جزوه درس

نظریه زبانها و ماشینها

نام استاد : **علی اصغر پورحاجی کاظم**

مثال:

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

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

$\sum \rightarrow A$
$A \rightarrow aABC$
$A \rightarrow abc$
$CB \rightarrow BC$
$bB \rightarrow bb$
$bC \rightarrow bc$
$cC \rightarrow cc$

$\sum \rightarrow A$
$A \rightarrow aABC$
$A \rightarrow abc$
$CB \rightarrow BC$
$bB \rightarrow bb$
$bC \rightarrow bc$

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

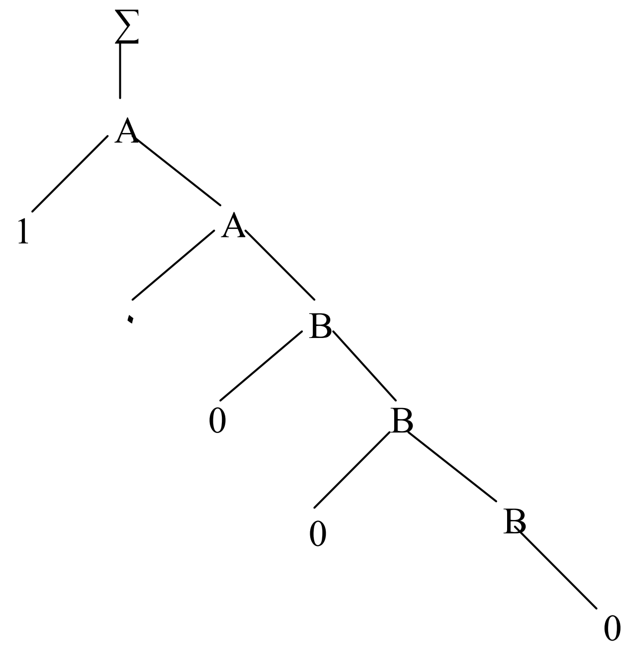
$\sum \rightarrow S$
$S \rightarrow aSb$
$S \rightarrow ab$

	Type	Protection	
Contracting (منقبض شونده)	0	$\varphi A \psi \rightarrow \varphi \omega \psi$	گرامر بدون محدودیت
Non-Contract (غیرمنقبض شونده)	1	$\varphi A \psi \rightarrow \varphi \omega \psi$, $\omega \neq \lambda$ $\sum \rightarrow \lambda$	گرامر های Content Sensitive
	2	$A \rightarrow \omega$, , $\omega \neq \lambda$ $\sum \rightarrow \lambda$	گرامرهای Content Free
	3	$A \rightarrow aB$, $A \rightarrow a$, $\sum \rightarrow \lambda$ } right linear **OR** $A \rightarrow Ba$, $A \rightarrow a$, $\sum \rightarrow \lambda$ } left linear	Regular Grammers

درخت اشتقاق(Derivation Tree):

$\sum \to A$
$A \to 1A$
$A \to 0B$
$B \to 0B$
$B \to 0$

10000

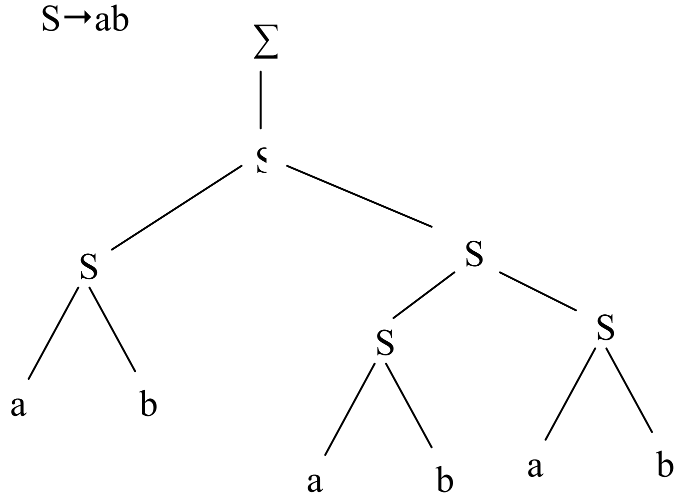


ابهام(Ambiguity):

فرض کنیمG یک گرامرcontent free باشد وفرض کنیم ωجمله ای در(G) Lباشد آنگاه ωمبهم است اگر اشتقاقی از ωوجود داشته باشد که مربوط به درختهای متفاوتی باشند.

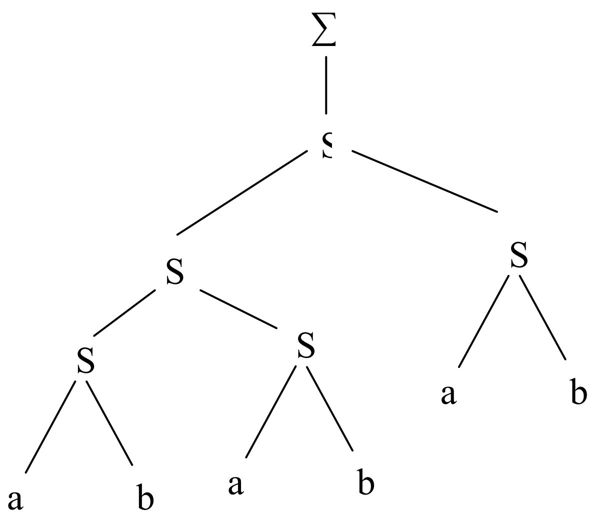
$\sum \to S$
$S \to SS$
$S \to ab$

ababab



1)

2)



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

$\omega_0 \Rightarrow \omega_1 \Rightarrow \ldots \Rightarrow \omega_n$

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

$\omega_{i+1}= \alpha\ \omega\ \beta$　　　$A \in N$

$\beta \in (N \cup T)^*$

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

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

مثال:

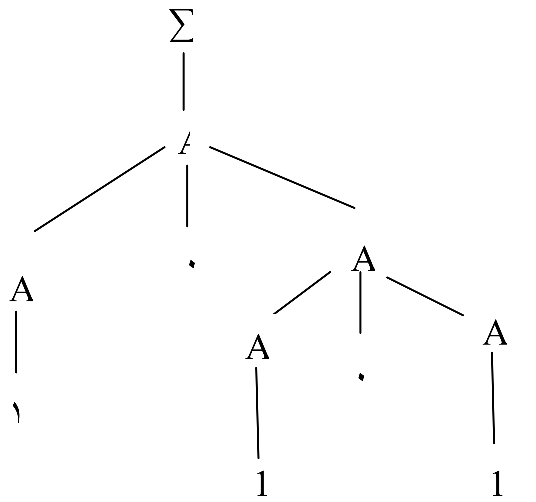
$\sum\rightarrow A$

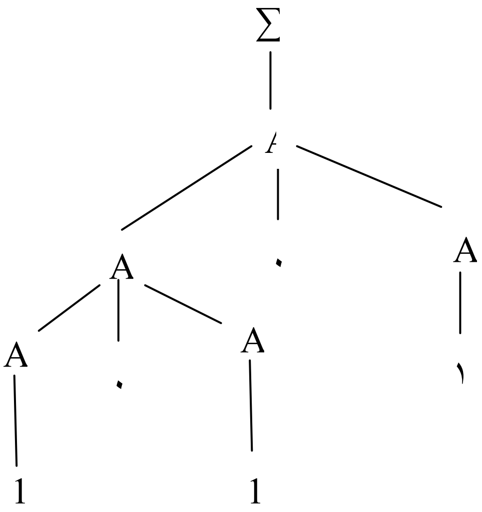
$A\rightarrow A0A$　　　**10101**

$A\rightarrow 1$

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

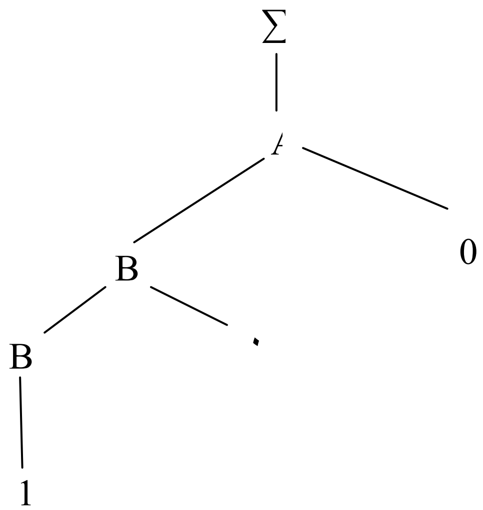
$\sum \Rightarrow A \Rightarrow A0A \Rightarrow 10A \Rightarrow 10A0A \Rightarrow 1010A \Rightarrow 10101$

