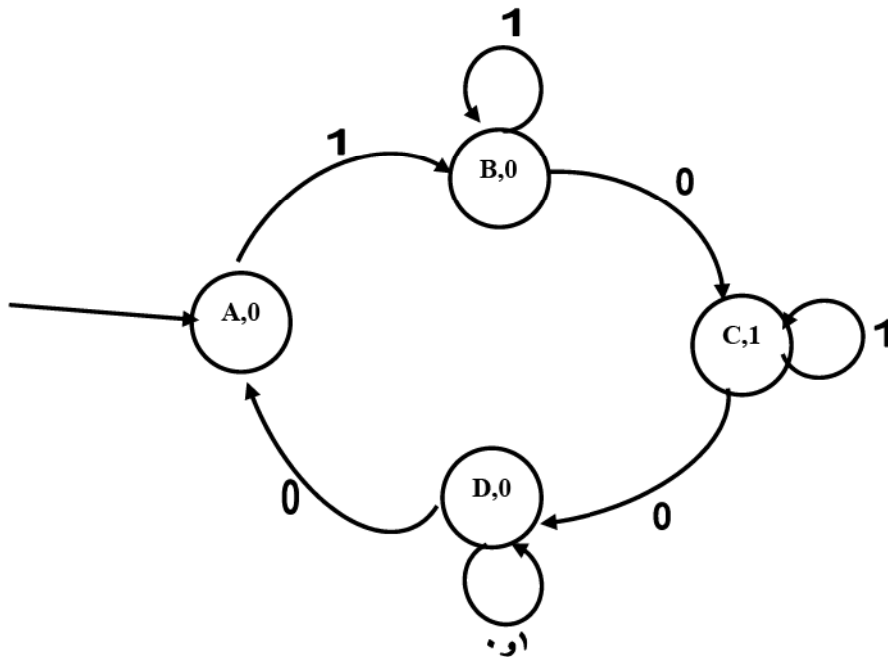
$$N_1(\omega) = \omega \text{ تعداد یکهای}$$

$$r(t) = [N_1(\omega) - N_0(\omega)] \text{ mod } 4$$

$$R = \{0, 1, 2, 3\}$$



مثال: تشخیص دهنده زبان (Language Recognizer):



پیچیدگی ماشین (machine complexity):

$$q = (q^{(1)}, q^{(2)}, \dots, q^{(n)})$$

n قطعه و هر قطعه دو حالت

تعداد کل حالات ماشینی 2^n خواهد بود.

(تعداد حالات ماشینی $\approx \log_2$ تعداد قطعات)

بیتی 2^{15} کلمه 2^{32} بیتی

$$\text{تعداد حالات حافظه} = 2^2 = 10^{300000}$$

دنباله های حالات (state sequences):

تعریف: فرض کنیم M یک ماشین محدود با تابع انتقال $F: Q \times S = Q$ باشد اگر $f(q, s) = q'$ گوئیم حالت

s -successor حالت q است می نویسیم:

$$q \xrightarrow{s} q'$$

اگر رشته ای از سمبلهای ورودی $\omega = s(1)s(2)\dots s(+)$ را از حالت $q = q(0)$ به حالت $q' = q(+)$ ببرد:

$$q(0) \xrightarrow{s(1)} q(1) \xrightarrow{s(2)} q(2) \xrightarrow{s(3)} \dots \xrightarrow{s(t)} q(t)$$

گوئیم حالت q' ، ω -successor حالت q است و می نویسیم:

$$q \xrightarrow{\omega} q'$$

$q(0), q(1), \dots, q(+)$ را یک دنباله حالات admissible گویند.

با این تعریف می توان دامنه f را به صورت زیر بسط دهیم:

$$f: Q \times S \longrightarrow Q \text{ قبلا:}$$

$$f: Q \times S^* \longrightarrow Q \text{ فعلی:}$$

$$q \xrightarrow{\omega} q' \text{ منظور } f(q, \omega) = q'$$

$$F(q, \lambda) = q \text{ برای هر } q \in Q$$

تعریف: فرض کنیم M یک ماشین حالت محدود بوده و فرض کنیم $q_i = q(0), q(1), \dots, q(t)$ یک دنباله حالات کمکی برای $\omega = s(1) \dots s(t-1)s(t)$ باشد:

الف- اگر M یک ماشین Transition Assigned با تابع خروجی $g : Q \times S \rightarrow R$ باشد آنگاه

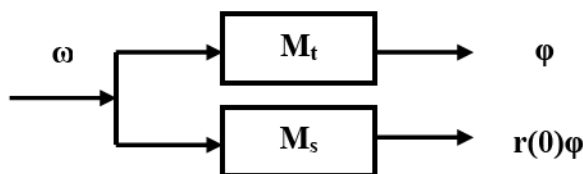
$$r(i) = g(q(i-1), s(i)) \text{ در آن } \omega \text{ نامیده میشود که در آن } \varphi = r(1)r(2) \dots r(t)$$

ب- اگر M یک ماشین State-Assigned با تابع خروجی $h : Q \times S \rightarrow R$ باشد سپس

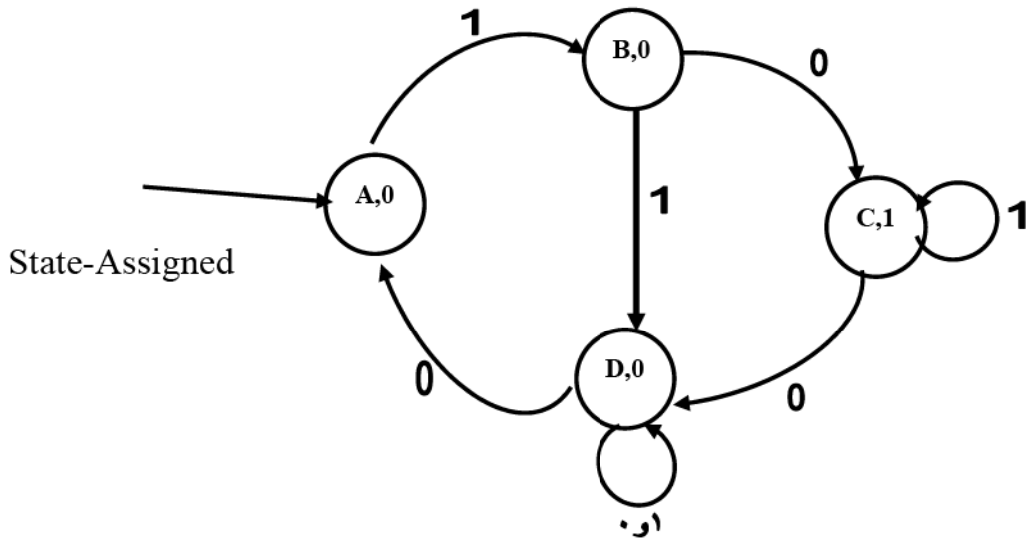
$$r(i) = h(q(i)) \text{ در آن } \omega \text{ نامیده میشود که در آن } \varphi = r(1)r(2) \dots r(t)$$

تبدیل ماشینهای State-assigned و Transition Assigned :

تعریف: یک ماشین Transition Assigned مثل M_t و یک ماشین state-assigned مثل M_s مشابه هستند اگر برای هر تحریک پاسخ M_s دقیقاً معادل M_t که با یک سمبل اضافی دلخواه ولی ثابت شروع میشود.



قضیه: برای هر ماشین state-assigned مثل M_s یک ماشین مشابه Transition Assigned مثل M_t وجود دارد و برعکس.



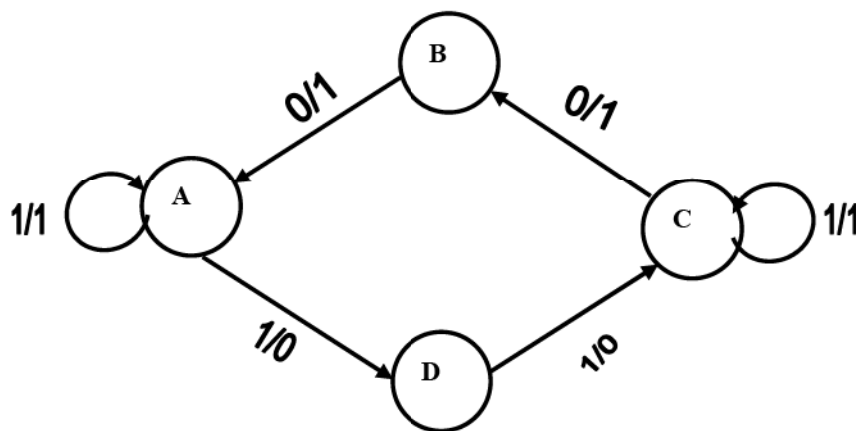
۱- تشکیل ماشین M_t در M_s خروجی r را مشخص کند، هر انتقال در M_t به q را با r بر حسب گذاری می کنیم.

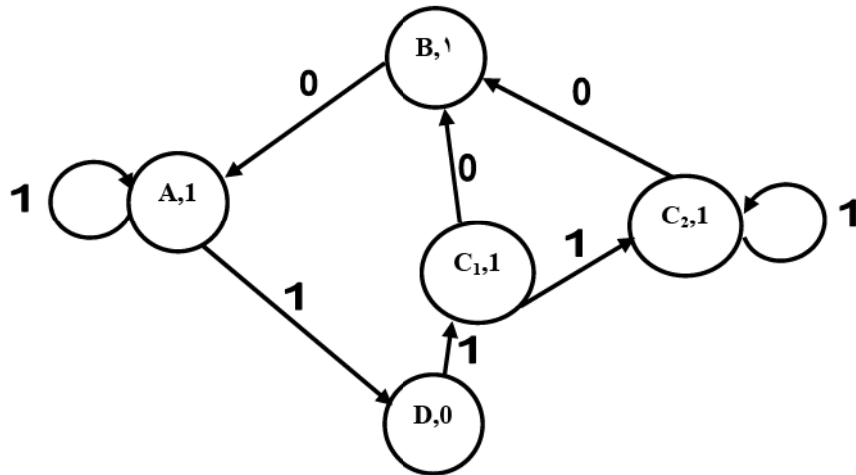
اگر $M_s = (Q, S, R, f, h, q_i)$ آنگاه $M_t = (Q, S, R, f, h, q_i)$

$$g(q, s) = h(f(q, s))$$

$$g(A, 1) = h(f(A, 1)) = h(B)$$

۲- تشکیل ماشین M_s از روی ماشین M_t :



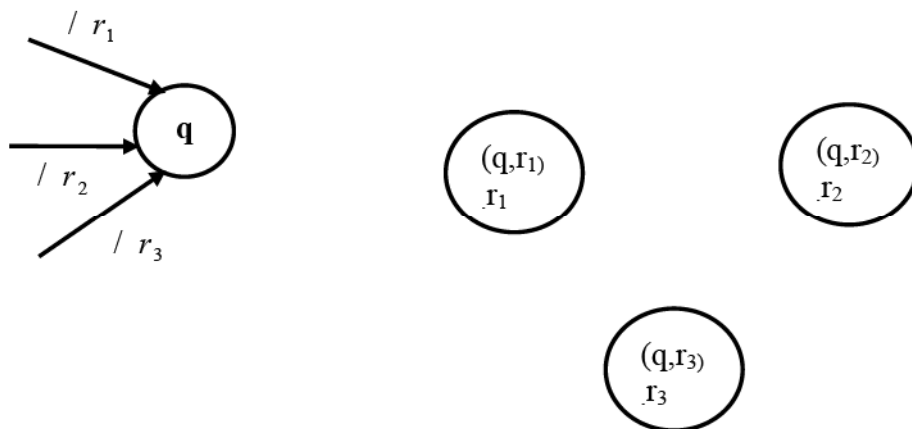


$$M_t = (Q_t, S, R, FT, G, q_i)$$

$$M_s = (Q, S, R, F_s, H, (q_i, r_0))$$

M_t

M_s

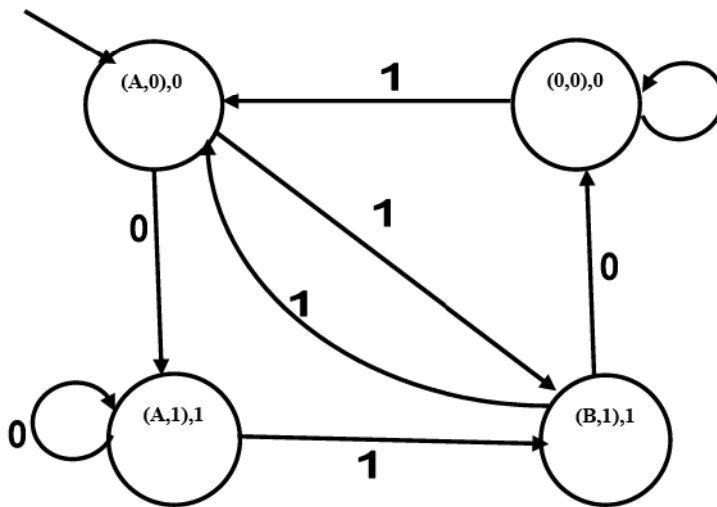
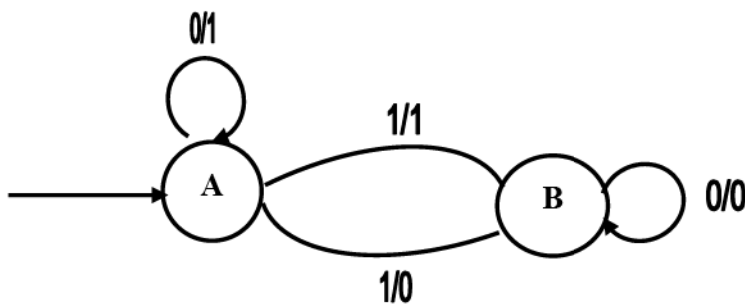


هنگامی در M_t انتقالی به صورت $q \xrightarrow{r} q'$ داشته باشیم در M_s انتقال

$((q, r'), r') \xrightarrow{s} ((q', r), r)$ خواهیم داشت.
 $(r' \in R)$

انتقال $q_i \xrightarrow{s/r} q'$ در M_t به انتقال $((q_i, r_0), r_0) \xrightarrow{s} ((q', r), r)$ در M_s تبدیل می شود.

مثال:



معادل بودن ماشینهای حالت محدود:

۱- با داشتن دیاگرام حالت ماشین آیا ممکن است حالات اضافی را تشخیص داده و حذف نمائیم بدون اینکه

رفتار ماشینی تغییر نماید؟

۲- آیا این امکان وجود دارد که با حذف حالات اضافی یک ماشین با حداقل حالت معادل با ماشین اولیه پیدا

کنیم؟

تعریف: دو ماشین M_1 و M_2 معادل هستند اگر و فقط اگر:

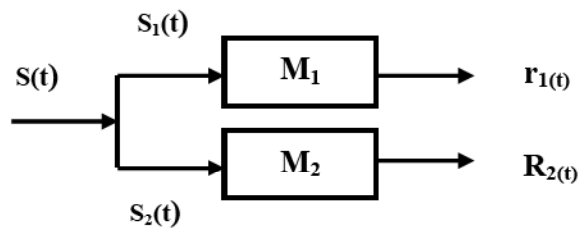
۱. الفبای ورودی آنها و الفبای خروجی آنها یکسان باشد.

$$S_1 = S_2, R_1 = R_2$$

۲- برای هر تحریک M_1 و M_2 پاسنهای یکسانی تولید نمایند یعنی اینکه اگر $s_1(t) = s_2(t)$

$$r_1(t) = r_2(t) \text{ آنگاه}$$

حالت معادل بودن را به صورت $M_1 = M_2$ نمایش می دهیم.