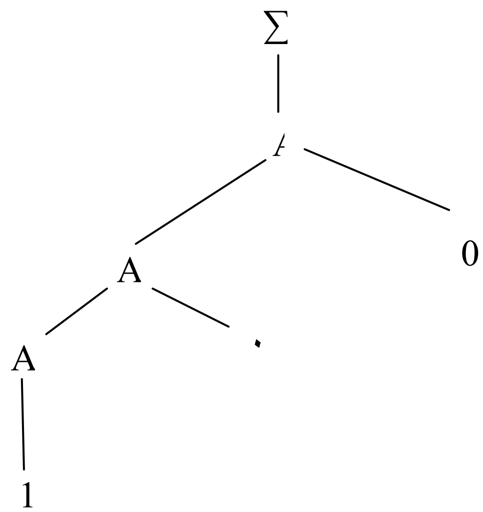


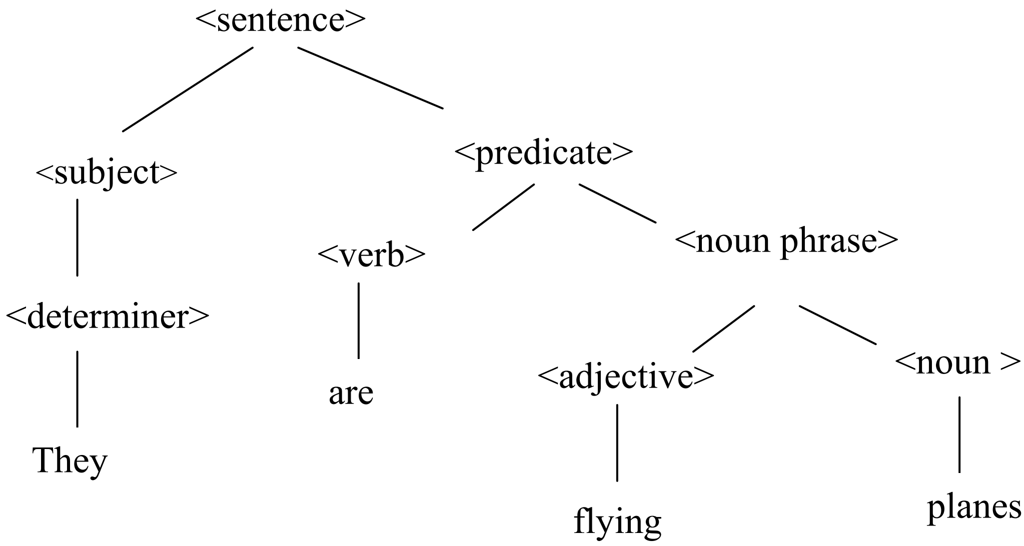


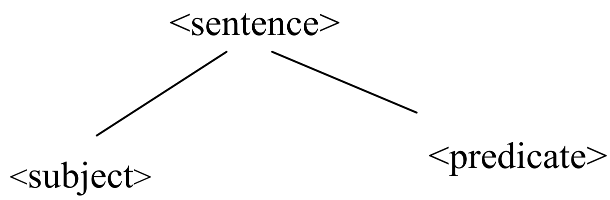
100

G: $\sum \rightarrow A$
$A \rightarrow B0$
$A \rightarrow A0$
$B \rightarrow B0$
$A \rightarrow 1$
$B \rightarrow 1$



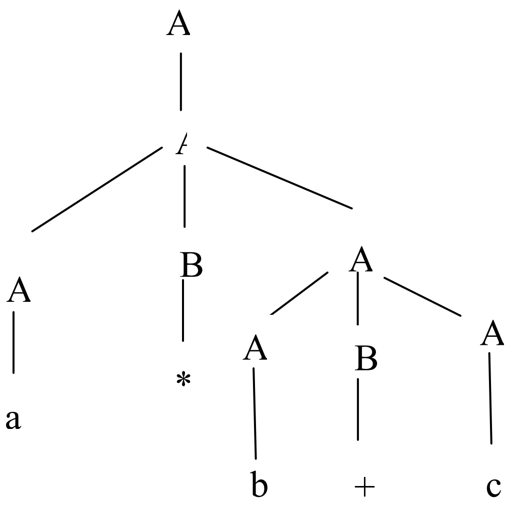


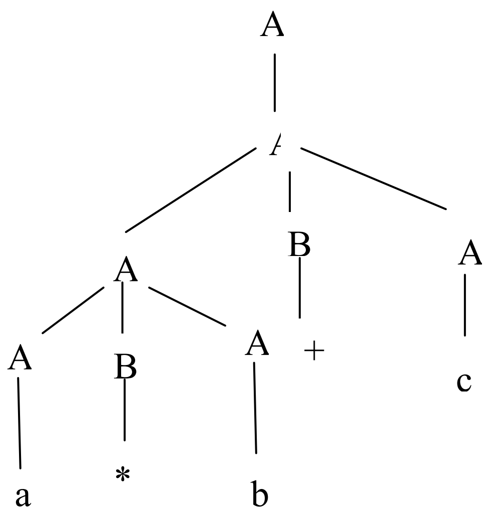




G:A→ABA
A→a
A→b
A→C
B→+
B→*

a*b+c



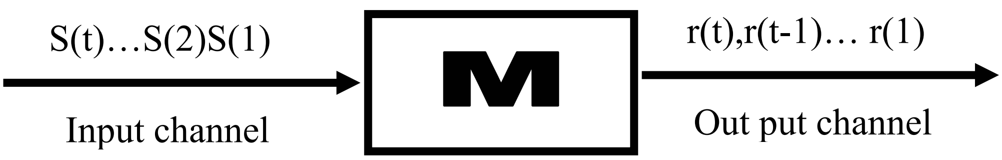


مسائل فصل سوم:3.1,3.2,3.4,3.5,3.6,3.7,3.8,3.9

فصل چهارم

Finite State Machine(ماشین حالت محدود) :

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



N part

$q^{i}(t)$ حالت part، iام در لحظه t باشد

حالت کلی ماشینی M در لحظه t

هر part، k وضعیت دارد

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

ماشینی M، k^n حالت می تواند داشته باشد.

q_i وضعیت اولیه ماشینی را نشان می دهد. Q در لحظه صفر $q(0)=q_i$

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

$Q\in q(t)$

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 Mدر زمانهای $t=0,1,2,\ldots$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 Mبصورت دنباله ای از حالات که هر یک عضوی از qاست نشان داده میشود

۶- یک حالت ابتدایی از Mوجود دارد(q_i)که تشریح کننده وضعیت ابتدایی partهای ماشینی Mقبل از ورود تحریکی به آن است.

Elementهای ریاضی یک ماشین محدود:

۱-مجموعه های محدود S,R,Q

۲-تابع انتقالfکه تعریف کننده حالت بعدی بر حسب حالت فعلی و سمبلهای ورودی بعدی است.

۳- تابع خروجی g که تعریف کننده سمبل خروجی بر حسب حالتهای 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

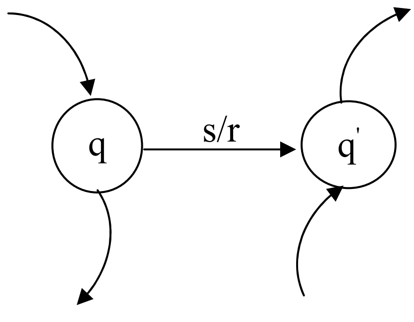
S:مجموعه سمبلهای ورودی

R:مجموعه سمبلهای خروجی

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

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

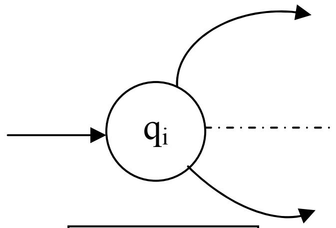
State Diagram



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

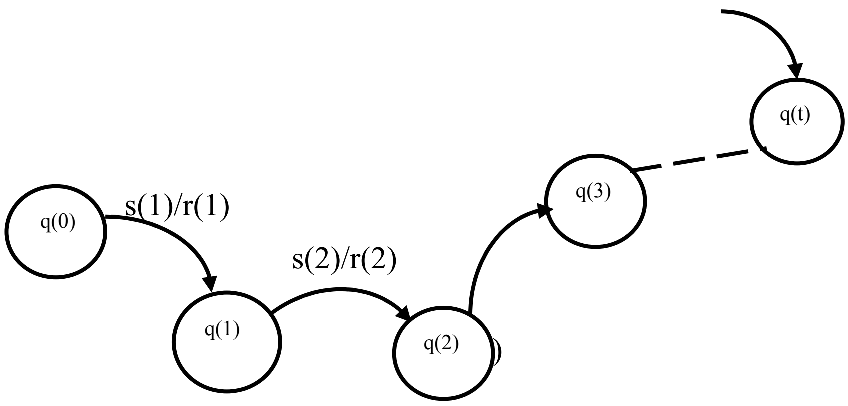
State Table

$q_i\rightarrow 1$	·	·	S		
۲					
.					
.		......		....	
q			$\boxed{q',r}$		
.		....		......	



حالت اولیه

رفتارماشین: با دنباله ای از اتصالات نشان می دهیم.



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

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

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

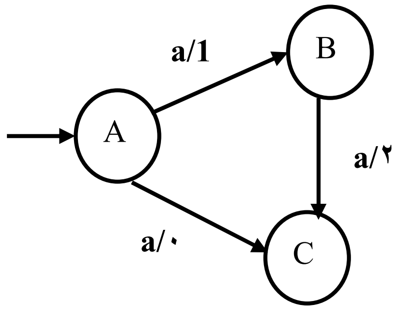
مثال:ماشینی که MOD 3تعداد سمبلهای ورودی را به ما بدهد.

q(t) A means | ω| mod 3=0
B means | ω| mod 3=1
C means | ω| mod 3=2

$s = \{a\}$

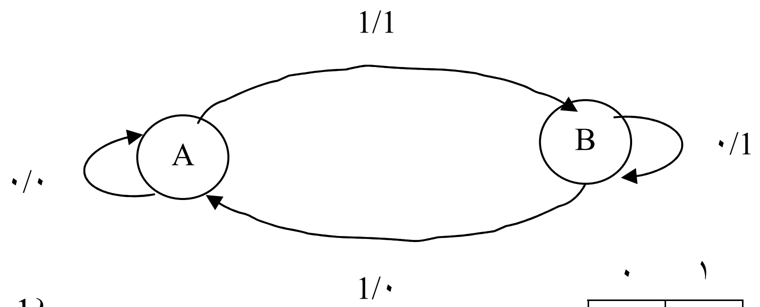
$Q = \{A, B, C\}$

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



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

Parity checker:



	٠	١
A	A,0	B,1
B	B,1	A,0

s= {0, 1}　　, R= {0, 1}

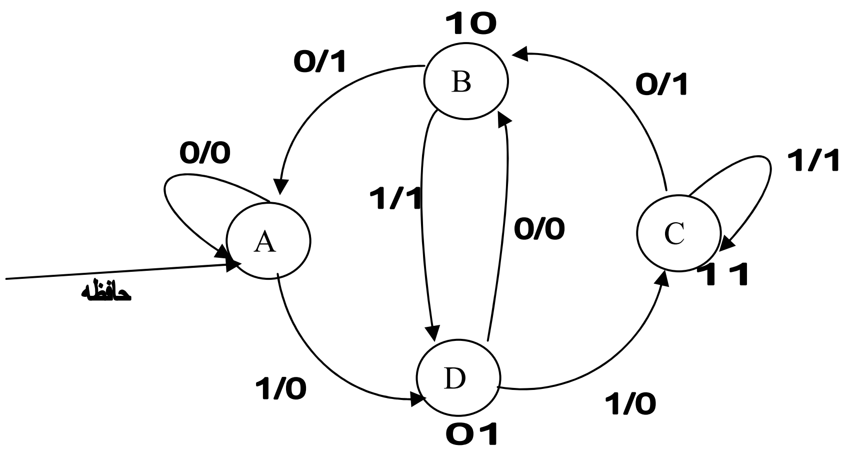
even　　odd

مثال۳: Tωo unit delay

S (t)　　r (t+s) =s (t)

S=R= {0, 1}　　r (1) =0, r (2) =0

S(t-1)	S(t)	State
٠	٠	A
٠	١	B
١	١	C
١	٠	D



$\omega = 0010110$

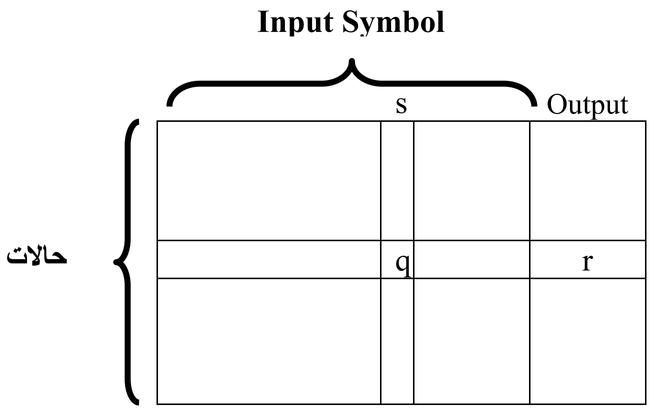
$\varphi = 0000101$

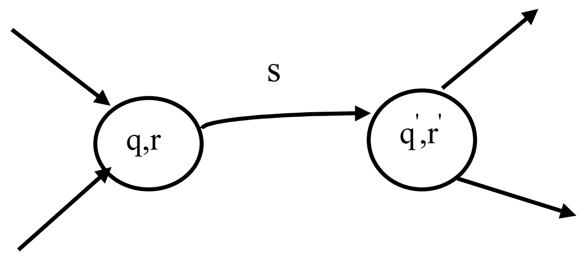
ماشینهای State Assigned Output :

تعریف: یک State assigned output finite state machine یک شش تائی بفرم $M=(Q,S,R,F,H,q_i)$ است که:

$F: Q\times S \rightarrow Q$ تابع انتقال حالت

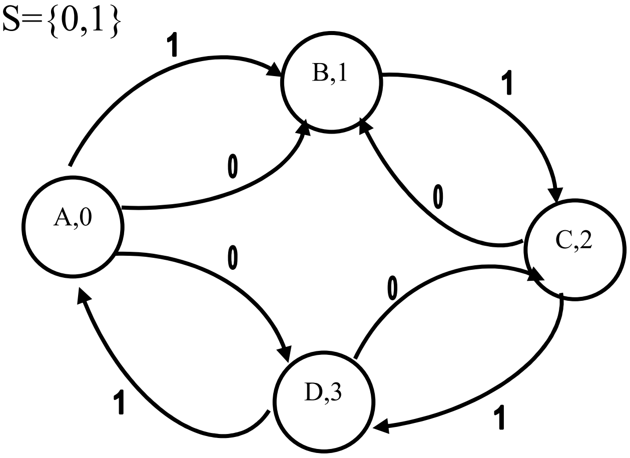
$H: Q \rightarrow R$ تابع خروجی





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

$S=\{0,1\}$



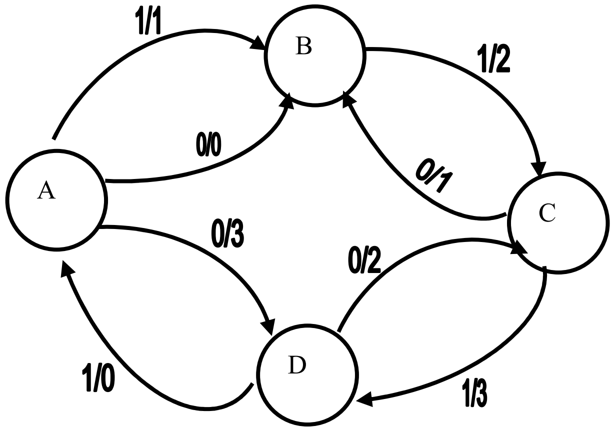
$\omega = s(1)\, s(I) \ldots s(I)$

$N_0(\omega)=\omega$ تعداد صفرهای

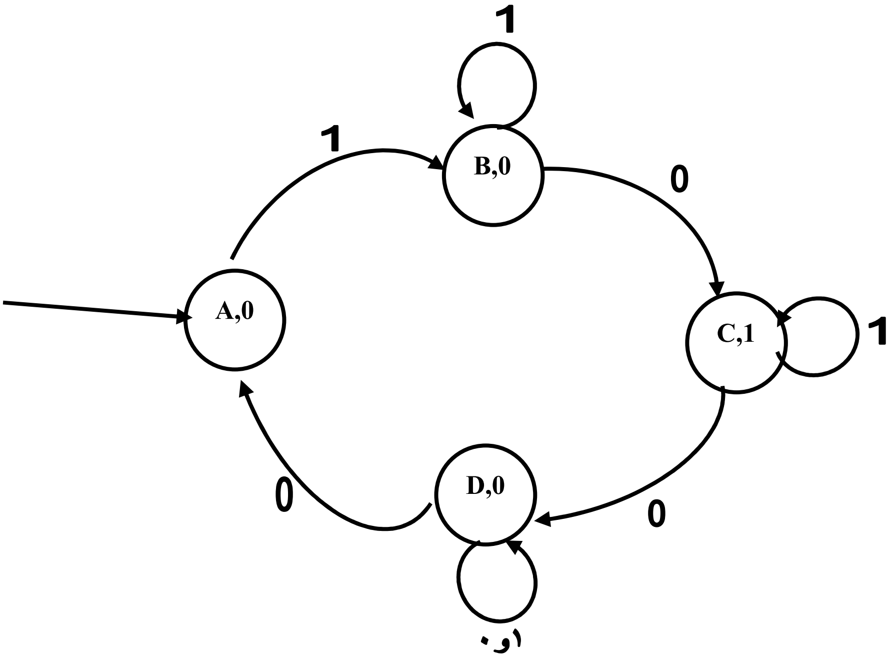
$N_1(\omega)=\omega$ تعداد یکهای

$r(t) = [N_1(\omega)-N_0(\omega)] \bmod 4$

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



مثال:تشخیص دهنده زبان(Language Recognizer):



پیچیدگی ماشین(machine complexity):

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

n قطعه و هر قطعه دو حالت

تعداد کل حالات ماشینی 2^nخواهد بود.

تعداد قطعات $\approx \log 2^{(\text{تعداد حالات ماشینی})}$

بیتی 2^{15} کلمه ۳۲بیتی

تعداد حالات حافظه=2^{2}=10^{300000}

دنبا له های حالات(state sequences):

تعریف:فرض کنیم M یک ماشین محدود با تابع انتقال F:Q×S=Q باشد اگر f(q,s)=q گوییم حالت s-successor حالت q است می نویسیم:

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

اگر رشته ای از سمبلهای ورودی M,ω=s(1)s(2)…s(+) را از حالت q=q(0) به حالت q'=q(+) ببرد:

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

گوئیم حالت q', ω –successor حالت q است و می نویسیم:

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

q (0), q (1),…,q(+) را یک دنباله حالات admissible گویند.

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

قبلا: $f : Q \times S \longrightarrow Q$

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

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

F (q,λ)=q برای هر q ∈ Q

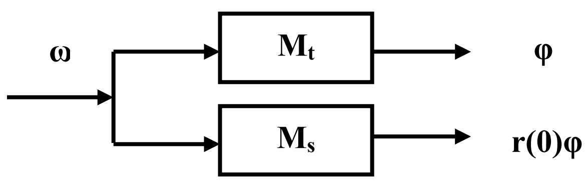
تعریف:فرض کنیمMیک ماشین حالت محدود بوده و فرض کنیم$qi=q(0),q(1),\ldots,q(t)$ یک دنباله حالات کمکی برای $\omega=s(1)\ldots s(t-1)s(t)$ باشد:

الف-اگر Mیک ماشینTransition Assigned با تابع خروجی $g : Q\times S\rightarrow R$ باشد آنگاه $\varphi=r(1)r(2)\ldots r(t)$پاسخ Mبه تحریک ω نامیده میشود که در آن $r(i)=g(q(i-1),s(i))$

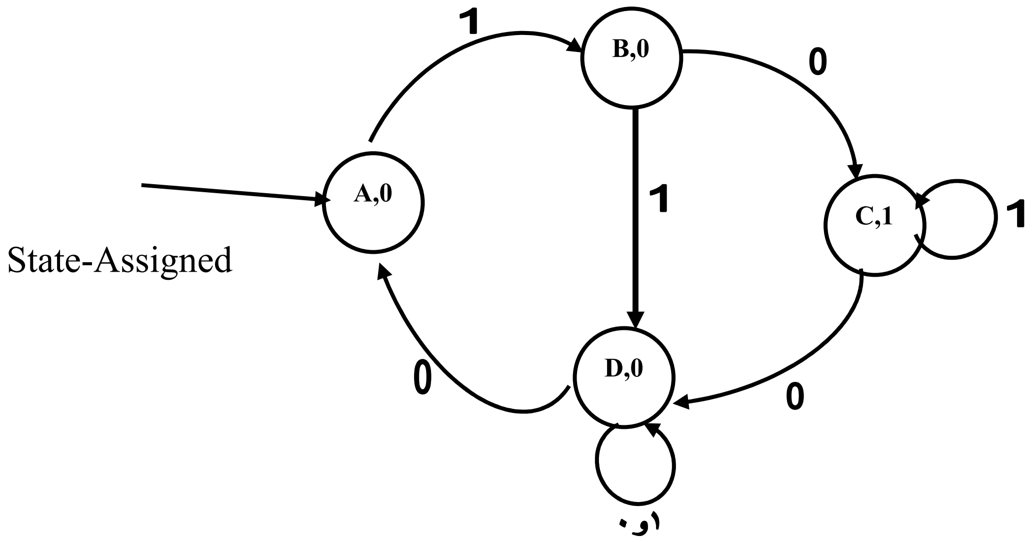
ب- اگرM یک ماشین State-Assigned با تابع خروجی $h : Q\times S\rightarrow R$ باشدسپس $\varphi=r(1)r(2)\ldots r(t)$ پاسخ Mبه تحریک ω نامیده میشود که در آن $r(i)=h(q(i))$

تبدیل ماشینهای Transition Assigned و State-assigned :

تعریف:یک ماشین Transition Assigned مثلMtویک ماشین state-assigned مثل Msمشابه هستند اگر برای هر تحریک پاسخMs دقیقا معادلMt که با یک سمبل اضافی دلخواه ولی ثابت شروع میشود.



قضیه:برای هر ماشینstate-assignedمثل M_s یک ماشین مشابهTransition Assigned مثل M_t وجود دارد و برعکس.

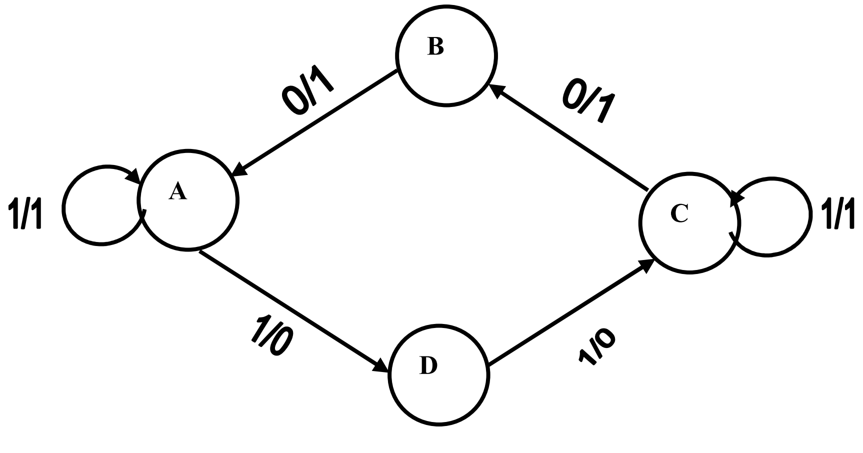


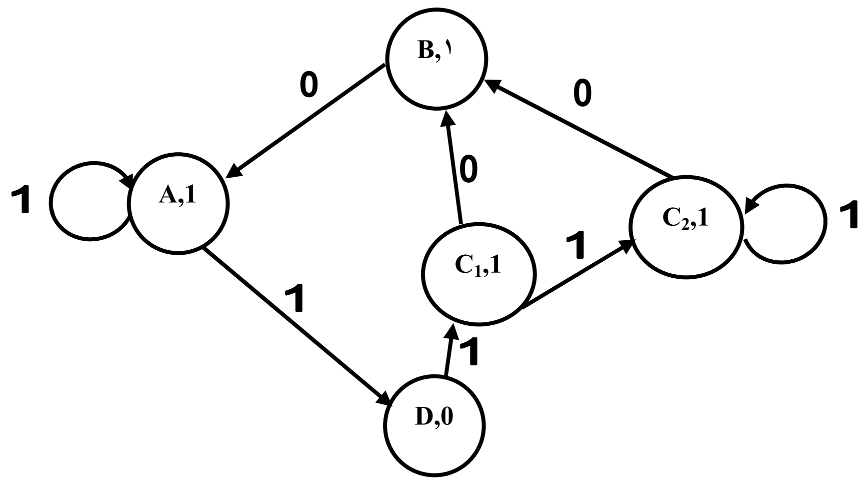
۱–تشکیل ماشین M_t در M_s خروجی r را مشخص کند ،هر انتقال در M_t به q را با r بر چسب گذاری می کنیم.

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 Ms از روی ماشین M_t:

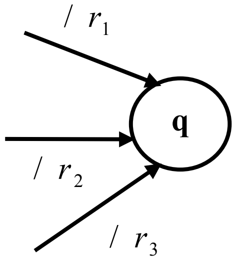


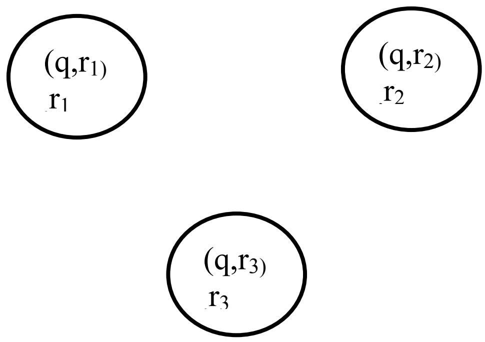


$M_t=(Q_t,S,R,FT,G,q_i)$

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

M_t **M_s**



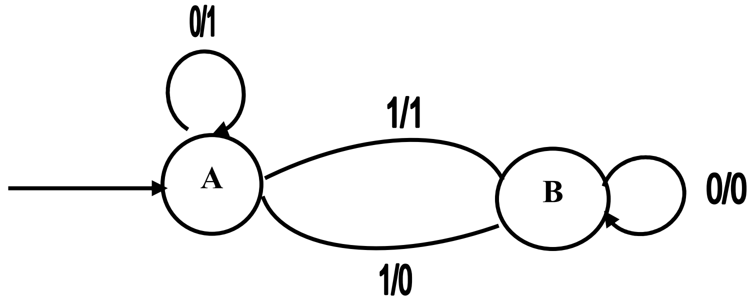


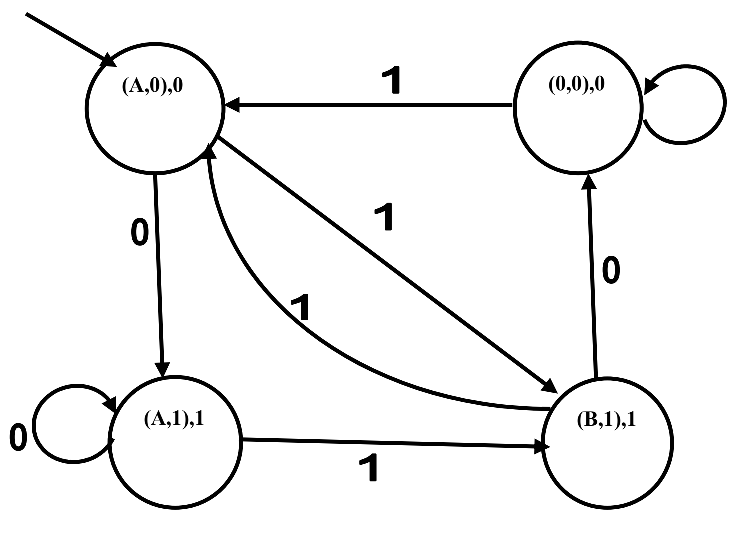
هنگامی در M_t انتقالی به صورت $q \xrightarrow{s/r} q'$ داشته باشیم در M_s انتقال

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

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 خذف 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:

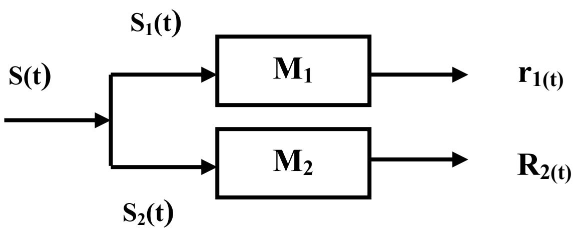
۱.الفبای ورودی آنها و الفبای خروجی آنها یکسان باشد.

$S_1=S_2, R_1=R_2$

2-برای هر تحریک M_1 و M_2 پاسخهای یکسانی تولید نمایند یعنی اینکه اگر $t \quad 1, s_1(t)=s_2(t)$

آنگاه $r_1(t)=r_2(t)$

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



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

۱-انعکاسی $M \approx M$

۲-تقارنی $if \quad M_1 \approx M_2 \Rightarrow M_2 \approx M_1$

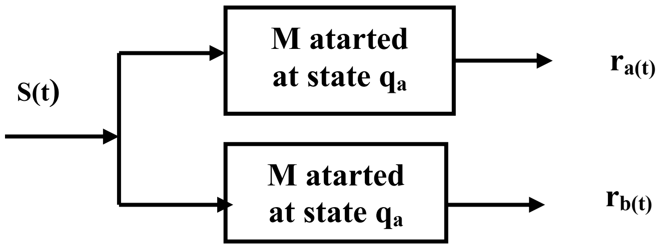
۳-تعدی $if \quad M_1 \approx M_2 \wedge M_1 \approx M_3 \Rightarrow M_1 \approx M_3$

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

حالات معادل:

تعریف:دو حالت q_b,q_a 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€s، $g(q_b,s)=g(q_b,s)$

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

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

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

اثبات:اثبات اگر: بایستی نشان بدهیم اگر شرایط 1و2 برقرار باشند آنگاه q_b,q_a معادل هستند.

فرض کنیم رشته Sω یک رشته ورودی انتخابی شامل سمبل $s \in S$ ورشته $\omega \in S^*$ باشد رفتار M رادر دوحالت مورد بررسی قرار می دهیم.

$$q_a \xrightarrow{s/r} q'_a \xrightarrow{\omega/\varphi} q''_a \qquad r\varphi = r'\varphi'$$
$$q_b \xrightarrow{s/r'} q'_b \xrightarrow{\omega/\varphi} q''_b$$

اثبات فقط اگر:بایستی نشان دهیم اگر $q_a \approx q_b$ آنگاه شرایط 1و2 برقرار هستند.