رابطه معادل بودن در ماشین یک رابطه هم ارزی است زیرا:

$$1- \text{انعکاسی } M \approx M$$

$$2- \text{تقارنی } \text{if } M_1 \approx M_2 \Rightarrow M_2 \approx M_1$$

$$3- \text{تعدی } \text{if } M_1 \approx M_2 \wedge M_1 \approx M_3 \Rightarrow M_1 \approx M_3$$

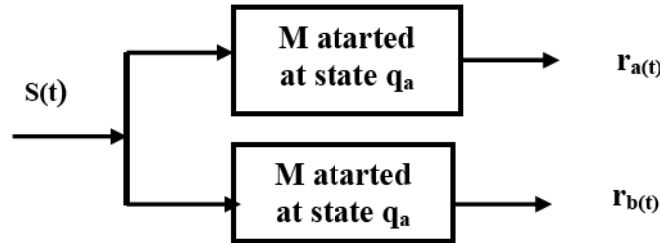
$$\begin{cases} r_1(t) = r_2(t) \\ r_2(t) = r_3(t) \end{cases} \Rightarrow r_1(t) = r_3(t)$$

حالات معادل:

تعریف: دو حالت q_a, q_b در یک ماشین Transition Assigned مثل $M = (Q, S, R, F, G, q_I)$ معادل هستند اگر

و فقط اگر ماشینهای $M_a = (Q, S, R, F, G, q_a)$ و $M_b = (Q, S, R, F, G, q_b)$ معادل باشند.

نحوه نشان دادن $(q_a \approx q_b)$



$$r_a(t) = r_b(t) \\ t \geq 1$$

قضیه: حالات q_b, q_a از یک ماشین حالت محدود مثل M معادل هستند اگر و فقط اگر:

1.a Transition Assigned: برای تمامی $s \in S$ $g(q_b, s) = g(q_a, s)$

یعنی اینکه خروجی برای انتقالات متناظر در دو حالت یکسان باشد.

1.b state-assigned: $h(q_a) = h(q_b)$ یعنی اینکه سمبل خروجی در دو حالت یکسان باشد.

2. برای تمامی $s \in S$ و $f(q_a, s) \approx f(q_b, s)$ یعنی اینکه s -successorهای دو حالت معادل باشند.

اثبات: اثبات اگر: بایستی نشان بدهیم اگر شرایط 1 و 2 برقرار باشند آنگاه q_b, q_a معادل هستند.

فرض کنیم رشته $S\omega$ یک رشته ورودی انتخابی شامل سمبل $s \in S$ و رشته $\omega \in S^*$ باشد رفتار M را در

دو حالت مورد بررسی قرار می دهیم.

$$r\varphi = r'\varphi' \quad \begin{array}{l} q_a \xrightarrow{s/r} q'_a \xrightarrow{\omega/\varphi} q''_a \\ q_b \xrightarrow{s/r'} q'_b \xrightarrow{\omega/\varphi} q''_b \end{array}$$

اثبات فقط اگر: بایستی نشان بدهیم اگر $q_a \approx q_b$ آنگاه شرایط 1 و 2 برقرار هستند.

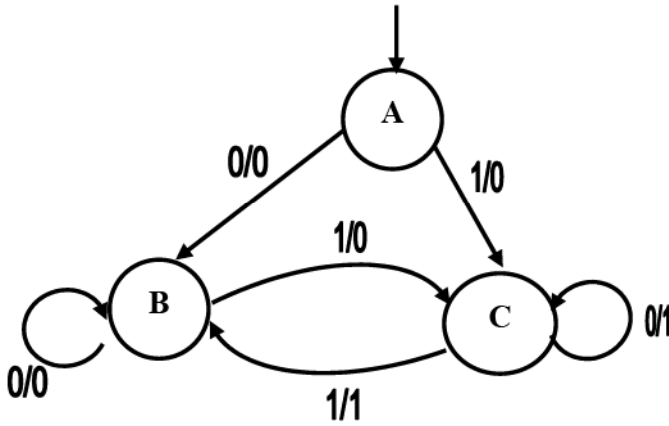
$$\begin{array}{l} q_a \xrightarrow{s/r} q'_a \xrightarrow{\omega/\varphi} q''_a \\ q_b \xrightarrow{s/r'} q'_b \xrightarrow{\omega/\varphi} q''_b \end{array} \quad \text{داریم: } r\varphi = r'\varphi'$$

پس $\varphi = \varphi', r = r'$

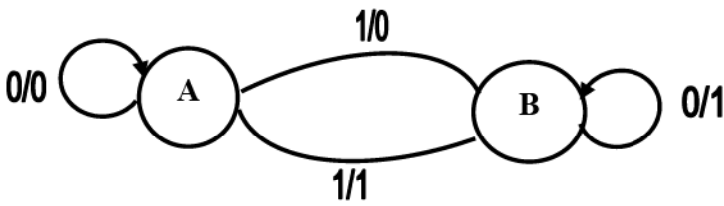
وقتی $r = r'$ در نتیجه شرط 1 برقرار است.

وقتی $\varphi = \varphi'$ در نتیجه شرط 1 برقرار است.

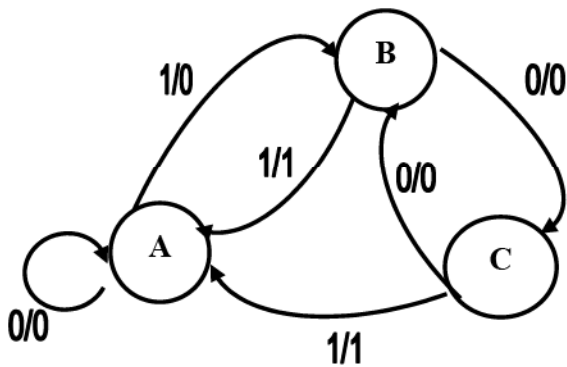
مثال: آیا $A \approx B$ است؟



- | | | |
|------------|---|------------|
| $g(A,0)=0$ | , | $g(B,0)=0$ |
| $g(A,1)=0$ | , | $g(B,1)=0$ |
| $g(A,0)=B$ | , | $g(B,0)=B$ |
| $g(A,1)=C$ | , | $g(B,1)=C$ |



مثال: آیا $B \approx C$ است؟



- | | | |
|------------|---|------------|
| $g(B,0)=0$ | , | $g(C,0)=0$ |
| $g(B,1)=1$ | , | $g(C,1)=1$ |
| $g(B,0)=C$ | , | $g(C,0)=B$ |
| $g(B,1)=A$ | , | $g(C,1)=A$ |

کم کردن حالت و تست معادل بودن حالات

تعریف: یک ماشین حالت محدود تقلیل یافته (REDUCED) است اگر شامل هیچ جفت حالات معادل نباشد.

تعریف: یک حالت q در یک اتوماتای محدود قابل دسترسی (accessible) است اگر ورودیهایی مثل ω وجود

داشته باشد که $q_i \xrightarrow{\omega} q$ یک ماشین حالت محدود متصل (connected) است اگر هر حالت از آن قابل

دسترسی باشد.

دنباله های تمیز دهنده و معادل بودن k تایی: Distinguishing sequences & k -Equivalence:

تعریف: حالت های q_b, q_a از یک ماشین Transition Assigned مثل $M=(Q,S,R,F,G,QI)$

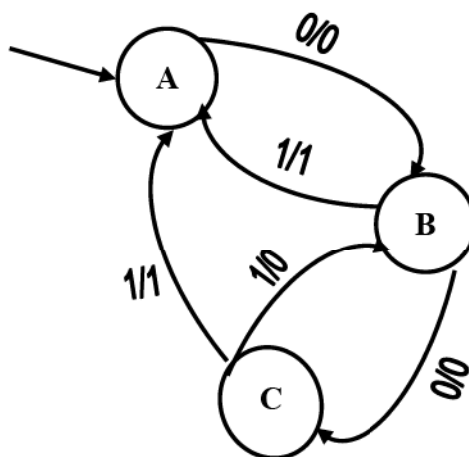
K -distinguishable (متمايز از مرتبه K) هستند اگر و فقط اگر رشته ای مثل $\omega \in s^*$ با $|\omega| \leq k$ وجود

داشته باشد که پاسخ ماشینهای $M_a=(Q,S,R,F,G,q_a)$ و $M_b=(Q,S,R,F,G,q_b)$ برای ω حداقل در یک سمبل

متفاوت باشد.

رشته ω را دنباله تمیز دهنده در حالت q_b و q_a گویند q_b و q_a متممايز از مرتبه k نباشد آنها را معادل از مرتبه k گویند

(k -Equivalence).



قضیه: دو حالت در یک ماشین حالت محدود k -Equivalence هستند اگر و فقط اگر:

۱- آنها 1 -equivalence باشند.

۲. برای هر عمل ورودی مثل s -successors آنها $(k-1)$ -Equivalence باشند.

افزار نمودن مجموعه حالات:

اگر p_1, p_2 افزارهایی از مجموعه X باشند و اگر هر بلاک از p_2 دقیقاً زیر مجموعه یک بلاک از p_1 باشد گوئیم

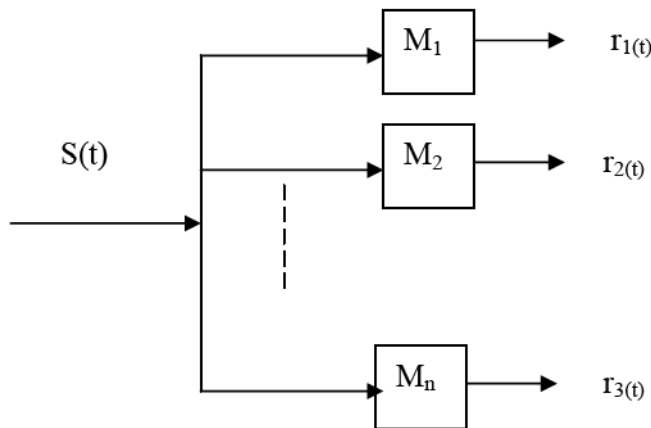
p_2 تصفیه ای (refinement) باری p_1 است. یعنی اگر $p_1 = \{A_1, A_2, \dots, A_n\}$ و

$p_2 = \{B_1, B_2, \dots, B_m\}$ افزارهایی از مجموعه X باشند و p_2 تصفیه ای از p_1 باشد آنگاه برای هر بلاک B_j در

p_1 یک بلا مثل A_i در p_1 وجود داشته باشد بقسمی که:

$$B_j \subseteq A_i$$

$$m \geq n$$



فرض کنیم $M = (Q, S, R, F, G, q_1)$ ماشینی است که می خواهیم مجموعه حالات آن یعنی Q را افزار نمائیم.

فرض می کنیم $Q = \{q_1, \dots, q_n\}$ باشد و فرض کنیم $M_i = (q, s, r, f, g, q_i)$

حالات q, q_i متعلق به یک بلاک که از افزاری مثل p از Q هستند اگر هیچ تحریکی به ماشینهای M_i, M_j پاسخهای متفاوتی را در این دو ماشین ایجاد نکند.

می خواهیم دنباله های افزارهای $p_1, p_2, p_3, \dots, p_m, p_{m+1}$ تشکیل دهیم تبسمی که هر بلاک از افزار $p_k, 1 < k < m+1$ شامل تنها حالاتی که به صورت دبدو k -Equivalence.

$$P_1 = \{A_{11}, A_{12}, \dots, A_{1n}\}$$

$$P_2 = \{A_{21}, A_{22}, \dots, A_{2n}\}$$

تمرین

۱، ۳. گرامرهای زیر را در نظر بگیرید.

$$\begin{array}{l} \mathbf{G_1:} \\ \Sigma \longrightarrow \lambda \\ \Sigma \longrightarrow S \\ S \longrightarrow ss \\ S \longrightarrow c \end{array}$$

$$\begin{array}{l} \mathbf{G_2:} \\ \Sigma \longrightarrow \lambda \\ \Sigma \longrightarrow S \\ S \longrightarrow cSd \\ S \longrightarrow cd \end{array}$$

$$\begin{array}{l} \mathbf{G_3:} \\ \Sigma \longrightarrow \lambda \\ \Sigma \longrightarrow S \\ S \longrightarrow Sd \\ S \longrightarrow cS \\ S \longrightarrow c \\ S \longrightarrow d \end{array}$$

$$\begin{array}{l} \mathbf{G_4:} \\ \Sigma \longrightarrow cS \\ S \longrightarrow d \\ S \longrightarrow cS \\ S \longrightarrow Td \\ T \longrightarrow Td \\ T \longrightarrow d \end{array}$$

$$G_5: \begin{aligned} \Sigma &\longrightarrow \lambda \\ \Sigma &\longrightarrow S \\ S &\longrightarrow ScS \\ S &\longrightarrow c \end{aligned}$$

الف) زبانهای $L(G_i)$ را برای $i=1, \dots, 5$ مشخص کنید.

ب) رابطه بین $L(G_i)$ ها را مشخص کنید.

ج) برای هر زبان یک اشتقاق برای رشته ای با طول ۴ پیدا کنید.

۲، ۳. یک گرامر که هر یک از زبانهای زیر را تولید می کند تشکیل دهید.

$$\text{الف) } \{0^m 1^n \mid m > n \geq 0\}$$

$$\text{ب) } \{0^k 1^m 0^n \mid n = k + m\}$$

$$\text{ج) } \left\{ \begin{array}{l} 0^m 1^n \mid \begin{array}{l} m(\text{even}) \\ n(\text{odd}) \end{array} \\ \text{or} \\ \begin{array}{l} m(\text{odd}) \\ n(\text{odd}) \end{array} \end{array} \right\}$$

$$\text{د) } \{\omega c \omega \mid \omega \in \{0,1\}^*\}$$

افراز نمودن مجموعه در جملات

اگر P_1 و P_2 افرازهایی از مجموعه X باشند و اگر هر بلاک از P_2 دقیقاً زیر مجموعه یک بلاک از P_1 باشد

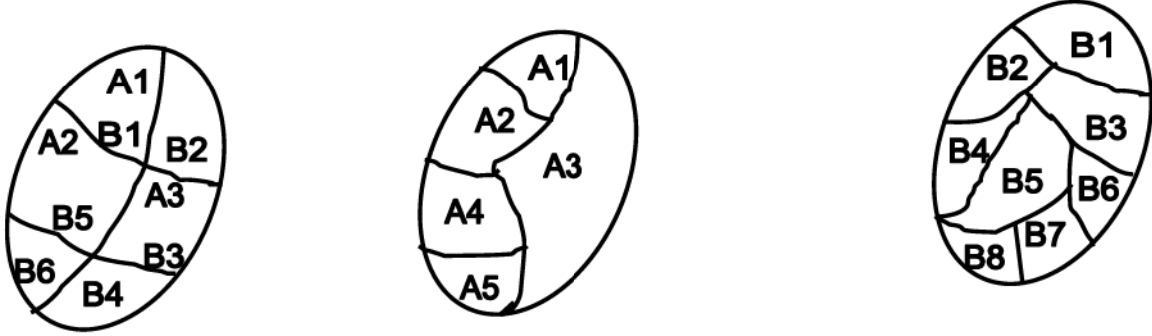
گوئیم P_2 تصفیه ای برای P_1 است

$$P_1 = \{A_1, A_2, \dots, A_n\}$$

$$P_2 = \{B_1, B_2, \dots, B_n\}$$

افرازهایی از مجموعه X باشد و P_2 تصفیه ای از P_1 باشد آنگاه برای هر بلاک B_j در P_2 یک بلاک مثل A_i در

P_1 وجود داشته باشد به قسمی $B_j \subseteq A_i$ $m \geq n$ باشد.