داریم: $r\varphi = r'\varphi'$

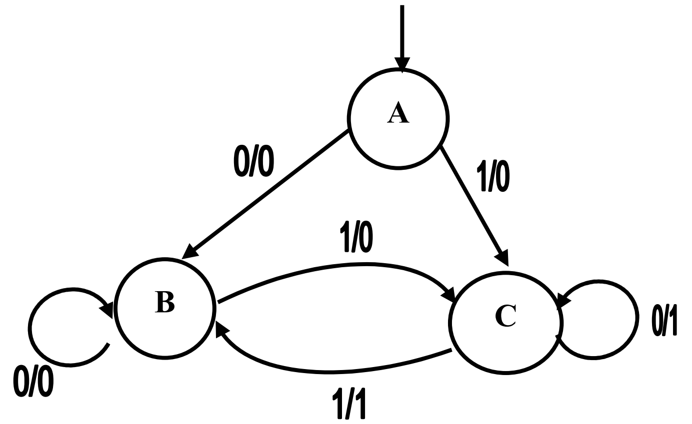
$$q_a \xrightarrow{s/r} q'_a \xrightarrow{\omega/\varphi} q''_a$$
$$q_b \xrightarrow{s/r'} q'_b \xrightarrow{\omega/\varphi} q''_b$$

پس $\varphi = \varphi', r = r'$

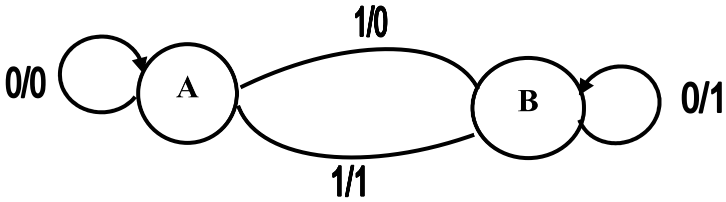
وقتی $r = r'$ در نتیجه شرط 1برقرار است.

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

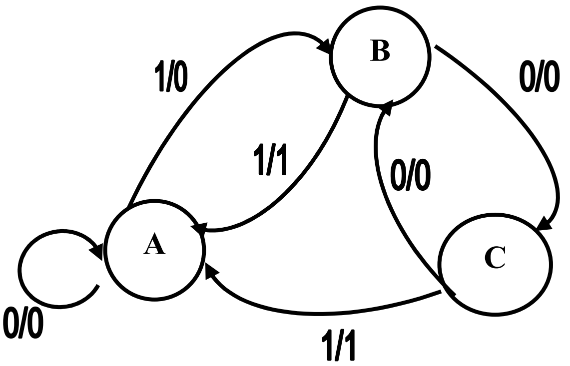
مثال :آیا $A \approx B$ است؟



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.

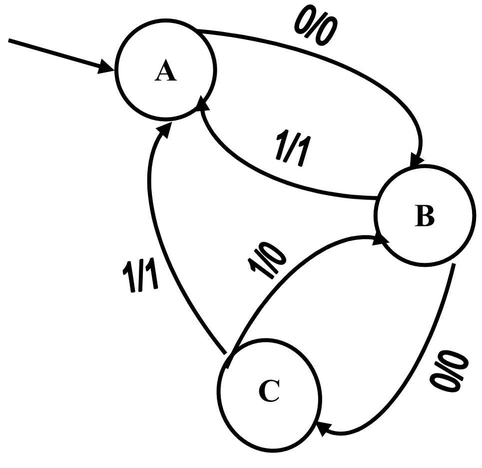
تعریف:یک حالت qدر یک اتوماتای محدود قابل دسترسی(accessible)است اگر ورودیهایی مثل ω وجود داشته باشد که $q_i \xrightarrow{\omega} q$ یک ماشین حالت محدود متصل(connected) است اگر هر حالت از آن قابل دسترسی باشد.

دنبا له های تمیز دهنده و معادل بودن k تائی:Distinguishing sequences &k-Equivalence

تعریف: حالتهای qb,qa از یک ماشین Transition Assigned مثل M=(Q,S,R,F,G,QI)

K-distinguishable(متمایز از مرتبهK)هستند اگر و فقط اگر رشته ای مثل $\omega \in s^*$ با $|\omega| \le k$ 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 ωحداقل در یک سمبل متفاوت باشد.

رشتهω را دنباله تمیز دهنده در حالتq_aوq_bگویندq_aوq_bمتمایز از مرتبهk نباشد آنها را معادل از مرتبهk گویند (k-Equivalence).



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

۱-آنها1-equivalenceباشند.

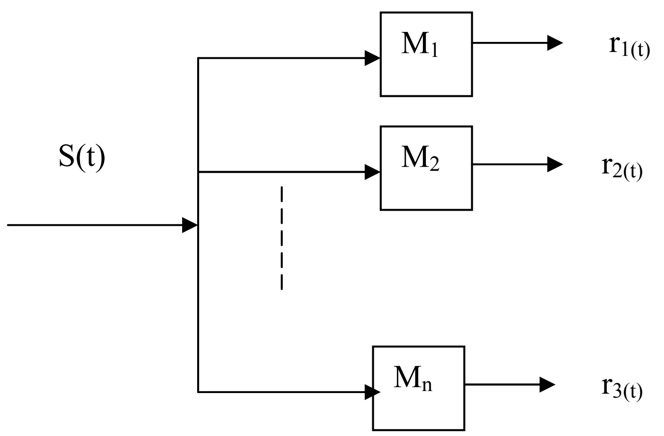
۲.برای هر عمل ورودی مثل s-successorsهای آنها(k-1)-Equivalence باشند.

افزار نمودن مجموعه حالات:

اگرp_1,p_2افزارهایی از مجموعه x باشند و اگر هر بلاک از p_2دقیقا زیر مجموعه یک بلاک از p_1 باشد گوئیم p_2تصفیه ای (refinement)باری p_1است. یعنی اگر $p_1=\{A_1,A_2,\ldots,A_n\}$ و $p_2=\{B_1,B_2,\ldots B_m\}$افزارهائی از مجموعه x باشند و p_2تصفیه ای از p_1 باشد آنگاه برای هر بلاک B_jدر p_1یک بلا مثل A_iدر p_1 وجود داشته باشد بقسمی که:

$$B_j \subseteq A_i$$

$$m \geq n$$



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

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

حالات q_i,q_j متعلق به یک بلاک که از افرازی 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 دنبا له های افزارهای $p_1,p_2,p_3,\ldots p_m,p_{m+1}$ تشکیل دهیم تبسمی که هر بلاک از افزار $1<k<m+1,p_k$ شامل تنها حالاتی که به صورت دوبدو k-Equivalence .

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

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

تمرین

۳،۱.گرامرهای زیر را در نظر بگیرید.

G_1:

$$\Sigma \longrightarrow \lambda$$
$$\Sigma \longrightarrow S$$
$$S \longrightarrow ss$$
$$S \longrightarrow c$$

G_2:

$$\Sigma \longrightarrow \lambda$$
$$\Sigma \longrightarrow S$$
$$S \longrightarrow cSd$$
$$S \longrightarrow cd$$

G_3:

$$\Sigma \longrightarrow \lambda$$
$$\Sigma \longrightarrow S$$
$$S \longrightarrow Sd$$
$$S \longrightarrow cS$$
$$S \longrightarrow c$$
$$S \longrightarrow d$$

G_4:

$$\Sigma \longrightarrow cS$$
$$S \longrightarrow d$$
$$S \longrightarrow cS$$
$$S \longrightarrow Td$$
$$T \longrightarrow Td$$
$$T \longrightarrow d$$

G_5:
$$\Sigma \longrightarrow \lambda$$
$$\Sigma \longrightarrow S$$
$$S \longrightarrow ScS$$
$$S \longrightarrow c$$

الف)زبانهای($L(G_i)$را برای $i=1,\ldots s$ 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.

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

الف) $\{0^m1^n \mid m > n \geq 0\}$

ب)$\{0^k1^m0^n \mid n = k + m\}$

ج) $\left\{0^m1^n \mid \begin{matrix} m(even) \\ n(odd) \end{matrix} \quad oR \begin{matrix} m(odd) \\ n(odd) \end{matrix}\right\}$

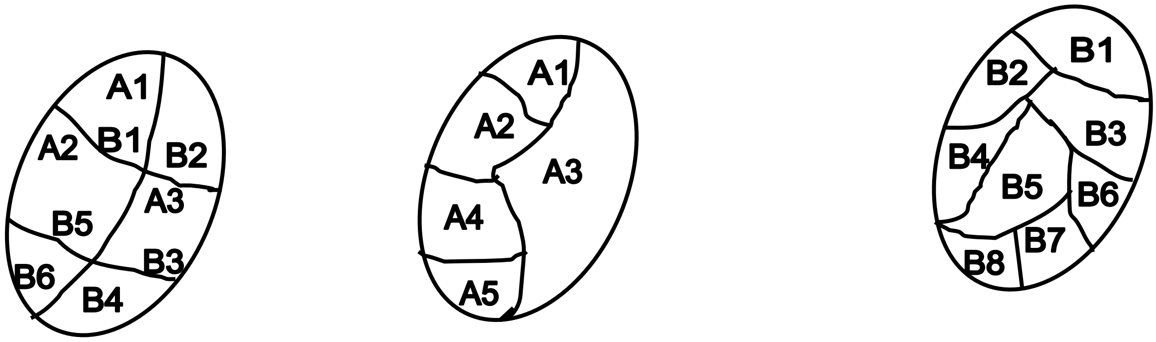
د) $\{\omega c \omega \mid \omega \in \{0,1\}^*\}$

افراز نمودن مجموعه در جملات

اگرP_1وP_2افرازهایی از مجموعه Xباشند و اگر هر بلاک ازp_2دقیقا زیر مجموعه یک بلاک ازP_1 باشد گوئیمP_2تصفیه ای برای P_1است

$P_1=\{A_1,A_2,\ldots,A_n\}$
$P_2=\{B_1,B_2,\ldots,B_n\}$

افرازهایی از مجموعهXباشد و P_2تصفیه ای از P_1باشد آنگاه برای هر بلاک B_jدرP_2یک بلاک مثلA_i در P_1وجود داشته باشد به قسمی $B_j \subseteq \underset{m \geq n}{A_i}$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 $Q=\{q_1,q_2,\ldots,q_n\}$و فرض می کنیم حالت آغازین $M_i=(Q<S<R,f,g,q_i)$باشد حالات q_i,q_jمتعلق به یک بلاک(از افرازی مثل Pاز 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 $P_1,P_2,\ldots,P_m,P_{m+1}$ را تشکیل دهیم به قسمی که هر بلاک از افرازP_Kکه $1 \le k \le m+1$ شامل تنها حالاتی باشد که به صورت دو به دو K-Equivalence هستند.