هر افرازی یک تصفیه برای خودش می باشد

فرض کنیم $M=(Q,S,R,f,G,q_i)$ ماشین حالت محدودی است که می خواهیم مجموعه حالات آن یعنی Q را افراز کنیم فرض می کنیم $Q=\{q_1, q_2, \dots, q_n\}$ و فرض می کنیم حالت آغازین $M_i=(Q \leq S \leq R, f, g, q_i)$ باشد حالات q_i, q_j متعلق به یک بلاک (از افرازی مثل P از Q) هستند اگر هیچ تحریکی به ماشینهای M_i, M_j پاسخهای متفاوتی را در این دو ماشین ایجاد نکنند. (Equivalence)

می خواهیم دنباله افرازهای $P_1, P_2, \dots, P_m, P_{m+1}$ را تشکیل دهیم به قسمی که هر بلاک از افراز P_k که $1 \leq k \leq m+1$ شامل تنها حالاتی باشد که به صورت دو به دو K -Equivalence هستند.

الگوریتم افزار سازی

۱- یک افزار ابتدایی مثل p_1 از Q با گروه بندی حالاتی که 1 -equivalent هستند تشکیل بده یعنی حالاتی که خروجی یکسانی برای هر سمبل ورودی دارند را در یک بلاک قرار بده. حالات q, q' در یک بلاک از p_1 هستند

اگر و فقط اگر $\forall s \in S \quad g(q,s)=g(q',s)$

۲- p_{k+1} را از روی p_k بصورت زیر بدست آور: q, q' در یک بلاک از p_{k+1} قرار دارند اگر و فقط اگر:

a. در یک بلاک از p_k قرار داشته باشند.

b. برای هر $s \in S'$ و successorهای آنها یعنی حالات $f(q,s)$ و $f(q',s)$ در یک بلاک از p_k قرار داشته باشند.

۳. قدم ۲ را تا جائیکه $p_m = p_{m+1}$ برای یک m شود تکرار کن p_m ، هفراز های Q است.

$$A P_m = \{A_1, A_2, \dots, A_n\}$$

قضیه: یک روال موثر برای افزاز کردن حالات یک ماشین حالت محدود به بلاکهایی از حالات معادل وجود دارد.

مثال:

	0	1
A	B,0	C,0
B	C,1	D,1
C	D,0	E,0
D	C,1	B,1
E	F,1	E,1
F	G,0	C,0
G	F,1	G,1
H	J,1	B,0
J	H,1	D,0

$$P_1 : \{A, C, F\}, \{B, D, E, G\}, \{H, J\}$$

$$P_2 : \{A, F\}, \{C\}, \{B, D, E, G\}, \{H, J\}$$

$$P_3 : \{A, F\}, \{C\}, \{B, D\}, \{E, G\}, \{H, J\}$$

$$P_4 : \{A\}, \{F\}, \{B, D\}, \{E, G\}, \{H, J\} \text{ پارتیشن نهائی:}$$

$$P_5$$

⋮

مثال:

	0	1	
A	B	A	0
B	C	D	0
C	E	C	0
D	F	B	0
E	G	E	0
F	H	F	0
G	I	G	0
H	J	H	0
I	A	K	1
J	K	J	0
K	A	k	1

$$P_1 : \{A, B, C, D, E, F, G, H, J\}, \{I, K\}$$

$$P_2 : \{A, B, C, D, E, F, H\}, \{G, J\}, \{I, K\}$$

$$P_3 : \{A, B, C, D, F\}, \{E, H\}, \{G, J\}, \{I, K\}$$

$$P_4 : \{A, B, D\}, \{C, F\}, \{E, H\}, \{G, J\}, \{I, K\}$$

$$P_5 : \{A\}, \{B, D\}, \{C, F\}, \{E, H\}, \{G, J\}, \{I, K\}$$

P_6
⋮

در ماشین تقلیل یافته بجای ۱۱ حالت ما ۶ حالت داریم.

$$P_5 : \underbrace{\{A\}}_U, \underbrace{\{B, D\}}_V, \underbrace{\{C, F\}}_W, \underbrace{\{E, H\}}_X, \underbrace{\{G, J\}}_Y, \underbrace{\{I, K\}}_Z$$

	0	1	
U	V	U	0
V	W	V	0
W	X	W	0
X	Y	X	0
Y	Z	Y	0
Z	U	Z	1

A با صفر به B رفته و چون B در مجموعه V قرار دارد

پس در داخل جدول V را قرار می دهیم.

مثال: فرض کنید $M = (Q, R, S, f, g, q_I)$ یک ماشین Transition Assigned باشد. می خواهیم ماشینی مثل

$M' = (Q', R', S', f', g', q'_I)$ که معادل ماشین M و بصورت تقلیل یافته است تشکیل دهیم. فرض کنیم p_F

افزار نهائی Q که شامل بلاکهائی از حالات معادل است، باشد. حالت ابتدائی ماشین M' با قواعد زیر تشکیل می‌شود:

۱. برای یافتن S-Successor حالت q' در M' که متناظر با q' را در نظر گرفته S-Successor آن حالت را یافته و بررسی می‌کنیم که این S-Successor در چه بلاکی از p_F قرار دارد. بلاک متناظر با آن نمایشگر S-Successor حالت q' در M' است.

$$p_F = \{ \}, \{ \}_{q'} \rightarrow \{ \}$$

۲. خروجی یک حالت مثل q' در M' معادل خروجی یکی از حالات بلوک متناظر با q در p_F است.

Finite State Language

فصل پنجم

ارتباط Finite State Machines و گرامرهای Regular:

هدف:

۱. زبان L بوسیله یک یا چند Finite State acceptor تشخیص داده می شود.
۲. زبان L بوسیله یک یا چند گرامر منظم تولید می شود.
۳. زبان L بوسیله یک یا چند عبارت منظم تشریح می شود.

: Finite State Acceptor

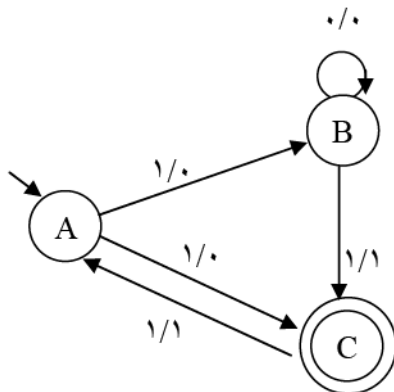
اگر بخواهیم یک ماشین Transition Assigned را با F.S.A. نشان دهیم، سمبل های خروجی $R = \{0,1\}$

خواهند بود. انواع حالات عبارت خواهند بود از:

۱. حالات قبول: خروجی ۱ است.
۲. حالات Rejecting: خروجی ۰ است.

:Nondeterministic Acceptors

شکل مقابل نمونه ای از یک N.A. میباشد. در این شکل از نمایش خروجی ها صرف نظر شده و بجای آن حالاتی را که با دو دایره مشخص شده اند بعنوان حالت نهائی معرفی می کنیم. اینها تنها حالاتی هستند که خروجی ۱ دارند. این خروجی ها با رنگ قرمز مشخص شده اند.



در $F.S.M$ باید در هر حالت بر اساس هر سمبل ورودی یک انتقال داشته باشیم ولی در $N.A$ این شرط را نداریم.

تعریف: یک $F.S.A$ یک پنج تایی بفرم $M=(Q,S,P,I,F)$ است که در آن:

Q : نشان دهنده مجموعه حالات است.

S : الفبای ورودی است.

$I \subseteq Q$: I و نشاندهنده حالات پایانی است.

$F \subseteq Q$: F و نشاندهنده حالات پایانی است.

$P \subseteq Q \times S \times Q$: P و رابطه انتقال ماشین M است.

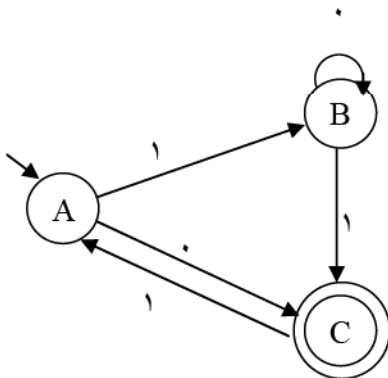
(q,s,q') عضوی از P است اگر $q \xrightarrow{s} q'$

تعریف: فرض کنیم $M=(Q,S,P,I,F)$ یک $F.S.A$ باشد. اگر انتقالات P از M را بتوان بصورت یک تابع

$P: Q \times S \rightarrow Q$ نوشت و اگر M دقیقاً دارای یک حالات ابتدائی باشد. آنگاه M را **Deterministic Finite**

State Acceptor گویند.

برای رشته $|0|$ ، دنباله‌های انتقال زیر را میتوان نوشت:



$$A \xrightarrow{1} B \xrightarrow{0} B \xrightarrow{1} C$$

$$A \xrightarrow{1} C \xrightarrow{0} A \xrightarrow{1} C$$

$$A \xrightarrow{1} C \xrightarrow{0} A \xrightarrow{1} B$$

دنباله سوم مورد قبول نمیباشد، زیرا حالت B ، حالت نهائی نمیباشد اما دو دنباله اول قابل قبولند. اگر با رفتن از یک سیر بجواب نرسیدیم (در یک $N.A.$)، باید کار را با مسیرهای دیگر ادامه دهیم و این باعث کند شدن ماشینی (شبيه سازی شده) میگردد.

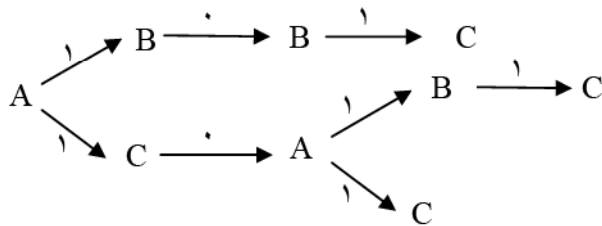
عبارت $q \xrightarrow{w} q'$ در ماشینهای قبلی، در این جا مفهوم درستی ندارد و می نویسم:
اما میتواند ماشینی را از حالت q به q' منتقل کند.

تعریف: فرض کنید $M=(Q,S,P,I,F)$ یک FSA (نامعین) باشد، آنگاه رشته ای مثل $W \in S^*$ را قبول می کند،

اگر و فقط اگر برای برخی از $q \in I, q' \in F$ داشته باشیم: $q \xrightarrow{w} q'$.

$$L(M) = \{ W \in S^* \mid \text{m رشته } w \text{ را قبول کند.} \}$$

مثال: آیا ماشین بالا رشته 1011 را می پذیرد یا نه؟



چون C روی 1 انتقالی ندارد، مسیر قابل قبول است.

چون C روی 1 انتقالی ندارد، مسیر غیر قابل قبول میباشد.

ماشینهای FSA از نوع نامعین برای پیاده سازی مناسب نیستند و آنها را به ماشینهای معین تبدیل می کنیم. با در

نظر گرفتن رفتار ماشین در هر تصمیم گیری بصورت در ماشین مجزا (که موازی کار میکنند)، رابطه ی بشکل

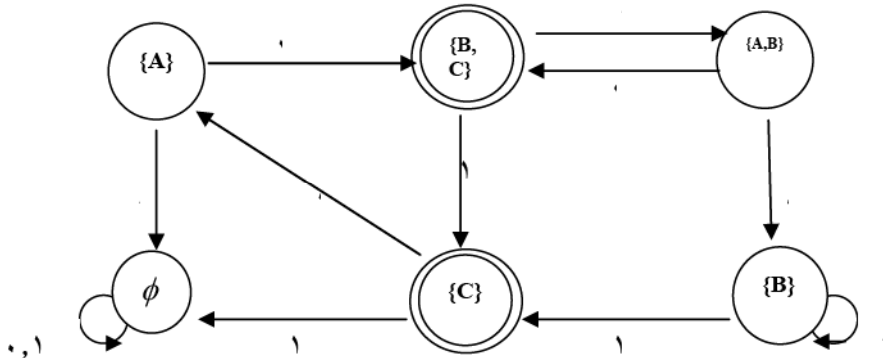
زیر خواهیم داشت:

$$\{A\} \xrightarrow{1} \{B, C\} \xrightarrow{0} \{B, A\} \xrightarrow{1} \{B, C\} \xrightarrow{1} \{C\}$$

کل مسیرهای قابل پیمایش را بصورت زیر بدست می آوریم:

بردار

با در نظر گرفتن هر یک از مجموعه‌های بدست آمده، بعنوان یک حالت ماشین را به نوع معین تبدیل میکنیم:



از آنجائیکه هنوز بعضی از حالات بازاء بعضی سمبل‌های ورودی انتقالی مشخص نمی‌کنند، این ماشین، معین نخواهد بود. برای تبدیل آن یک حالت جدید ϕ را به ماشین اضافه می‌کنیم (حالت قرمز)

- در ماشین جدید حالاتی پایانی هستند که شامل حالات پایانی ماشین اولیه باشد. (در اینحالت $\{B,C\}$ ، $\{C\}$)

- اگر حالتی از ماشین بازاء یک سمبل ورودی S متعلق به S ، انتقالی را مشخص نکرد، از آن حالت انتقال جدیدی را روی همان سمبل ورودی به حالت ϕ ایجاد می‌نمائیم.

فرض کنید $M_n = (Q_n, S, P_n, I_n, F_n)$ یک FSA (نامعین) باشد، می‌خواهیم ماشین $M_d = (Q_d, S, P_d, I_d, F_d)$ (معین) را تشکیل می‌دهیم، بطوریکه $L(M_n) = L(M_d)$.

$$X[W] = \left\{ q' \in Q_n \mid q \xrightarrow{w} q', q \in I_n \right\} \quad X[w] \text{ را بصورت مقابل تعریف میکنیم:}$$

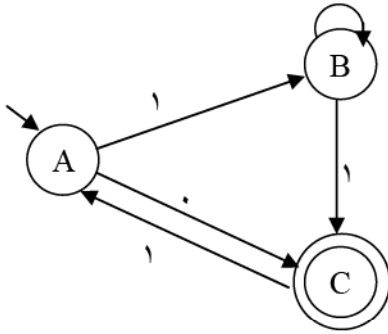
بعبارتی X مجموعه حالاتی است که ماشین با دیدن رشته W ، می‌تواند در آن قرار داشته باشد. یا:

X مجموعه حالات قابل رسیدن (Reachable) برای رشته W میباشد.

$$X[\lambda] = I_n$$

$$X[\gamma.S] = \left\{ q' \in Q_n \mid q'' \xrightarrow{s} q', q'' \in X[\gamma] \right\}$$

مثال:



$$X[\lambda] = \{A\}$$

$$X[\underset{\lambda,0}{0}] = \{q' \in Q \mid q' \xrightarrow{0} q'', q'' \in X[\lambda]\} = \emptyset \quad \text{انتقالات A روی } \emptyset$$

$$X[\underset{\lambda,1}{1}] = \{B, C\} \quad \text{و} \quad X[1\underset{\lambda,1}{0}] = \{q' \mid q' \xrightarrow{0} q'', q'' \in X[1]\} = \{A, B\} \quad \text{انتقالات C, B روی } \emptyset$$

ساخت ماشین معین:

$$Q_d = \{X[W] \mid W \in S^*\} \quad 1. \text{ عناصر } Q_d \text{ زیر مجموعه‌هائی از } Q_n \text{ هستند:}$$

$$X[\lambda], I_d \quad 2. \text{ حالت شروع در } M_d \text{، } X[\lambda] \text{ است:}$$

$$F_d = \{X \in Q_d \mid X \cap F_n \neq \emptyset\} \quad 3. \text{ حالت قبول ماشین } M_d:$$

4. حالت X' در M_d ، S-Successor، X است اگر X' شامل دقیقاً حالتی از M_n باشد که S-

Successor حالت‌های مجموعه X در M_n هستند.