الگوریتم افزار سازی

۱-یک افزار ابتدایی مثلp_1ازQبا گروه بندی حالاتی که1-equivalentهستند تشکیل بده یعنی حالاتی که خروجی یکسانی برای هر سمبل ورودی دارند را در یک بلاک قرار بده.حالاتq',qدر یک بلاک از p_1هستند اگر و فقط اگر$g(q,s)=g(q',s)$ $\forall s \in S$

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

a.در یک بلاک از p_kقرار داشته باشند.

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

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

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

قضیه:یک روال موثر برای افراز کردن حالات یک ماشین حالت محدود بـه بلاکهـایی از حـالات معـادل وجـود 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

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

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

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

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

P_5

$\vdots$

مثال:

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

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

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

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

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

P_6
$\vdots$

	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

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 مجموعهٔ V قرار دارد پس در داخل جدول V را قرار می‌دهیم.

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

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

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

## فصل پنجم		Finite State Language

ارتباط Finite State Machines و گرامرهای Regular:

هدف :

1. زبان L بوسیلهٔ یک یا چند Finite State acceptor تشخیص داده می‌شود.
2. زبان L بوسیلهٔ یک یا چند گرامر منظم تولید می‌شود.
3. زبان L بوسیلهٔ یک یا چند عبارت منظم تشریح می‌شود.

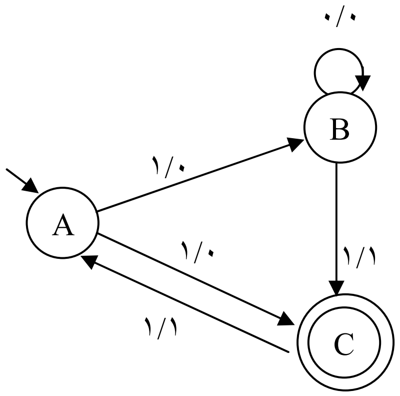
: Finite State Acceptor

اگر بخواهیم یک ماشین Transition Assigned را با .F.S.A نشان دهیم، سمبل‌های خروجی R={0,1} خواهند بود. انواع حالات عبارت خواهند بود از:

1. حالات قبول: خروجی ۱ است.
2. حالات Rejecting: خروجی ۰ است.

:Nonditerministic Accpters

شکل مقابل نمونه‌ای از یک .N.A میباشد. در این شکل از نمایش خروجی‌ها صرف نظر شده و بجای آن حالاتی را که با دو دایره مشخص شده‌اند بعنوان حالت نهائی معرفی می‌کنیم. اینها تنها حالاتی هستند که خروجی ۱ دارند. این خروجی‌ها با رنگ قرمز مشخص شده‌اند.



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 تائی بفرم $M=(Q,S,P,I,F)$ است که در آن:

Q : نشان دهندهٔ مجموعهٔ حالات است.

S : الفبای ورودی است.

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

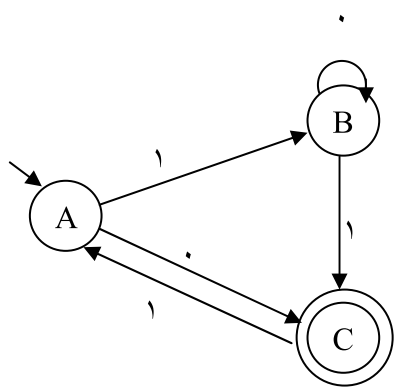
F : $F \subseteq Q$ و نشاندهندهٔ حالات پایانی است.

P : $P \subseteq Q \times S \times Q$ و رابطهٔ 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 $P: Q \times S \to Q$ نوشت و اگر M دقیقاً دارای یک حالات ابتدائی باشد. آنگاه M را Deterministic Finite State Acceptor گویند.

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



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

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

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

دنبالهٔ سوم مورد قبول نمیباشد، زیرا حالت B، حالت نهائی نمیباشد اما دو دنبالهٔ 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 (شبیه سازی شده) میگردد.

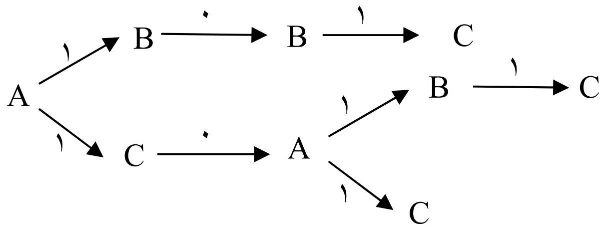
عبارت $q \overset{w}{\Rightarrow} q'$ در ماشینهای قبلی، در این جا مفهوم درستی ندارد و می‌نویسم:

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 رشته‌ای مثل $W \in S^*$ را قبول می‌کند، اگر و فقط اگر برای برخی از $q' \in F, q \in I$ داشته باشیم: $q \overset{w}{\Rightarrow} q'$.

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

مثال: آیا ماشین بالا رشتهٔ ۱۰۱۱ را می‌پذیرد یا نه؟



چون C روی ۱ انتقالی ندارد، مسیر قابل قبول است.

چون C روی ۱ انتقالی ندارد، مسیر غیر قابل قبول میباشد.

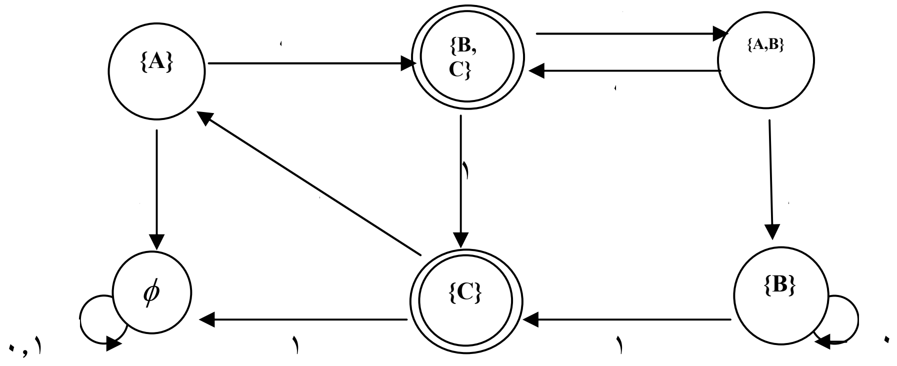
ماشینهای FSA از نوع نامعین برای پیاده سازی مناسب نیستند و آنها را به ماشینهای معین تبدیل می‌کنیم. با در نظر گرفتن رفتار ماشین در هر تصمیم گیری بصورت در ماشین مجزا (که موازی کار میکنند)، رابطه‌ی بشکل زیر خواهیم داشت:

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

کل مسیرهای قابل پیمایش را بصورت زیر بدست می‌آوریم:

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

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

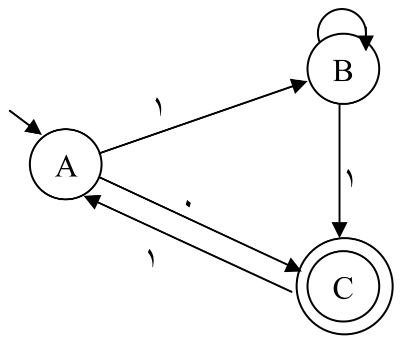
بعبارتی X مجموعه حالاتی است که ماشین با دیدن رشتهٔ W، می‌تواند در آن قرار داشته باشد. یا :

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

$$X[\lambda] = I_n$$

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

مثال:



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

انتقالات A روی ϕ　　$X[\underbrace{0}_{\lambda.0}]=\left\{q'\in Q \mid q''\overset{\circ}{\Rightarrow}q', q''\in X[\lambda]\right\}=\phi$

انتقالات C,B روی ϕ　　$X[10]=\left\{q' \mid q''\overset{\circ}{\Rightarrow}q', q''\in X[1]\right\}=\{A,B\}$　و　$X[\underbrace{1}_{\lambda.1}]=\{B,C\}$

ساخت ماشین معین:

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

۲. حالت شروع در $X[\lambda], M_d$ است:　　$X[\lambda], I_d$

۳. حالت قبول ماشین M_d:　　$F_d=\{X\in Q_d \mid X\cap F_n\neq\varphi\}$

۴. حالت X' در M_d ، S-Successor ، X است اگر X' شامل دقیقاً حالاتی از M_n باشد که S-Successor حالتهای مجموعه X در M_n هستند.

۴ عمل فوق را تا زمانیکه Xها 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]=ϕ

چون {B,C} قبلاً بدست آمده بود، رشتهٔ ۱۰۱ را ادامه نمی‌دهیم.

چون {A} قبلاً بدست آمده بود، رشتهٔ ۱۱۰ را ادامه نمی‌دهیم.

با توجه به محاسبات اخیر، مجموعهٔ حالات و انتقالهای همانند شکل جلسهٔ قبل بدست می‌آید.

قضیّه: 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 خواهد بود. باید نشان دهیم $L(M_d) = L(M_n)$

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 رشتهٔ مثل w ماشین M_d را بحالت $X[w]$ منتقل می‌سازد:

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

بصورت استقراء:

پایه: اگر $w = \lambda$ باشد $X[\lambda] \overset{\lambda}{\Rightarrow} X[\lambda]$

فرض کنیم $W = \gamma.S$ همچنین $X[\lambda] \overset{\gamma}{\Rightarrow} X[\gamma]$. (اگر γ رشته‌ای بطول n باشد $\gamma.S$ رشته‌ای با طول n+1 خواهد بود.) می‌خواهیم ثابت کنیم: $X[\lambda] \overset{\gamma.s}{\Rightarrow} X[w]$

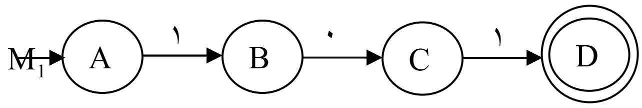
$$X[w] = X[\gamma.s]\left\{q' \mid q'' \overset{s}{\Rightarrow} q', q'' \in X[\gamma]\right\}$$

$$\text{فرض} : X[\lambda] \overset{\gamma}{\Rightarrow} X[\gamma] \xrightarrow{s} X[w] \qquad \Rightarrow \qquad X[\lambda] \overset{\overset{w}{\overline{\gamma.s}}}{\Rightarrow} X[w]$$

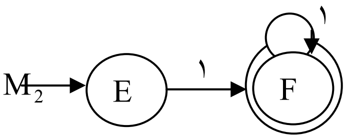
نتیجه: قدرت ماشینهای نامعین و ماشینهای معین بیک اندازه است. بنابراین در شرایطی که طراحی N.A ساده‌تر باشد، پس از طراحی ماشین نامعین، آنرا، ماشین معین تبدیل میکنیم.

مثال : فرض کنید میخواهیم یک ماشین FSA معین درست کنیم که رشته‌هائی که شامل تعـدادی صـفر بـوده و یکبار زیر رشته ۱۰۱ یا زیر رشته‌ای از تعدادی ۱ در آن ظاهر شده باشد را تشخیص دهد.

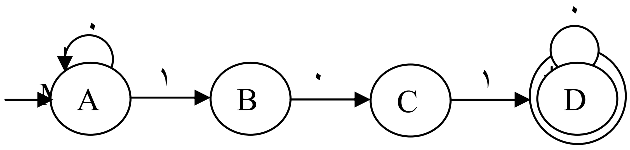
– رشته‌هائی که ماشین باشد تشخیص دهد بصورت $0^*(101 \cup 11^*)0^*$ می‌باشند.



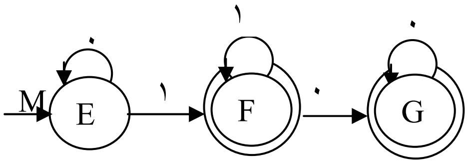
$L(M_1) = \{101\}$



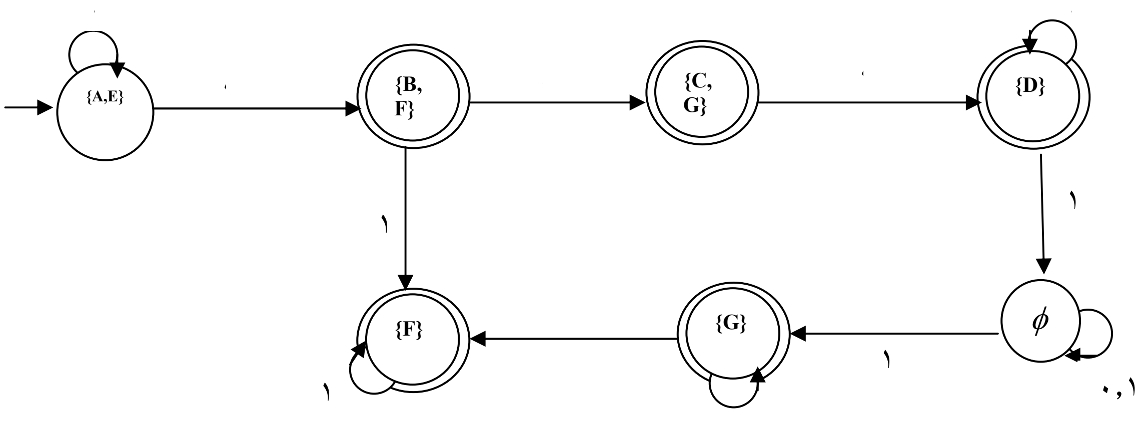
$L(M_2) = \{11^*\}$



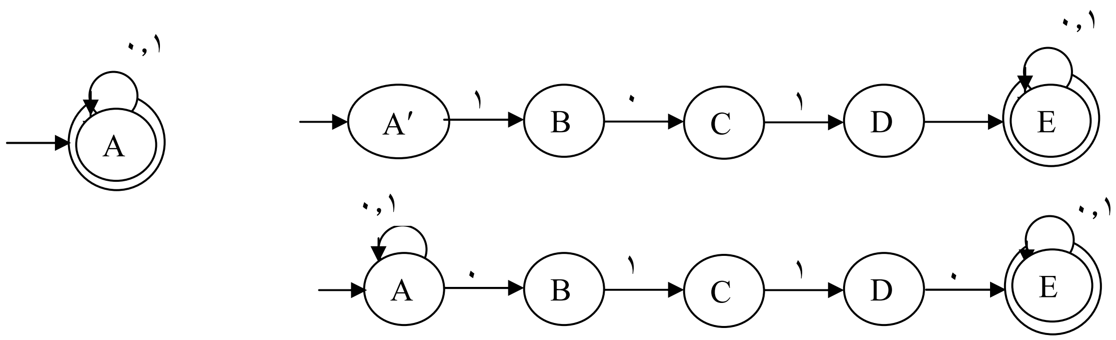
$L(M_3) = \{0^*1010^*\}$



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



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:

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

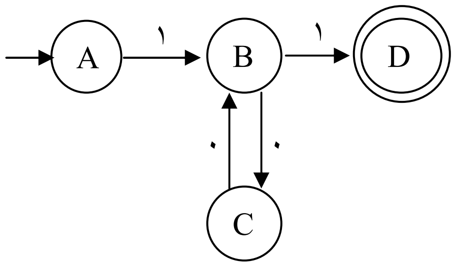
$$\begin{array}{l} A\rightarrow 0\mid 1B \\ B\rightarrow 0\mid 0B \end{array} \quad \Rightarrow \quad \begin{array}{l} L(G,A)=\{0\cup 100^{*}\} \\ L(G,B)=\{00^{*}\} \end{array}$$

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

$$E(q)=\left\{W\in S^{*}\mid q\overset{w}{\Rightarrow}q',q'\in F\right\}$$

مثال:

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



نتایج ۱:

۱. $\lambda\in E(q)$ اگر و فقط اگر $q\in F$

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

۳. $L(M)=\bigcup\limits_{q\in I}(q)$

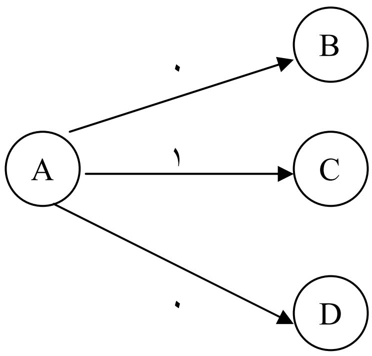
نتیجهٔ ۲:

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

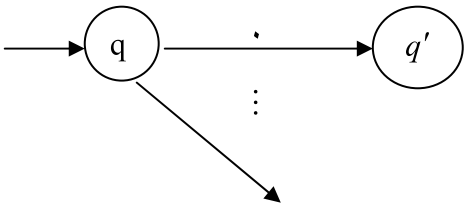
$$q\in Q\quad,\quad E(q)=\bigcup_{q'\in Q}V(q,q')E(q')\cup W(q)$$

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

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



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



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

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

$$q\xrightarrow{s}q' \qquad A\rightarrow sB$$

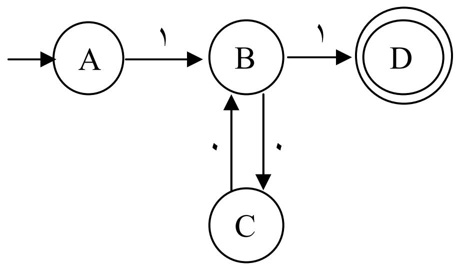
$$E(q)\supseteq S.E(q') \qquad L(G,A)\supseteq S.L(G.B)$$

مثال: $B\rightarrow 1$ $\quad A\rightarrow 1B$ $\quad G:\Sigma\rightarrow A$

$B\rightarrow 0C$

$C\rightarrow 0B$

M:



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

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

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

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

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

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

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

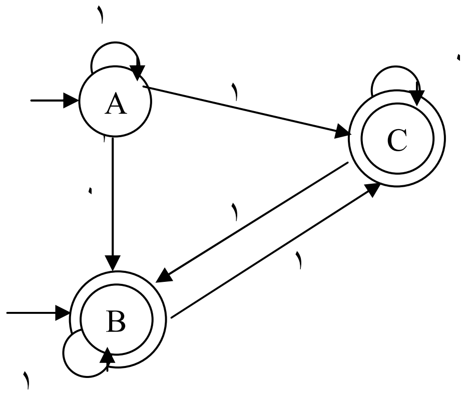
ساخت یک گرامر از روی یک ماشین پذیرندۀ حالت محدود:

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

مثال:

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

$F=\{B,C\}$



$\Sigma \to \lambda$ طبق قاعده ۱

$\Sigma \to A$ طبق قاعده ۲

$\Sigma \to B$ طبق قاعده ۲

$A \to 1$ طبق قاعده ۳

$B \to 1$ طبق قاعده ۳

$C \to 0$ طبق قاعده ۳

$C \to 1$ طبق قاعده ۳

$A \to 1A$ طبق قاعده ۴

$A \to 1C$ طبق قاعده ۴

$B \to 1B$ طبق قاعده ۴

$B \to 1C$ طبق قاعده ۴

$B \to 0A$ طبق قاعده ۴

$C \to 0C$ طبق قاعده ۴

$C \to 1B$ طبق قاعده ۴

قضیه: برای هر FSA مثل M، می‌توان 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 بوسیلهٔ و گفته شده از M تشکیل شده است.