4 عمل فوق را تا زمانی که X ها تکراری نباشند، ادامه می‌دهیم. (در هر مرحله)

اگر $X[s]$ ، تهی شد، $X[s, \gamma]$ هم تهی می‌شود و آنرا ادامه نمی‌دهیم.

$$X[11] = \{C\}$$

$$X[100] = \{B\}$$

$$X[101] = \{B, C\}$$

$$X[110] = \{A\}$$

$$X[111] = \emptyset$$

چون $\{B, C\}$ قبلاً بدست آمده بود، رشته 101 را ادامه نمی‌دهیم.

چون $\{A\}$ قبلاً بدست آمده بود، رشته 110 را ادامه نمی‌دهیم.

با توجه به محاسبات اخیر، مجموعه حالات و انتقال‌های همانند شکل جلسه قبل بدست می‌آید.

قضیه: برای هر ماشین FSA مثل M_n میتوان یک ماشین معین مثل M_d تشکیل داد بقسمی که:

$$L(M_d) = L(M_n)$$

اثبات: فرض کنیم M_d ، FSA تشکیل شده بوسیله روال گفته شده باشد، بطور وضوح M_d یک ماشین معین

$$L(M_d) = L(M_n) \text{ خواهد بود. باید نشان دهیم}$$

حالت $X[w]$ در M_d ، حالت قبول در M_d است اگر و فقط اگر شامل یکی از حالات قبول M_n باشد. چون

هر حالت در $X[w]$ برای رشته w در M_n قابل دسترسی است بنابراین $X[w] \in F_d$ اگر و فقط اگر $w \in L(M_n)$.

کافیست ثابت کنیم هر رشته مثل w ماشین M_d را بحالت $X[w]$ منتقل می سازد:

$$W \in S^*, X[\lambda] \xrightarrow{w} X[w]$$

بصورت استقراء:

$$\text{پایه: اگر } w = \lambda \text{ باشد } X[\lambda] \xrightarrow{\lambda} X[\lambda]$$

فرض کنیم $W = \gamma.S$ همچنین $X[\lambda] \xrightarrow{\gamma} X[\gamma]$. (اگر γ رشته ای بطول n باشد $\gamma.S$ رشته ای با طول $n+1$ خواهد

$$\text{بود. می خواهیم ثابت کنیم: } X[\lambda] \xrightarrow{\gamma.S} X[w]$$

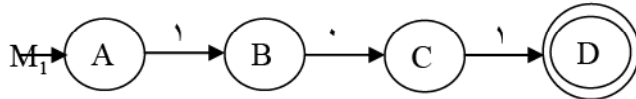
$$X[w] = X[\gamma.S] \left\{ q' \mid q'' \xrightarrow{s} q', q'' \in X[\gamma] \right\}$$

$$\text{فرض: } X[\lambda] \xrightarrow{\gamma} X[\gamma] \xrightarrow{s} X[w] \quad \Rightarrow \quad X[\lambda] \xrightarrow{\gamma.S} X[w]$$

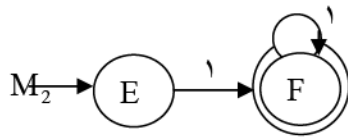
نتیجه: قدرت ماشینهای نامعین و ماشینهای معین بیک اندازه است. بنابراین در شرایطی که طراحی N.A ساده تر

باشد، پس از طراحی ماشین نامعین، آنرا، ماشین معین تبدیل میکنیم.

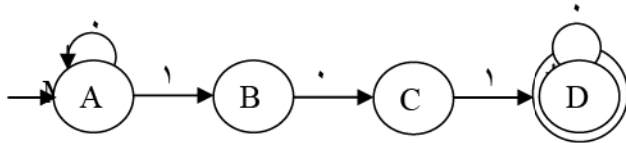
مثال : فرض کنید میخواهیم یک ماشین FSA معین درست کنیم که رشته‌هایی که شامل تعدادی صفر بوده و یکبار زیر رشته ۱۰۱ یا زیر رشته‌ای از تعدادی ۱ در آن ظاهر شده باشد را تشخیص دهد.
 - رشته‌هایی که ماشین باشد تشخیص دهد بصورت $0^*(101 \cup 11^*)0^*$ می‌باشند.



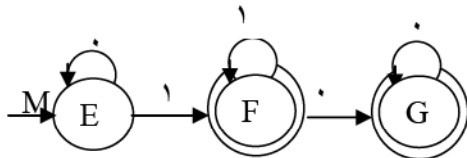
$$L(M_1) = \{101\}$$



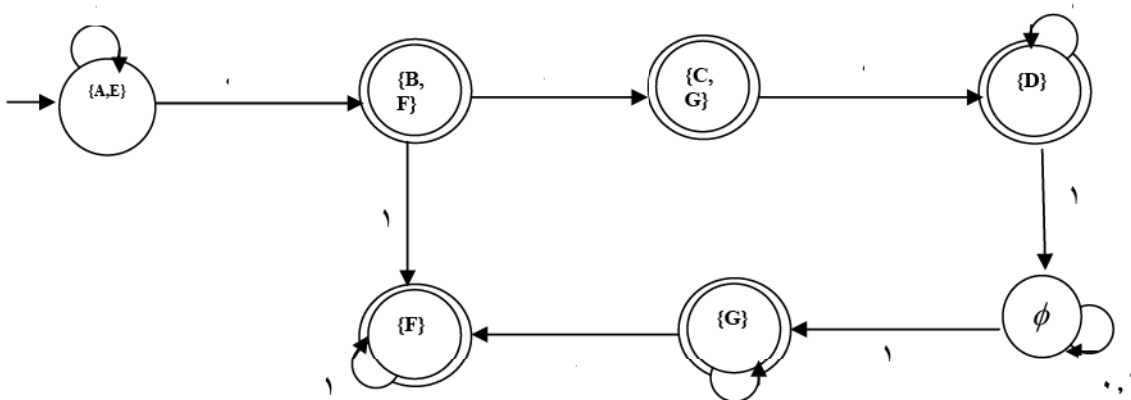
$$L(M_2) = \{11^*\}$$



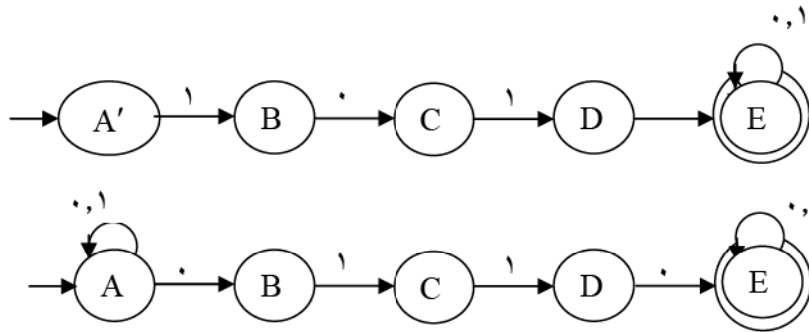
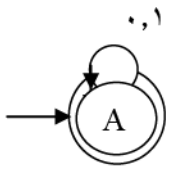
$$L(M_3) = \{0^*1010^*\}$$



$$L(M_4) = \{0^*11^*0^*\}$$



مثال : ماشین FSA را که رشته‌های زیر را قبول می‌کند طراحی کنید.



	0	1	
A	A,B	A	
B	-	C	
C	-	D	
D	E	-	
E	E	E	1

	0	1	
A	AB	A	
AB	AB	AC	
AC	AB	AD	
AD	ABE	A	
ABE	ABE	ACE	1
ACE	ABE	ADE	1
ADE	AB	AE	1
AE	AB	AE	1

پذیره‌های حالت محدود و گرامرهای منظم:

$$L(G, A) = \{ W \in T^* \mid A \Rightarrow^* W \text{ in } G \}$$

$$\begin{aligned} A &\rightarrow 0 \mid 1B \\ B &\rightarrow 0 \mid 0B \end{aligned}$$

\Rightarrow

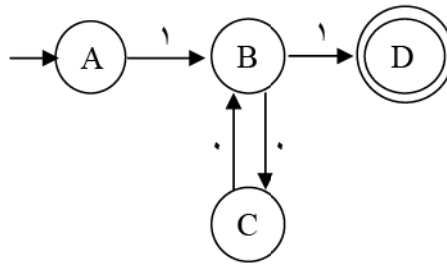
$$\begin{aligned} L(G, A) &= \{0 \cup 100^*\} \\ L(G, B) &= \{00^*\} \end{aligned}$$

تعریف: فرض می‌کنیم $M=(Q,S,P,I,F)$ یک FSA باشد، $E(q)$ (end set) برای حالتی مثل q از M مجموعه رشته‌های ورودی هستند که ماشین را از حالت q به یک حالت قبول در M منتقل می‌کنند.

$$E(q) = \{W \in S^* \mid q \xrightarrow{W} q', q' \in F\}$$

مثال:

$$E(A) = 1.E(B) \quad \text{و} \quad E(B) = 0.E(C) \cup 1 \quad \text{و} \quad E(C) = 0.E(B) \quad \text{و} \quad E(D) = \lambda$$



نتایج ۱:

$$1. \lambda \in E(q) \text{ اگر و فقط اگر } q \in F$$

۲. اگر $W = S.\gamma$ آنگاه $W \in E(q)$ اگر و فقط اگر $q \xrightarrow{s} q', \gamma \in E(q')$ انتقالی در M باشد.

$$3. L(M) = \bigcup_{q \in I} E(q)$$

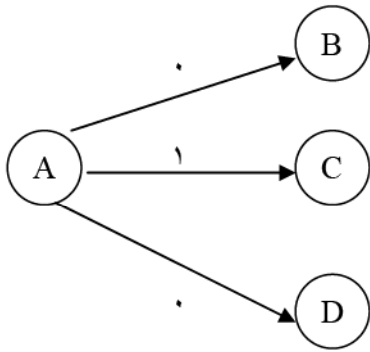
نتیجه ۲:

End set های یک ماشین پذیرنده محدود مثل $M=(Q,S,P,I,F)$ یک سیستم مجموعه معادلات خطی از سمت راست را (right-linear set equations) ارضا می‌نماید. یعنی:

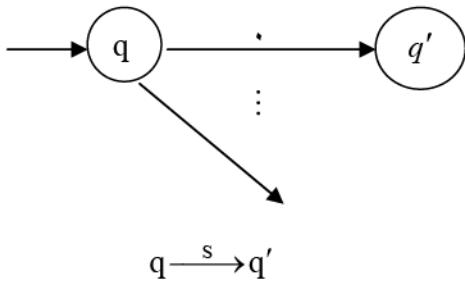
$$q \in Q, \quad E(q) = \bigcup_{q' \in Q} V(q, q') E(q') \cup W(q)$$

$$V(q, q') = \{s \in S \mid q \xrightarrow{s} q'\}$$

$$W(q) = \begin{cases} \lambda & \text{if } q \in F \\ \emptyset & \text{other wise} \end{cases}$$



$$E(A) = 0.E(B) \cup 1.E(C) \cup 0.E(D)$$



$$E(q) = 0.E(q') \cup \dots$$

$$E(q) \supseteq 0.E(q')$$

$$A \rightarrow sB$$

$$E(q) \supseteq S.E(q')$$

$$L(G, A) \supseteq S.L(G, B)$$

$$G: \Sigma \rightarrow A$$

$$A \rightarrow 1B$$

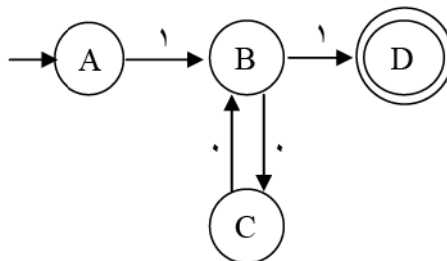
$$B \rightarrow 1$$

مثال:

$$B \rightarrow 0C$$

$$C \rightarrow 0B$$

M:



$$L(\Sigma) = L(A)$$

$$L(A) = 1.L(B)$$

$$L(B) = 0.L(C) \cup 1$$

$$L(C)=0.L(B)$$

$$E(A)=1.E(B) , E(B)=0.E(C) \cup 1$$

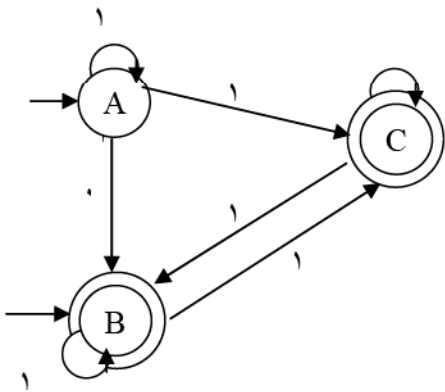
$$E(C)=0.E(B) , E(D)=\lambda$$

IN M	IN G
$A \xrightarrow{1} B$	$A \rightarrow 1B$
$B \xrightarrow{0} C$	$B \rightarrow 0C$
$C \xrightarrow{0} B$	$C \rightarrow 0B$
$B \xrightarrow{1} D, D \in F$	$B \rightarrow 1$
$A \in I$	$\Sigma \rightarrow A$

ساخت یک گرامر از روی یک ماشین پذیرنده حالت محدود:

دلیل	آنگاه G دارد	اگر M دارای	قاعده
$\lambda \in L(M)$	$\Sigma \rightarrow \lambda$	$I \cap F \neq \emptyset$	۱.
$E(q) = L(M)$	$\Sigma \rightarrow N(q)$	$q \in I$	۲.
$S \in E(q)$	$N(q) \rightarrow S$	$q \xrightarrow{s} q', q' \in F$	۳.
$E(q) \supseteq S.E(q')$	$N(q) \rightarrow S.N(q')$	$q \xrightarrow{s} q'$	۴.

مثال:



$$N = \{A, B, C\} , I = \{A, B\}$$

$$F = \{B, C\}$$

$\Sigma \rightarrow \lambda$	طبق قاعده ۱
$\Sigma \rightarrow A$	طبق قاعده ۲
$\Sigma \rightarrow B$	طبق قاعده ۲
$A \rightarrow 1$	طبق قاعده ۳
$B \rightarrow 1$	طبق قاعده ۳
$C \rightarrow 0$	طبق قاعده ۳
$C \rightarrow 1$	طبق قاعده ۳
$A \rightarrow 1A$	طبق قاعده ۴
$A \rightarrow 1C$	طبق قاعده ۴
$B \rightarrow 1B$	طبق قاعده ۴
$B \rightarrow 1C$	طبق قاعده ۴
$B \rightarrow 0A$	طبق قاعده ۴
$C \rightarrow 0C$	طبق قاعده ۴
$C \rightarrow 1B$	طبق قاعده ۴

قضیه: برای هر FSA مثل M ، می توان یک گرامر right-linear مثل G ساخت که $L(G)=L(M)$.

اثبات: فرض کنیم $M=(Q,S,P,I,F)$ داده شده باشد و فرض کنیم $G=(N,S=T,P,\Sigma)$ گرامر right-linear باشد که بوسیله و گفته شده از M تشکیل شده است.

کافیست اثبات کنیم اگر $W \in L(G)$ باشد آنگاه $W \in L(M)$ نیز می باشد و برعکس.

$$W \in L(G) \quad \text{if and only if} \quad W \in L(M)$$

اگر $\lambda \in L(M)$ پس داریم: $I \cap F \neq \emptyset$. اما با روش ساخت G, G باید شامل قاعده تولید $\lambda \rightarrow \Sigma$ باشد، در صورتیکه $I \cap F \neq \emptyset$. بنابراین اگر $\lambda \in L(G)$ و فقط اگر $\lambda \in L(M)$.

قواعد ساخت گرامر G از روی ماشین M ، تناظر یک به یک بین انتقالات M و قواعد تولید در G برقرار می-کند:

IN M	IN G
$q \xrightarrow{s} q'$	$N(q) \rightarrow S.N(q')$
$q \xrightarrow{s} q', q' \in F$	$N(q) \rightarrow S$

فرض کنیم $W = S_1 S_2 S_3 \dots S_k$ ، دنباله حالات زیر را در نظر می گیریم:

$$q_0 \xrightarrow{s_1} q_1 \xrightarrow{s_2} q_2 \xrightarrow{s_3} \dots \xrightarrow{s_k} q_k \quad q_0 \in I$$

$$q_k \in F$$

متناظر با این دنباله حالات میتوان اشتقاقی در G بصورت زیر تشکیل داد:

$$\begin{aligned} \Sigma \rightarrow N(q_0) &\Rightarrow S_1 N(q_1) \Rightarrow S_1 S_2 N(q_2) \Rightarrow \dots \\ &\Rightarrow S_1 S_2 \dots S_{k-1} N(q_{k-1}) \Rightarrow S_1 S_2 \dots S_k \end{aligned}$$

بنابراین $N(q_0) \stackrel{*}{\Rightarrow} W$ اگر و فقط اگر $W \in N(q_0)$.

پس اثبات کامل است.

نتیجه ۱: دنباله های حالات در M از $q \in I$ به $q' \in F$ در تناظر یک به یک با اشتقاقهای رشته های ترمینال از Σ در G می باشد.

نتیجه ۲: برای هر q ، مجموعه $E(q)$ از M و عنصر غیر پایانی $N(q)$ در G ، رابطه زیر را ارضا می کنند:

$$L(G, N(q)) = E(q)$$

(یعنی رشته‌هایی که میتوان با شروع از سمبل غیر پایانی $N(q)$ در G تولید نمود و $E(q)$ نشاندهنده رشته‌هایی است که میتوانند توسط M با شروع از q پذیرفته شوند).

ساخت یک FSA از روی یک گرامر Right-linear:

$$G = (N, T, P_G, \Sigma) \quad \text{ورودی:}$$

$$M = (Q, T, P_M, I, \{q_F\}) \quad \text{خروجی:}$$

مجموعه حالت ماشین M بصورت مقابل تعریف می شود: $Q = \{q_A \mid A \in N\} \cup \{q_F\}$

Rule	If G has	Then M has	Reason
1	$A \rightarrow sB$	$q_A \xrightarrow{s} q_B$	$L(G, A) \supseteq SL(G, B)$ $E(q_A) \supseteq SE(q_B)$
2	$A \rightarrow S$	$q_A \xrightarrow{s} q_F$	$S \in L(G, A)$ $S \in E(q)$
3	$\Sigma \rightarrow A$	$q_A \in I$	$L(G, A) \subseteq L(G)$ $E(q_A) \subseteq L(M)$
4	$\Sigma \rightarrow \lambda$	$q_F \in I$	$\lambda \in L(G)$

مثال: فرض کنید زبان $L = a^*b^* \cup (ab)^*$ را داشته باشیم. برای زبان L گرامری مانند G را بصورت زیر می -

توان نوشت:

$$G: \Sigma \rightarrow \lambda$$

$$A \rightarrow aA$$

$$C \rightarrow aD$$

$$\Sigma \rightarrow A$$

$$A \rightarrow aB$$

$$D \rightarrow bC$$

$$\Sigma \rightarrow C$$

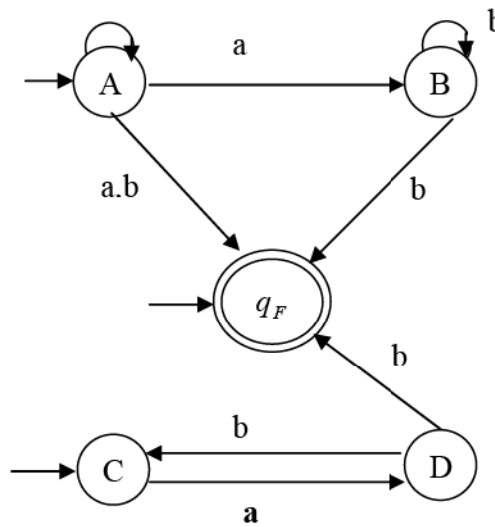
$$B \rightarrow bB$$

$$D \rightarrow b$$

$$A \rightarrow a$$

$$A \rightarrow b$$

$$B \rightarrow b$$



قضیه: برای هر گرامر right-linear مثل G می توان یک FSA مثل M تشکیل داد به قسمی که

$$L(M)=L(G)$$

اثبات: فرض کنیم M یک FSA باشد که طبق روش گفته شده از روی گرامر تشکیل شده است. اثبات اینکه

$$L(M)=L(G)$$

را می توان بطور مشابه با قضیه قبل صورت داد.

نتیجه مهم: قدرت گرامرهای منظم و ماشین FSA به یک اندازه است.

تبدیل گرامر Right-linear به Left-linear:

If G has Then G' has

$$\begin{array}{ll}
 \Sigma \rightarrow SA & A \rightarrow S \\
 A \rightarrow SB & S \rightarrow As \\
 A \rightarrow S & \Sigma \rightarrow As \\
 \Sigma \rightarrow S & \Sigma \rightarrow S \\
 \Sigma \rightarrow \lambda & \Sigma \rightarrow \lambda
 \end{array}$$

مثال:

$$\begin{array}{llll}
 G: \Sigma \rightarrow \lambda & A \rightarrow aA & G': \Sigma \rightarrow \lambda & A \rightarrow aA \\
 \Sigma \rightarrow aB & B \rightarrow aB & A \rightarrow a & B \rightarrow Aa \\
 \Sigma \rightarrow bA & B \rightarrow bB & B \rightarrow a & B \rightarrow Bb \\
 \Sigma \rightarrow bB & A \rightarrow a & B \rightarrow b & \Sigma \rightarrow Aa \\
 \Sigma \rightarrow b & B \rightarrow b & \Sigma \rightarrow a & \Sigma \rightarrow Bb \\
 \Sigma \rightarrow b & & \Sigma \rightarrow b &
 \end{array}$$

نتیجه: جملات زیر معادل هستند:

۱. زبان L با یک FSA تشخیص داده می شود.

۲. زبان L با یک گرامر Right-Linear تولید می شود.

۳. زبان L با یک گرامر Left-Linear تولید می شود.

عبارات منظم و FSA:

هدف: برای یک FSA می توان یک عبارت منظم تشکیل داد که نشاندهنده زبانی است که FSA تشخیص می دهد و بر عکس.

عبارت منظم

تعریف: فرض کنیم V یک الفبای محدود باشد. یک عبارت منظم روی V هر رشته محدودی از سمبل های مجموعه $\{U, *, (), \lambda, \phi\}$ می باشد که می تواند بر اساس قواعد زیر تشکیل شود.

مثال: $V = \{0,1\} \Leftarrow 1U0^*$ یک عبارت منظم است.

رشته ای است شامل صفر سمبل.

تهی مجموعه ای است شامل هیچ عضو.

۱. λ یک عبارت منظم است.

۲. ϕ یک عبارت منظم است.

۳. اگر $a \in V$ باشد آنگاه a یک عبارت منظم است.

۴. اگر α, β عبارات منظم باشند عبارات زیر منظم هستند.

$$4.a \quad (\alpha\beta)$$

$$4.b \quad (\alpha \cup \beta)$$

$$4.c \quad (\alpha^*)$$

مثال: اگر $V = \{a,b\}$ آیا $\underset{\alpha}{a}^* \cup \underset{\beta}{abb}^*$ عبارت منظم است؟ بله

α منظم است $\Rightarrow a^*$ منظم است \Rightarrow چون a منظم است.

چون b, a منظم
هستند.
چون b منظم

$\left. \begin{array}{l} ab \\ b^* \end{array} \right\} \Rightarrow \mu$ مصم است $\Rightarrow abb^*$ منظم است \Rightarrow منظم $(\alpha \cup \beta)$ است.

$$1. (\alpha^*)^* = \alpha^*$$

$$2. \alpha\alpha^* = \alpha^*\alpha$$

$$3. \alpha\alpha^* \cup \lambda = \alpha^*$$

$$4. \alpha(\beta \cup \varnothing) = \alpha\beta \cup \alpha\varnothing$$

$$5. \alpha(\beta\alpha)^* = (\alpha\beta)^*\alpha$$

$$6. (\alpha \cup \beta)^* = (\alpha^* \cup \beta^*)^*$$

$$7. (\alpha \cup \beta)^* = (\alpha^* \beta^*)^*$$

$$8. (\alpha \cup \beta)^* = \alpha^*(\beta\alpha^*)^*$$

وارون یک رشته: \varnothing^R را وارون رشته \varnothing گویند.

$$1. \text{ if } \varnothing = \begin{cases} \lambda \\ \varnothing \\ a \end{cases} \text{ then } \varnothing = \varnothing^R$$

$$2. \text{ if } \varnothing = (\alpha\beta) \text{ then } \varnothing^R = (\beta^R\alpha^R)$$

$$3. \text{ if } \varnothing = (\alpha \cup \beta) \text{ then } \varnothing^R = (\alpha^R \cup \beta^R)$$

$$4. \text{ if } \varnothing = \alpha^* \text{ then } \varnothing^R = \alpha^{R*}$$

حل مجموعه معادلات سیستم

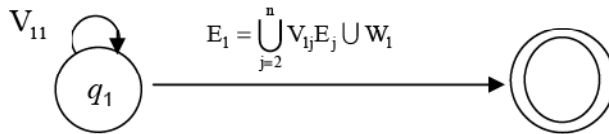
هدف: تشکیل عبارت منظمی برای یک FSA می باشد که تشریح کنند رفتار ماشین است (یعنی رشته هائی که ماشین پذیرش می کند).

$$E(q) = \{W \mid \text{است } q \text{ حالت شروع از شروع با ماشین}\}$$

$$E_K = \bigcup_{j=1}^n V_{kj} E_j \cup W_K \quad K=1,2,\dots,n$$

$$E_k = \text{end set } q_k \quad V_{kj} = \{s \in S \mid q_k \xrightarrow{s} q_j\}$$

$$W_k = \begin{cases} \lambda, & q_k \in F \\ \varnothing, & \text{other wise} \end{cases}$$



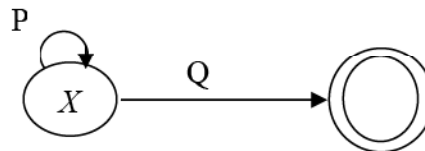
$$E_1 = \bigcup_{j=1}^n V_{1j} E_j \cup W_1$$

$$E_1 = V_{11} E_1 \cup \bigcup_{j=2}^n V_{1j} E_j \cup W_1$$

$$X = PX \cup Q$$

⇓

$$X = P^*Q$$

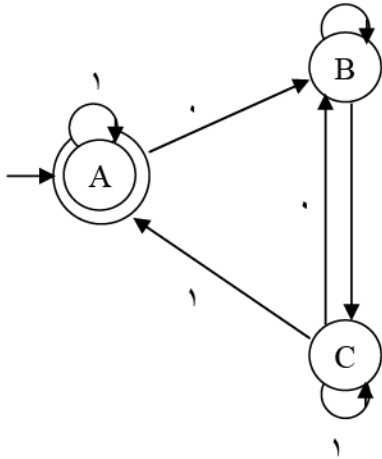


$$E_1 = V_{11}^* \left(\bigcup_{j=2}^n V_{1j} E_j \cup W_1 \right)$$

$$E_K = \bigcup_{j=2}^n \overbrace{(V_{k1} V_{11}^* V_{1j} \cup V_{kj})}^{Const} \overbrace{\tilde{E}_j}^{Variable} \cup \overbrace{V_{k1} V_{11}^* W_1 \cup W_k}^{Const}$$

$$K=2, \dots, n$$

مثال:



$$A = 1A \cup 0B \cup \lambda$$

$$B = 1B \cup 1C$$

$$C = 1A \cup 0B \cup 1C$$

$$L(M) = A$$

$$C = \overbrace{1A \cup 0B}^Q \cup \underbrace{1C}_P$$

$$C = 1^*(1A \cup 0B)$$

$$B = 1B \cup 1(1^*(1A \cup 0B))$$

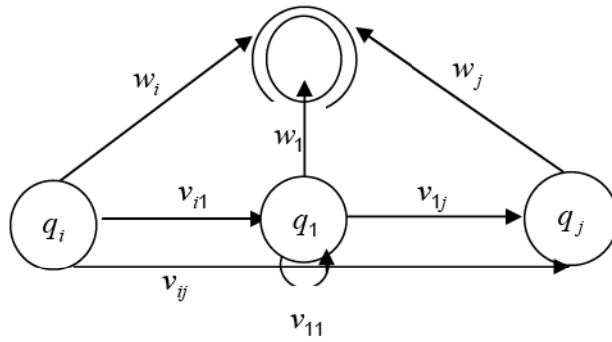
$$= (1 \cup 11^*0)B \cup 11^*1A$$

$$B = (1 \cup 11^*0)^* 11^*1A$$

$$A = 1A \cup 0(1 \cup 11^*0)^* 11^*1A \cup \lambda$$

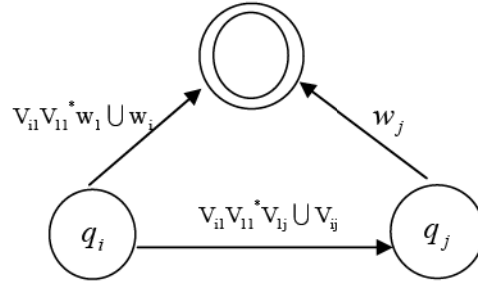
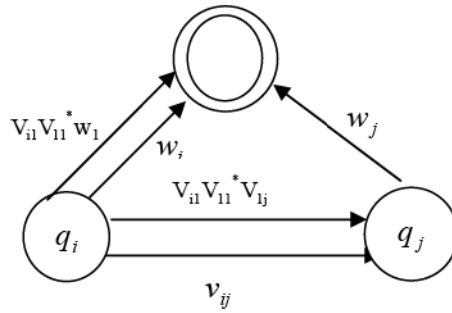
$$A = \underbrace{(1 \cup 0(1 \cup 11^*0)^* 11^*)}_P 1A \cup \underbrace{\lambda}_Q$$

$$A = (1 \cup 0(1 \cup 11^*0)^* 11^*)^* \lambda = L(M)$$



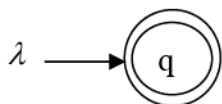
$$V_{i1}V_{11}^*w_1$$

۶

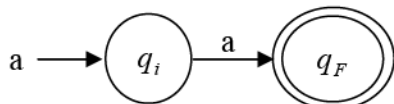


قضیه : هر FSA زبانی را تشخیص می دهد که می تواند با یک عبارت منظم تشریح گردد.

تشکیل پذیرنده برای یک عبارت منظم:



ماشینی که فقط یک حالت (شروع و پایان) دارد پذیرنده عبارت λ است.

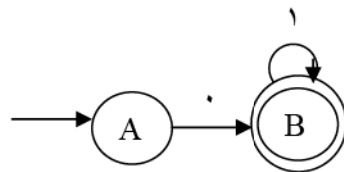


می‌خواهیم برای عبارات پیچیده نظیر $(a \cup b)^*(a^*b^*c^*)^*$ ماشین تشکیل دهیم.

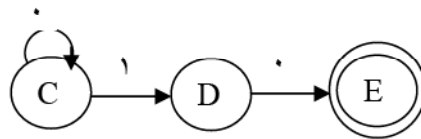
در حالت کلی باید برای عبارتی مانند $R_1^*, R_1R_2, R_1 \cup R_2$ پذیرنده تشکیل دهیم.

مثال: فرض کنیم می‌خواهیم ماشینی که چنین رشته‌هایی را بپذیرد تشکیل بدهیم: $R_1 = 01^*$

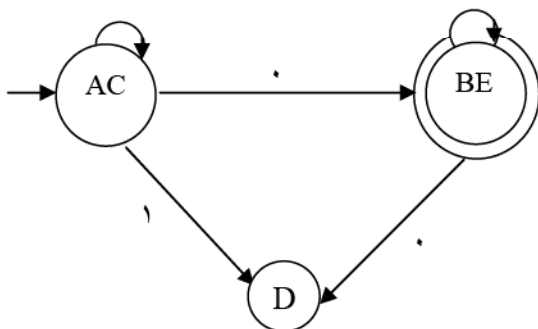
در حقیقت می‌خواهیم ماشین $R_1 \cup R_2$ را تشکیل بدهیم. $R_2 = 0^*10$



R_1 :



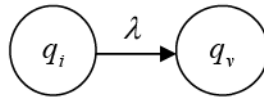
R_2 :



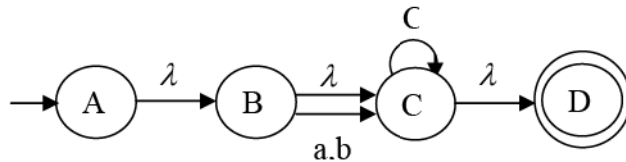
$$0^*(0 \cup 10)1^* = 0^*01^* \cup 0^*101^*$$

$$R_1 \cup R_2 = 01^* \cup 0^*10$$

Transition - λ : یعنی بدون دریافت ورودی از حالت q_i به حالت q_v برسیم.



پذیرنده‌های با انتقال روی λ :



انتقال روی λ ، قطعیت ماشین را خیلی کمتر می‌کند یعنی تصمیماتی که گرفته می‌شوند خیلی پیچیده‌تر از ماشین‌های قبلی است.

تعریف: یک λ - acceptor FSA است بفرم $M = (Q, S, P, q_I, q_F)$ که دارای Transition - λ بوده به قسمی که:

۱. M فقط یک حالت شروع یعنی q_i را دارد بطوریکه انتقالی به q_i وجود ندارد.

۲. M فقط یک حالت شروع یعنی q_F را دارد بطوریکه انتقالی به q_F وجود ندارد.

حالت شروع و حالت پایانی ماشین نمی‌تواند یکی باشد چون دو شرط دیگر برقرار نخواهد بود.

تبدیل FSA به λ - acceptor:

۱. فرض کنیم M FSA بفرم $M = (Q, S, P, I, F)$ باشد می‌خواهیم ماشین λ - acceptor مثل

$M' = (Q', S, P', q_I, q_F)$ را بسازیم بقسمی که $L(M') = L(M)$ فرض کنیم:

$$Q' = Q \cup \{q_I, q_F\}$$

۲. انتقال M' شامل تمامی انتقال M و علاوه انتقالاتی بفرم زیر می‌باشد:

$$q_I \xrightarrow{\lambda} q \quad q \in I, \quad q' \xrightarrow{\lambda} q_F \quad q' \in F$$

اگر بتوانیم ماشین λ -acceptor را به FSA تبدیل کنیم ثابت می شود که قدرت این دو ماشین با هم برابر است.

تبدیل λ -acceptor به FSA

بردار

اگر یک Loop با انتقال λ داشته باشیم، اگر بجای تام حالات فقط یک حالت را در نظر بگیریم رفتار ماشین هیچ تغییری نمی کند در حقیقت ماشین باز هم یک λ -acceptor است اما بدون حلقه ای از λ .

مراحل تبدیل λ -acceptor به FSA:

۱. فرض کنید λ -acceptor مثل $M' = (Q', S, P', q_I, q_F)$ شامل یک λ -Loop باشد، فرض کنیم $Q_L \subseteq Q$ حالات Loop باشد. λ -acceptor $M' = (Q', S, P', q_I, q_F)$ را تشکیل می دهیم بصورت زیر:

$$Q' = (Q - Q_L) \cup \{q_L\} \text{ a}$$

b. اگر M دارای انتقالی بفرم $q_1 \xrightarrow{s} q_2$ که $S \in \{S \cup \{\lambda\}\}$ باشد. آنگاه در M' انتقالی بفرم زیر قرار می دهیم:

$$q_1' \xrightarrow{s} q_2$$

بطوریکه:

$$q_2' = \begin{cases} q_L & \text{اگر } q_2 \in Q_L \\ q_2 & \text{other wise} \end{cases} \quad \text{و} \quad q_1' = \begin{cases} q_L & \text{اگر } q_1 \in Q_L \\ q_1 & \text{other wise} \end{cases}$$

با حذف Loop ها نمی توان گفت "حتماً می توان به یک FSA رسید."

بردار

۲. فرض کنیم $M' = (Q', S, P', q_I, q_F)$ یک λ -acceptor باشد. با این خصوصیات که در آن λ -Loop وجود نداشته باشد FSA، $M' = (Q', S, P, I, F')$ را بصورت زیر تشکیل می‌دهیم:

$$a. Q' = Q - \{q_F\}$$

$$b. F' = \left\{ q \in Q \mid q \xrightarrow{\lambda} q_F \right\}$$

۳. انتقالات M' بصورت زیر می‌باشد:

a. هر انتقال $q \xrightarrow{s} q'$ که در ماشین M باشد در M' نیز قرار می‌دهیم.

b. انتقال $q'' \xrightarrow{s} q'$ را در M' قرار می‌دهیم در حالتیکه داشته باشیم:

$$q'' \xrightarrow{\lambda} q \xrightarrow{s} q'$$

مثال: می‌خواهیم ماشین λ -acceptor مقابل را به FSA تبدیل کنیم.

بردار

در مرحله اول λ -Loop را حذف می‌کنیم.

بردار

در مرحله دوم ابتدا q_F را حذف می‌کنیم بنابراین حالت‌های q_I و ABC که توسط λ به q_F رسیده‌اند حالت‌های

پایانی می‌شوند.

بردار

نتیجه: هیچ ماشینی با انتقال روی λ نمی توانیم داشته باشیم که برای آن FSA وجود نداشته باشد و برعکس.

ساخت یک پذیرنده برای یک عبارت منظم:

$$R = \begin{cases} R_1 \cup R_2 \\ R_1 R_2 \\ R_1^* \end{cases}$$

بردار $L(M_1) = R_1$

بردار $L(M_2) = R_2$

بردار $L(M_3) = R_1 \cup R_2$

ماشین M_3 از نوع λ - acceptor می باشد.

بردار

$$L(M_4) = R_1.R_2$$

بردار

$$L(M_5) = R_1^*$$

ماشینهای M_4, M_5 نیز از نوع λ - acceptor می باشند.

قضیه: برای هر عبارت منظم مثل α روی الفبای V می توان یک ماشین FSA مثل M ساخت که $L(M)=R$

که در آن R نشان دهنده مجموعه رشته های مشخص شده بوسیله α است.

اثبات: ثابت می کنیم که می توان برای α یک λ - acceptor تشکیل داد.

اثبات با استقرار روی طول $|\alpha|$:

پایه: $|\alpha| = 1$

$$\alpha = \begin{cases} \lambda \\ a \\ \phi \end{cases}, a \in V$$

فرض: فرض میکنیم نحوه تشکیل acceptor λ - برای $1 \leq |\alpha| < K$ را می دانیم.

حکم: می خواهیم نحوه تشکیل acceptor λ - برای $|\alpha| = K$ را پیدا می کنیم.

$$\alpha = \begin{cases} \alpha_1 \cup \alpha_2 & |\alpha_1| < K \\ \alpha_1 \alpha_2 & |\alpha_2| < K \\ \alpha_1^* & \end{cases}$$

$$q_I \xrightarrow{w} q_F$$

$$W \in (\alpha_1 \cup \alpha_2)$$

$$q_{I1} \xrightarrow{w} q_{F1} \quad W \in \alpha_1$$

$$q_{I2} \xrightarrow{w} q_{F2} \quad W \in \alpha_2$$

مثال: می خواهیم ماشین برای $\alpha = ((b^*a \cup ab^*)c)^*$

مثال: $\alpha = ((ab)^*(ac)^* \cup a)^*$

تعریف: کلاسی از زبانهای مثل l تحت یک عمل یکانی مثل $F: l \rightarrow l$ بسته است اگر برای هر $L = l$, $F(L) \in l$, هم چنین l تحت یک عمل باینری مثل $G: l \times l \rightarrow l$ بسته است اگر برای هر

$$G(L_1, L_2) \in l, L_1, L_2 \in l$$

$$L_1 = a^*b = \{b, ab, aab, \dots\}$$

$$L_2 = ab^* = \{a, ab, aab, \dots\}$$

$$L_1 \cup L_2 = a^*b \cup ab^*$$

$$L_1 \cap L_2 = \{ab\}$$

$$L_1^c = A^* - L_1 = (a/b)^* - a^*b$$

$$L_1 - L_2 = a^*b - ab^* = a^*b - \{ab\}$$

ابهام (Ambiguity)

ابهام در رابطه با زبانهای منظم:

تعریف: گرامری مبهم است که برای یک جمله ورودی چندین اشتقاق داشته باشد.

$$\Sigma \rightarrow A \quad A \rightarrow 1B \quad C \rightarrow 0B$$

$$B \rightarrow 0B \quad C \rightarrow 1C$$

$$B \rightarrow 0C \quad C \rightarrow 1$$

$$I) \Sigma \Rightarrow A \Rightarrow 1B \Rightarrow 10B \Rightarrow 100B \Rightarrow 1000C \Rightarrow 10001$$

$$II) \Sigma \Rightarrow A \Rightarrow 1B \Rightarrow 10C \Rightarrow 100B \Rightarrow 1000C \Rightarrow 10001$$

چون برای جمله ورودی ۱۰۰۰۱ بیش از یک اشتقاق وجود دارد بنابراین گرامر مذکور مبهم است.

تشخیص وجود ابهام:

$$M=(Q,S,P,I,F)$$

برای گرامر یک FSA میسازیم:

$$W = s_1 s_2 \dots s_k ; s_i \in S$$

$$q_0 \xrightarrow{s_1} q_1 \xrightarrow{s_2} q_2 \xrightarrow{s_3} q_3 \xrightarrow{s_4} \dots \xrightarrow{s_k} q_k \quad , q_0 \in I, q_k \in F$$

$$W = \gamma\psi \quad \text{فرض کنیم}$$

بردار

برای تست اینکه آیا گرامر منظم ابهام دارد یا نه، میتوان ماشین متناظر با آن را ساخت و بدنبال حالاتی مثل

q, q' نشان داده شده در شکل بالا گشت. اگر چنین دو حالتی در ماشین وجود داشته باشند گرامر مبهم است

در غیر اینصورت گرامر مبهم نیست.

پیدا کردن ابهام دو زبانهای منظم:

بردار

$$q \neq q', W = \gamma\psi$$

$$\text{reachable States for } \gamma \quad X[\gamma] = \left\{ q' \mid q \xrightarrow{\gamma} q', q \in I \right\}$$

$$\text{leavable State for } \psi \quad Y[\psi] = \left\{ q' \mid q \xrightarrow{\psi} q', q \in F \right\}$$

$$X[\gamma] \cap Y[\psi] = \{ \quad \}$$

اگر حاصل این اشتراک مجموعه‌ای شامل حداقل ۲ حالت باشد آنگاه ماشین دارای حالاتی مثل q, q' بوده که برای رشته‌ها دو دنباله انتقال را تعریف می‌نمایند و بنابراین گرامر متناظر با این ماشین گرامر مبهمی است.

الگوریتم تشخیص وجود ابهام در گرامر منظمی مثل G :

۱. ماشین FSA مثل M را از روی گرامر G تشکیل می‌دهیم.

۲. نمودار درختی از حالات reachable برای M را با شروع از $X[\lambda] = I$ تشکیل می‌دهیم.

۳. نمودار درختی از حالات Leavable برای M را با شروع از $Y[\lambda] = F$ تشکیل می‌دهیم.

۴. گرامر G مبهم است اگر و فقط اگر بغضی از مجموعه‌ها شامل دو یا چند حالت در هر دو درخت وجود داشته باشند.

۵. هر رشته که مسیری از I به F بار گذر از مجموعه حالات پیدا شده در قدم ۴ حاصل شود رشته‌ای که گرامر G برای آن مبهم است.

مثال: مبهم بودن گرامر مقابل را بررسی کنید.

$$\Sigma \rightarrow A \quad A \rightarrow 1B \quad C \rightarrow 0B$$

$$B \rightarrow 0B \quad C \rightarrow 1C$$

$$B \rightarrow 0C \quad C \rightarrow 1$$

بردار

چون مجموعه $\{B, C\}$ که شامل دو عنصر است در بین reachable Sets و leavable Sets مشترک است پس گرامر G مبهم است.

$$10001$$

$$A \xrightarrow{1} B \xrightarrow{0} B \xrightarrow{0} B \xrightarrow{0} C \xrightarrow{1} D$$

$$A \xrightarrow{1} B \xrightarrow{0} B \xrightarrow{0} B \xrightarrow{0} C \xrightarrow{1} D$$

قضیه: یک روال محدود برای تصمیم گیری در مورد اینکه یک گرامر **right linear** مبهم است، و در صورت وجود ابهام برای یافتن جمله‌ای که برای آن گرامر بصورت ابهام در می‌آید، وجود دارد.

قضیه: برای هر گرامر منظم مثل G که مبهم است، میتوان یک گرامر منظم معادل مثل G' که مبهم نیست تشکیل داد بطوریکه $L(G') = L(G)$.

: Decision Problems

تعریف: با در نظر گرفتن کلاسی از اشیاء (objects) مثل l یک Predicate مثل $P: l \rightarrow \{\text{ture}, \text{false}\}$ **decidable** (قابل تصمیم گیری) نامیده میشود اگر روال قدم به قدمی برای تصمیم گیری در مورد اینکه آیا $P(x)$ برای هر x در l ، **Ture** یا **False** است وجود داشته باشد. چنین پروسه‌ای در صورت وجود یک الگوریتم موثر برای پیشگویی P است.

قضیه: فرض کنیم L_1, L_2 زبانهای منظم (زبانهای حالت محدود) و فرض کنیم G یک گرامر منظم دلخواه باشد. تصمیم گیری در موارد زیر **decidable** است:

۱. آیا $L_1 = L_2$ ؟
۲. آیا $L_1 = \emptyset$ ؟
۳. آیا L_1 محدود یا نامحدود است؟
۴. آیا $L_1 \cap L_2 = \emptyset$ ؟
۵. آیا $L_1 \subseteq L_2$ ؟
۶. آیا G مبهم است؟
- (آیا $L_1 \cap L_1^c = \emptyset$ ؟)

فصل ششم: محدودیتهای Final Automata

در این فصل می‌خواهیم جوابگوی سئوالاتی باشیم از قبیل:

۱. آیا رشته $w \in (0U1)^*$ even parity است؟
۲. آیا رشته $w \in (0U1)^*$ تعداد یکهای بیشتری از صفر دارد؟
۳. آیا x یک مربع کامل است؟
۴. حاصلضرب اعداد y, x چیست؟
۵. حاصلجمع اعداد صحیح y, x چیست؟

تمام سئوالات بالا را می‌توان با سئوال زیر نشان داد:

- آیا رشته w عضوی از مجموعه A می‌باشد؟

تعریف: گوئیم اتوماتای M نمونه‌ای از مسئله P را حل می‌کند، اگر وقتی آن نمونه در اختیارش قرار داده شود، M جواب درست را طی تعداد محدودی قدم تولید کند.

تعریف: کلاسی از اتوماتا مثل M به حد کافی برای مسئله P قدرتمند است اگر یک ماشین مشخص در M وجود داشته باشد که هر نمونه از P را حل کند.

هر ماشینی در کلاس اتوماتای M می‌تواند با رشته‌ای از سمبلها روی یک الفبای محدود تشریح شود.

تشریح عناصر M یک زیر مجموعه از A^* را مشخص می‌کند و بنابراین M حل شود محدود (شمارشپذیر) است.

میدانیم که کلاس تمام زبانهاییکه روی الفبا V ساخته میشود، V^* بوده و غیرقابل شمارش است. بنابراین برای هر کلاس از اتوماتا مسائل عضویتی می‌توان مثال زد که خارج از توان آن کلاس است.

مثال: نشان دهید که $L_m = \{a^k b^k | k \geq 1\}$ یک زبان منظم نیست.

باید ثابت کنیم که هیچ FSAئی وجود ندارد که زبان بالا را بپذیرد.

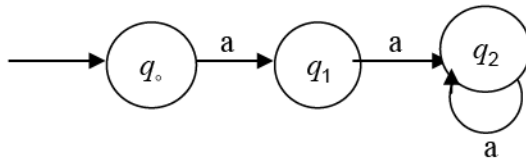
$$G: \Sigma \rightarrow S \quad S \rightarrow aSb$$

$$S \rightarrow ab$$

فرض کنیم L_m منظم باشد و فرض کنیم که M یک FAS که پذیرنده L_m است، باشد.

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} \dots \xrightarrow{a} q_r \xrightarrow{b} q_{r+1} \xrightarrow{b} \dots \xrightarrow{b} q_{2r} = q_F$$

فرض کنیم n تعداد حالات ماشین M باشد و فرض کنیم $r \geq n$. ثابت می‌کنیم که ماشین M علاوه بر پذیرش رشته $a^r b^2$ ، رشته‌های دیگری از مجموعه $a^* b^*$ را پذیرش می‌کند بطوریکه تعداد b, a در آنها مساوی نیست.



پس باید تکرار داشته باشیم.

$$E(q) \supseteq (a^p)^* E(q) \Rightarrow a^{r-p} (a^p)^* b^r \in L(M)$$

پس ماشین علاوه بر پذیرش رشته‌هایی که دارای تعداد مساوی b, a هستند، رشته‌هایی از $a^* b^*$ را می‌تواند پذیرش کند که دارای تعداد یکسانی از b, a نمی‌باشد.

$$L_m = \{a^r b^r \mid r < 10\} \rightarrow \text{زبان منظم هست.}$$

$$L_m = \{(ab)^r \mid r > 0\} \rightarrow \text{زبان منظم است زیرا نیازی به حافظه ندارد.}$$

$$L_m = \{a^m b^n \mid m + n \geq 0\} \rightarrow \text{زبان منظم هست.}$$

قضیه: فرض کنیم L یک زبان منظم باشد و فرض کنیم عدد صحیحی مثل $p \geq 0$ وجود دارد به قسمی که

$$X = \{\alpha_1 (\alpha_2)^k \alpha_3 (\alpha_4)^k \alpha_5 \mid k \geq p\}$$

وجود دارد. رشته‌هایی را نشان میدهند و α_4, α_2 غیر تهی

می‌باشند. آنگاه رشته‌هایی مثل $\beta_5, \dots, \beta_2, \beta_1$ وجود دارد. بقسمی که:

$$Y = \beta_1 (\beta_2)^* \beta_3 (\beta_4)^* \beta_5$$

زیر مجموعه‌ای از L است:

$$\beta_1 \in \alpha_1 \alpha_2^*$$

$$\beta_3 \in \alpha_3^* \alpha_4 \alpha_5^*$$

$$\beta_2 \in \alpha_2^* \alpha_4 - \lambda$$

$$\beta_4 \in \alpha_4^* \alpha_4 - \lambda$$

$$\beta_5 \in \alpha^* 4 \alpha_5^*$$

اثبات: فرض کنیم M یک FAS با n حالت باشد و M تشخیص دهنده زبان L باشد. فرض کنیم زبان L شامل مجموعه X تعریف شده در قضیه باشد. عددی مثل $r \geq \max(n, p)$ را در نظر می‌گیریم.

رفتار ماشین برای رشته‌ای مثل $\alpha_1(\alpha_2)^r \alpha_3(\alpha_4)^r \alpha_5$ بصورت زیر است:

$$q_1 = q_0 \xrightarrow{\alpha_1} q_1 \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_2} q_{r+1} \xrightarrow{\alpha_3} q_{r+2} \xrightarrow{\alpha_4} \dots \xrightarrow{\alpha_4} q_{2r+2} \xrightarrow{\alpha_5} q_{2r+3}$$

$$w = \alpha_1 \alpha_2 \dots \alpha_2 \alpha_2 \dots \alpha_2 \alpha_2 \dots \alpha_2 \alpha_3 \alpha_4$$

$$E(q_i) \supseteq \alpha_2^x E(q_i)$$

$$E(q_j) \supseteq \alpha_4^x E(q_j)$$

پس ماشین رشته‌هایی بفرم $\beta_1(\beta_2)^* \beta_3(\beta_4)^* \beta_5$ را پذیرش می‌کند.

$$\beta_1 \in \alpha_1 \alpha^*$$

$$\beta_4 \in \alpha^* 4 - \lambda$$

$$\beta_2 \in \alpha^* 2 - \lambda$$

$$\beta_5 \in \alpha^* 4 \alpha_5^*$$

$$\beta_3 \in \alpha^* 2 \alpha_3 \alpha_4^*$$

مثال: فرض کنیم زبان L_p شامل رشته‌های خوش ترکیب از پراتزها باشد.

$$1. () \in L_p$$

$$2. (w) \in L_p, \text{ if } w \in L_p$$

$$3. \text{if } w, \varphi \in L_p \text{ then } w\varphi \in L_p$$

گرامر context free مقابل زبان L_p را تولید می‌کند.

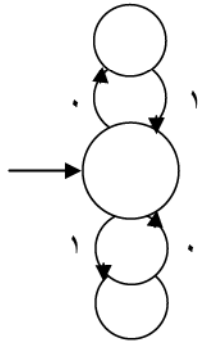
$$G: \Sigma \rightarrow S \quad S \rightarrow ()$$

$$S \rightarrow (S)$$

$$S \rightarrow SS$$

تعریف: یک FSG یک پنج تائی بفرم $M=(Q,R,P,I,F)$ است که در آن R نشان دهنده الفبای خروجی است

یک FSG وقتی محدود است که از هر حالت آن فقط یک Transition خارج شده باشد.



این FSG رشته‌هایی بفرم $(01U10)^*$ را تولید می‌کند.

اگر نشان دهنده یک تولید کننده باشد آنگاه رشته‌هایی بفرم $(01U10)^*$ را تولید می‌کند.

اگر نشان دهنده یک پذیرنده باشد آنگاه رشته‌هایی بفرم $(01U10)^*$ را پذیرش می‌کند.

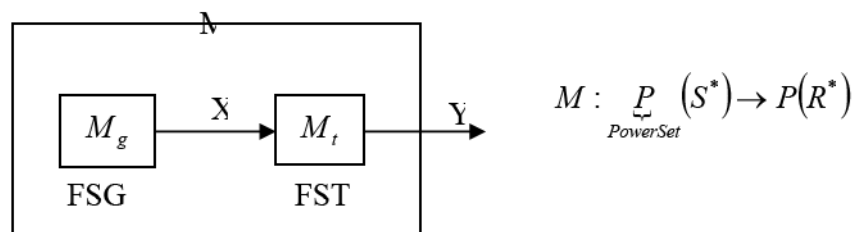
قضیه: یک زبان منظم است اگر فقط اگر بوسیله یک FSG تولید شود.

محدودیت‌های مترجم‌های حالت محدود:

تعریف: فرض کنیم $M = (Q, S, R, f, g, q_f)$ یک مترجم معین باشد. اگر X هر مجموعه از رشته‌های ورودی

باشد، ترجمه X بوسیله M $\{ \varphi \text{ پاسخ ماشین } M \text{ در برابر تحریک } W \in X \text{ باشد} \mid \varphi \in R^* \}$

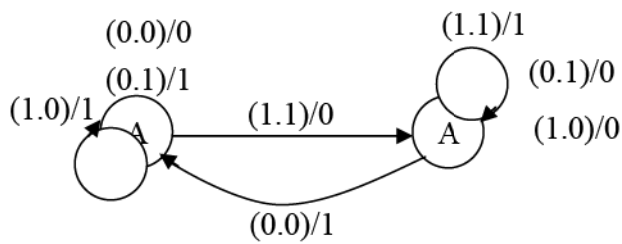
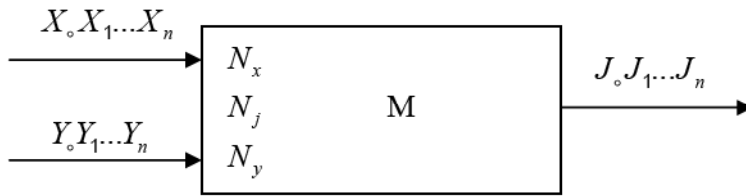
فاز Lexical در کامپایلر که توسط Scanner انجام می‌شود یک FST است. (مترجم).



ماشین M که ترکیبی از یک FST و یک FSG است خود یک FSG است.

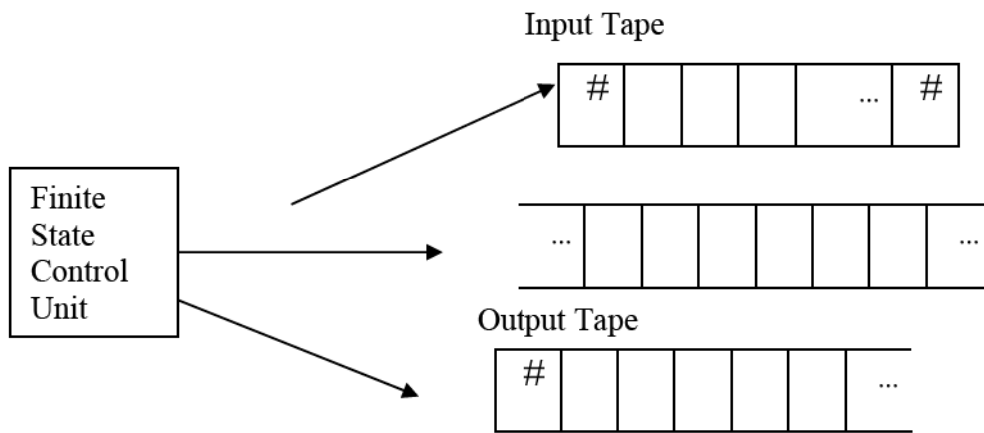
قضیه: ترجمه یک مجموعه منظم بوسیله یک FST یک زبان منظم است. یعنی اینکه کلاس مجموعه‌های منظم تحت عمل ترجمه به یک FST بسته است.

- ماشین M را که کار آن جمع زدن است بصورت زیر تشکیل میدهیم:



فصل هفتم

Tape Automata



مدل فوق یک T.A. را نشان میدهد. فرق اساسی این نوع ماشین با FSM، داشتن حافظه - STORAGE است. قسمت کنترل خودش یک FSM است که کار تصمیم‌گیری را انجام میدهد.

- ابتدا و انتهای Input Tape با سمبل # مشخص می‌شود. بنابراین تعداد سلول‌های نوار ورودی محدود است. از طرف دیگر طول نوار خروجی میتواند نامحدود باشد. البته ابتدای آن با # مشخص میگردد. طول Storage Tape نیز می‌تواند نامحدود باشد.

- حرکت نوارها در هر لحظه فقط به اندازه یک سلول میباشد.

عملیاتی که واحد کنترل انجام میدهد:

۱. حرکت head یکی به سمت چپ یا راست.

۲. خواندن یا نوشتن سمبلی از روی Tape.

۳. انتقال به یک حالت جدید.

× رفتار واحد کنترل بصورت روبرو بیان می‌شود: $more(t, q')$

- q' حالت جدیدی است که از q به آن منتقل می‌شویم.

- t یک سمبل الفبا می‌باشد.

- Move یکی از سه عمل ذکر شده بالا میباشد.

ویژگیهای یک Tape Automata :

الف- مشخصات ساختاری

۱. عمل ماشینی

- Transducer (tape ورودی و خروجی)

- Acceptor (فقط tape ورودی)

- Generator (فقط tape خروجی)

۲. وجود یا عدم وجود Storage Tape

۳. الفبای tape

ب- مشخصات رفتاری

۱. نوع عمل

- معین

- نامعین

۲. حرکت head نوار ورودی

- یک طرفه (از چپ بر راست)

- دو طرفه (از هر جهت)

۳. محدودیتهای دسترسی به نوار حافظه

- Last in First out (Push down storage)

- دسترسی دلخواه در یک ناحیه با طول محدود (Linear bowed)

- دسترسی دلخواه بصورت نامحدودی

Type of grammar	Type of Acceptor	characteristics
3 (Regular)	Finite – State	Storage tape ندارد
2 (Context Free)	Push down	یک نوار حافظه pushdown عمل بصورت نامعین
1 (Context Sensitive)	Linear bound	یک نوار Linear bowed عمل بصورت نامعین
0 (Irregular)	Turing Machine	یک نوار حافظه بصورت دسترسی دلخواه نامحدود

ویژگیهای ترتیبی عمومی (Generalized Sequential Machines)

تعریف: یک

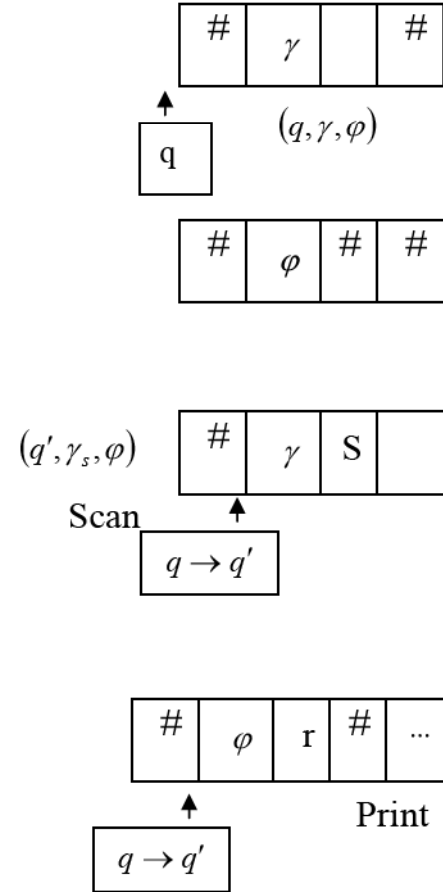
$$\text{GST} = M(Q, S, R, P, I, F) \quad \text{P برنامه}$$

Instructionها:

$$\left. \begin{array}{l} q] \text{Scan}(s, q') \\ q] \text{Print}(r, q') \end{array} \right\} \begin{array}{l} q, q' \in Q \\ s \in S \\ r \in R \end{array}$$

Machine configuration (بیکربندی ماشین):

۱. حالت واحد کنترل
۲. رشته‌ای از سمبلها که از نوار ورودی پیمایش کرده
۳. رشته‌ای از سمبلها که بر روی نوار خروجی نوشته



تعریف: فرض کنیم M یک GST با برنامه P باشد و رشته $w \in S^*$ روی نوار ورودی قرار گرفته باشد.

عملیات از برنامه P در یک $Configuration$ از M قابل اعمال (applicable) هستند به شرطی که:

۱. یک عمل $qJScan(s, q')$ وقتی قابل اعمال است که ماشین در $Configuration$ ای مثل (q, γ, φ) باشد و

γ_s یک پیشوند w باشد.

با اعمال این عمل M نوار ورودی را یکی به سمت راست انتقال داده، سمبل S قرار گرفته روی آنرا ملاحظه

کرده و بحالت q' منتقل می شود.

$$(q, \gamma, \varphi) \xrightarrow{S} (q', \gamma_s, \varphi)$$

۲. یک عمل $qJPrint(r, q')$ در هر Configuration قابل اعمال است. با اعمال این عمل M هد خروجی

را یکی بسمت راست منتقل کرده سمبل r را نوشته و وارد حالت q' می شود.

$$(q, \gamma, \varphi) \xrightarrow{P} (q', \gamma, \varphi_r)$$

$$(q_0, \gamma_0, \varphi_0) \rightarrow (q_1, \gamma_1, \varphi_1) \rightarrow \dots (q_k, \gamma_k, \varphi_k)$$

$$(q_0, \gamma_0, \varphi_0) \Rightarrow (q_k, \gamma_k, \varphi_k)$$

رفتار ماشین غیر معین است. x اگر در هر Conf. فقط یک Inst قابل اعمال باشد معین است.

$$\left(\begin{array}{c} q \\ \hline q \in I \end{array} \right)$$

Conf ابتدائی

$$\left(\begin{array}{c} q' \\ \hline q' \in F \end{array} \right)$$

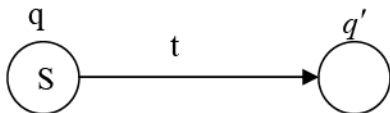
$$(q, \lambda, \lambda) \Rightarrow (q', \gamma, \varphi)$$

$$q \in I, q' \in F$$

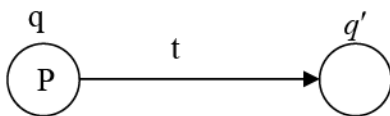
گوئیم φ پاسخ ماشین M به تحریک γ می باشد.

اگر $X \subset S^*$ ، آنگاه $\{ \varphi \text{ پاسخ ماشین M به تحریک است } \gamma \in X \mid \varphi \in R^* \}$ ترجمه M برای X است.

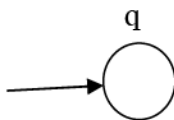
اگر $Y \subset R^*$ ، آنگاه $\{ \varphi \in Y \mid \varphi \text{ پاسخ ماشین M به تحریک می باشد } \}$ $X = \{ \varphi \in S^* \mid \varphi \text{ پاسخ ماشین M به تحریک می باشد } \}$



$$q]Scan(t, q')$$



$$q]Print(t, q')$$



$$q \in I$$



$$q \in F$$

مثال:

$$I = \{1\}$$

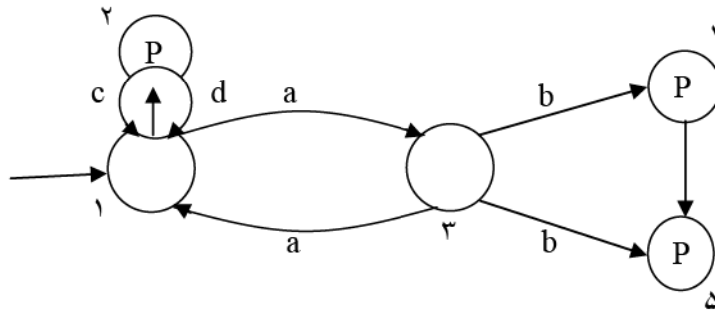
$$F = \{1.3\}$$

$$(1, \lambda, \lambda)$$

$$(1, \gamma, \varphi)$$

Conf ابتدائی

Conf های نهائی



1]Scan(b,2)

1]Scan(a,3)

$S = \{a, b\}$

2]Print(c,1)

$R = \{c, b, d\}$

2]Print(d,1) اگر رشته bab روی Tape ورودی باشد ترجمه آن بصورت زیر خواهد بود.

3]Scan(a,1)

bab

3]Scan(b,4)

$(1, \lambda, \lambda) \xrightarrow{S} (2, b, \lambda) \xrightarrow{P} (1, b, c) \xrightarrow{S} (3, ba, c)$

4]Print(b,5)

$\xrightarrow{S} (4, bab, c) \xrightarrow{P} (5, bab, cb)$

5]Print(b,3)

$\xrightarrow{P} (3, bab, cbb)$

قضیه: کلاس مجموعه‌های منظم تحت عمل ترجمه بوسیله یک GST بسته است.

Two-Way-Accepter

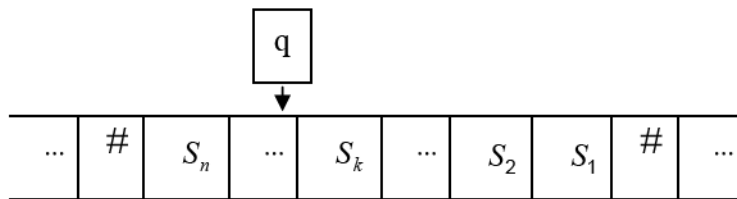
اهداف مطالعه T.W.A

۱. نشان دادن اینکه قدرت ماشین‌های نواردار بدون داشتن نوار حافظه به اندازه FSA است.

۲. نمایش وجود رابطه معکوس بین پیچیدگی محاسبات و زمان لازم برای اتمام محاسبات.

تعریف: یک T.W.A. یک ماشین بفرم $M=(Q,S,P,I,F)$ است.

$$q]left(s, q') \left\{ \begin{array}{l} q, q' \in Q \\ s \in S \cup \{\#\} \end{array} \right.$$



(q,k)

مکان هد نوار ورودی

تعریف: فرض کنیم M یک T.W.A. با رشته ورودی w نوشته شده بر روی نوار ورودی آن باشد.

دستورالعملهایی از M قابل اعمال هستند که شرایط زیر را ارضاء کنند:

۱. یک $Inst$ ، $q]Right(s, q')$ در $Conf$ ، (q,k) قابل اعمال است اگر $k+1$ امین سمبل نوار سمبل s باشد.

$$(q,k) \xrightarrow{R} (q',k+1)$$

۲. یک $Inst$ ، $q]left(s, q')$ در $Conf$ ، (q,k) قابل اعمال است اگر $k+1$ امین سمبل نوار سمبل s باشد.

$$(q,k) \xrightarrow{L} (q',k+1)$$

$$(q,0) \qquad (q',k)$$

$$q \in I \qquad q' \in F$$

موقعی که یک رشته مورد پذیرش ماشین واقع نمی شود:

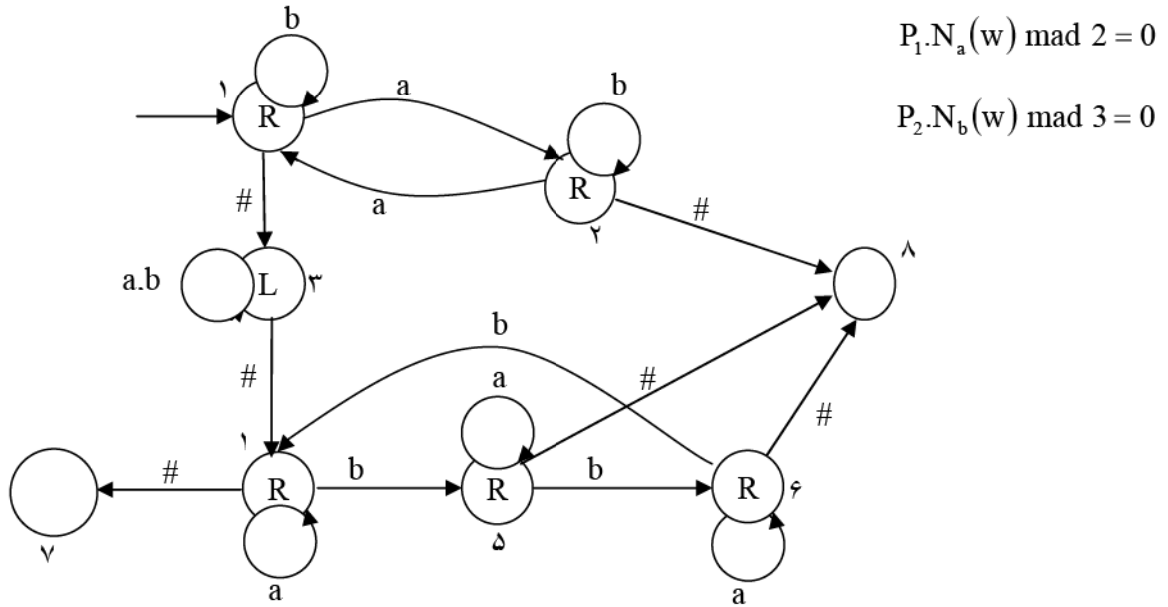
۱. وقتی که ماشین به حالت مرده ($dead\ state$) برسد.

۲. ماشین در حلقه بیافتد. یعنی اینکه یک سیکل از حرکات را بصورت بینهایت بدون رسیدن به یک

حالت پایانی تکرار کند.

۳. ماشین بصورت پایان ناپذیر به سمت راست Scan کند بدون اینکه به یک حالت پایانی برسد.

مثال: T.W.A برای پذیرش رشته‌هایی از $(a \cup b)^*$ بطوریکه هر دو شرط زیر برقرار باشد:



تمرین: FSA معادل ماشین بالا را تشکیل دهید.

مثال: یک F.S.A برای T.W.A بیان شده در مثال قبل پیدا کنید.

حالت	$N_a(w) \bmod 2$	$N_b(w) \bmod 3$
A	0	0
B	0	1
C	1	1
D	1	2
E	0	2
F	1	0

حال فرض می‌کنیم که حالت کلی زیر را برای T.W.A داشته باشیم:

$$P1-N_a(w) \bmod n=0$$

$$P1-N_b(w) \bmod n=0$$

تعداد حالاتی که T.W.A برای این حالت خواهد داشت برابر $n+m+3$ است.

تعداد حالاتی که F.S.A معادل دارد برابر $n.m$ است.

تعداد قدم‌ها در T.W.A برای پذیرش رشته W که $|w|=r$ برابر $3r+3$ است.

تعداد قدم‌ها در F.S.A برای پذیرش رشته W که $|w|=r$ برابر r است.

مثال: فرض کنید می‌خواهیم یک ماشین برای پذیرش $L(r) = \{a^x b^y \mid |x-y| \bmod r = 0\}$ درست کنیم.

یک F.S.A با $2r$ حالت می‌توان برای پذیرش $L(r)$ تشکیا داد.

هیچ F.S.A با کمتر از $2r$ حالت برای $L(r)$ وجود ندارد. (ثابت کنید).

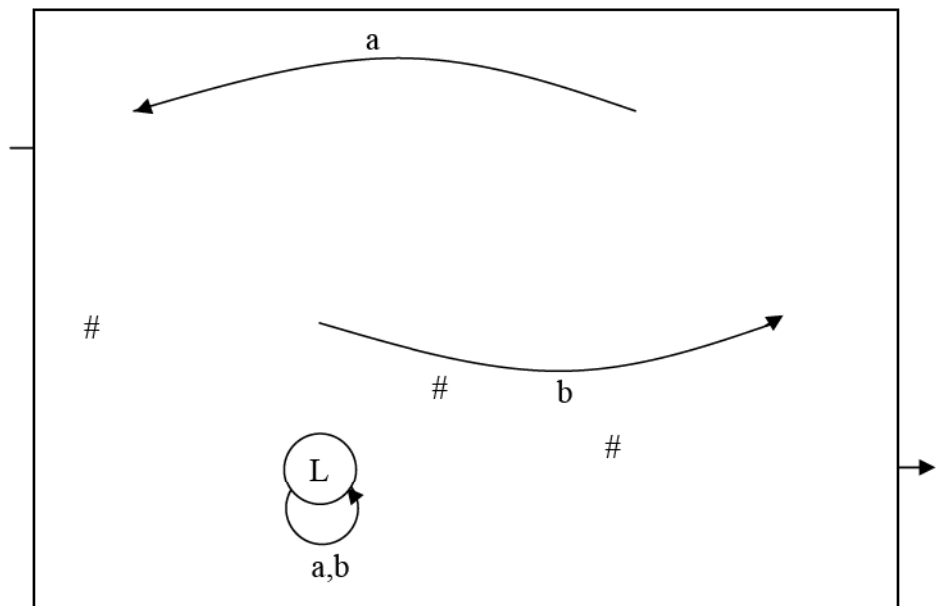
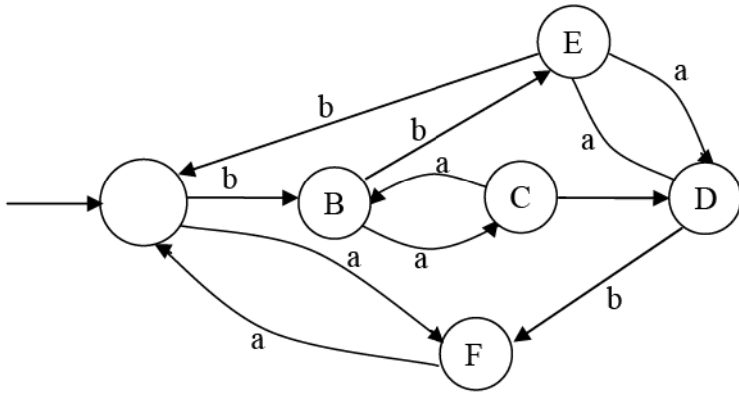
برای مقادیر مشخصی از r می‌توان یک T.W.A با تعداد حالات خیلی کمتر از $2r$ ساخت:

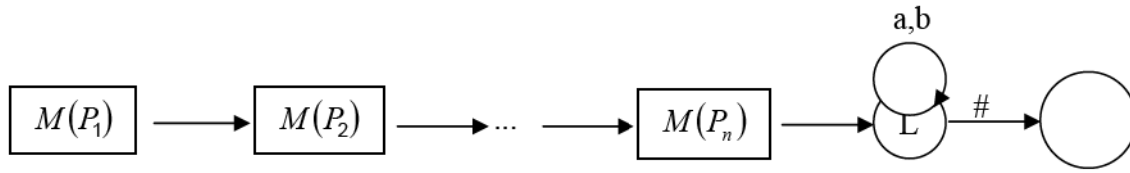
فرض کنیم P_1, P_2, \dots, P_n اولین n عدد اول باشند. ماشینی مثل M می‌سازیم که برنامه آن شامل n روتین مثل

$M(P_1), M(P_2), \dots, M(P_n)$ باشد. هر روتین دو عمل انجام می‌دهد:

۱. یک پیمایش از چپ بر راست که در آن برابری $N_a(w) \bmod P_i$ با $N_b(w) \bmod P_i$ تست می‌شود.

۲. یک پیمایش از راست به چپ که باعث می‌شود head نوار به ابتدا برگردد.





www.salampnu.com

سایت مرجع دانشجوی پیام نور

- ✓ نمونه سوالات پیام نور : بیش از ۱۱۰ هزار نمونه سوال همراه با پاسخنامه
- تستی و تشریحی
- ✓ کتاب ، جزوه و خلاصه دروس
- ✓ برنامه امتحانات
- ✓ منابع و لیست دروس هر ترم
- ✓ دانلود کاملاً رایگان بیش از ۱۴۰ هزار فایل مختص دانشجویان پیام نور

www.salampnu.com