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

$W \in L(G)$ if and only if $W \in L(M)$

اگر $\lambda \in L(M)$ پس داریم: $I \cap F \neq \phi$. اما با روش ساخت G، G باید شـامل قاعـدهٔ تولیـد $\Sigma \rightarrow \lambda$ باشـد، در صورتیکه $I \cap F \neq \phi$. بنابراین $\lambda \in L(G)$ اگر 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 مـی- 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 $W = S_1S_2S_3...S_k$، دنبالهٔ حالات زیر را در نظر می‌گیریم:

$$q_\circ \xrightarrow{S_1} q_1 \xrightarrow{S_2} q_2 \xrightarrow{S_3} ... \xrightarrow{S_k} q_k \qquad q_\circ \in I$$

$$q_k \in F$$

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

$$\Sigma \rightarrow N(q_\circ) \Rightarrow S_1N(q_1) \Rightarrow S_1S_2N(q_2) \Rightarrow ...$$

$$\Rightarrow S_1S_2...S_{K-1}N(q_{K-1}) \Rightarrow S_1S_2...S_K$$

بنابراین $W \in N(q_\circ)$ اگر و فقط اگر $N(q_\circ) \overset{*}{\Rightarrow} W$.

پس اثبات کامل است.

نتیجهٔ ۱: دنباله‌های حالات در M از $q \in I$ به $q' \in F$ در تناظر یک به یک با اشتقاقهای رشته‌های ترمینـال از Σ در G می‌باشد.

نتیجهٔ۲ : برای هر q، مجموعه E(q) از M و عنصر غیر پایانی N(q) در G، رابطهٔ زیر را ارضا می‌کنند:

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

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

ساخت یک FSA از روی یک گرامر Right-linear:

ورودی: $G=(N,T,P_G,\Sigma)$

خروجی: $M=(Q,T,P_M,I,\{q_F\})$

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

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

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

$G:\Sigma\to\lambda \qquad A\to aA \qquad C\to aD$

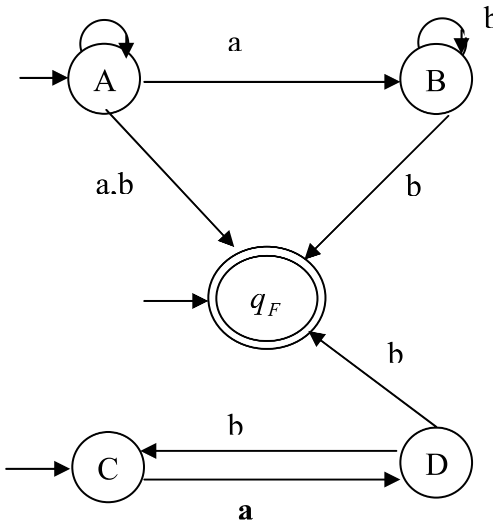
$\Sigma \to A$ $\qquad A \to aB$ $\qquad D \to bC$

$\Sigma \to C$ $\qquad B \to bB$ $\qquad D \to b$

$A \to a$

$A \to b$

$B \to b$



قضیه: برای هر گرامر right-linear مثل G می‌توان 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 می‌توان بطور مشابه با قضیه قبل صورت داد.

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

تبدیل گرامر Right-linear به Left-linear :

If G has $\quad$ Then G' has

$$\Sigma \to SA \qquad A \to S$$

$$A \to SB \qquad S \to As$$

$$A \to S \qquad \Sigma \to As$$

$$\Sigma \to S \qquad \Sigma \to S$$

$$\Sigma \to \lambda \qquad \Sigma \to \lambda$$

مثال:

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

نتیجه: جملات زیر معادل هستند:

۱. زبان L با یک FSA تشخیص داده می‌شود.

۲. زبان L با یک گرامر Right-Linear تولید می‌شود.

۳. زبان L با یک گرامر Left-Linear تولید می‌شود.

عبارات منظم و FSA:

هدف: برای یک FSA می‌توان یک عبارت منظم تشکیل داد که نشاندهندهٔ زبانی است که FSA تشخیص می‌دهد و بر عکس.

عبارت منظم

تعریف: فرض کنیم V یک الفبای محدود باشد. یک عبارت منظم روی V هر رشتهٔ محدودی از سمبل‌های مجموعهٔ $\{a \mid a \in V\} \cup \{\cup, *, (,), \lambda, \varphi\}$ می‌باشد که می‌تواند بر اساس قواعد زیر تشکیل شود.

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

λ رشته‌ای است شامل صفر سمبل.

تهی مجموعه‌ای 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 β,α عبارات منظم باشند عبارات زیر منظم هستند.

4.a $(\alpha\beta)$

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

4.c (α^*)

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

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

$(\alpha \cup \beta)$ منظم است. $\Rightarrow$ $\wp$ منظم است $\Rightarrow$ abb^* منظم است $\Rightarrow \left.\begin{matrix} ab \\ b^* \end{matrix}\right\}$ چون b,a منظم هستند. چون b منظم

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

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

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

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

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

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

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

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

وارون یک رشته: $\wp^R$ را وارون رشتهٔ $\wp$ گویند.

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

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

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

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

حل مجموعه معادلات سیستم

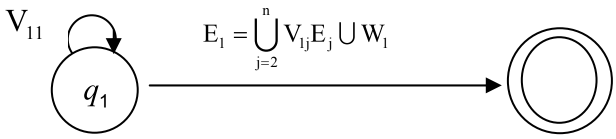
هدف: تشکیل عبارت منظمی برای یک FSA می‌باشد که تشریح کنند. رفتار ماشین است (یعنی رشته‌هائی که ماشین پذیرش می‌کند.

E(q)={W| مورد پذیرش ماشین با شروع از حالت q است W}

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

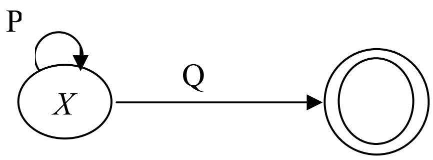
$$E_k = \text{end set } q_k \qquad V_{kj} = \{s \in S \mid q_k \xrightarrow{S} q_j\}$$

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



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

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



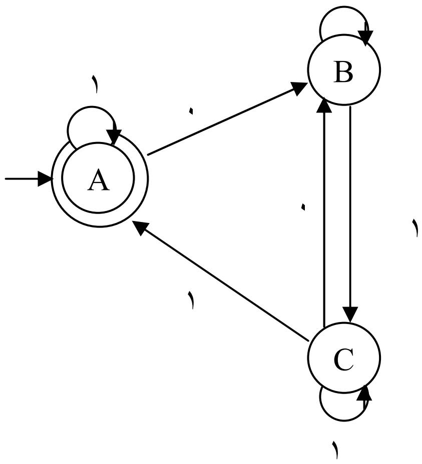
$$X = PX \cup Q$$
$$\Downarrow$$
$$X = P^* Q$$

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

$$E_K = \bigcup_{j=2}^{n} \overbrace{\left(V_{k1} V_{11}{}^* V_{1j} \cup V_{kj} \right)}^{Const} \overbrace{E_j}^{Uariable} \cup \overbrace{V_{k1} V_{11}{}^* W_1 \cup W_k}^{Const}$$

K=2,…,n

مثال:



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

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

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

$$C = \overbrace{1A \cup 0B}^{Q} \cup \overset{P}{1}C \qquad L(M)=A$$

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

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

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

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

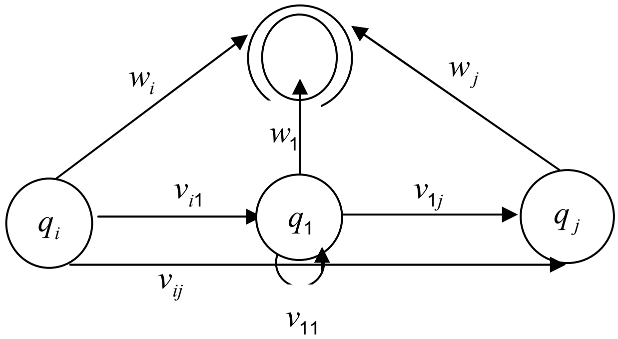
$$A = 1A \cup 0(1 \cup 11^*0)^* 11^*1A \cup \lambda$$

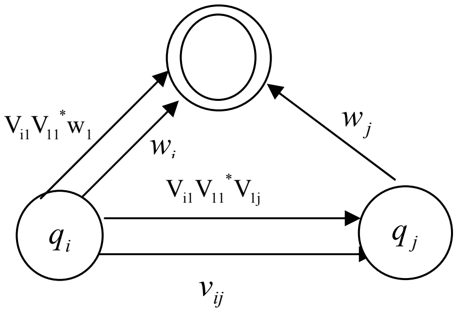
$$A = \underbrace{(1 \cup 0(1 \cup 11^*0)^* 11^*)}_{P} 1A \cup \underbrace{\lambda}_{Q}$$

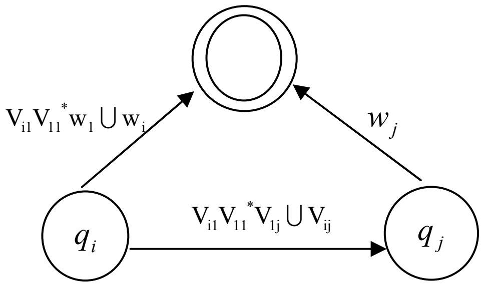
$$A = (1 \cup 0(1 \cup 11^*0)^* 11^*)^* \lambda = L(M)$$

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

۱



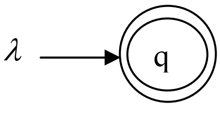


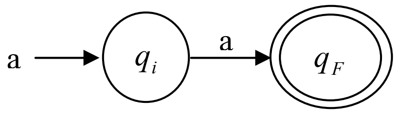


قضیه : هر FSA زبانی را تشخیص می‌دهد که می‌تواند با یک عبارت منظم تشریح گردد.

تشکیل پذیرنده برای یک عبارت منظم:

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





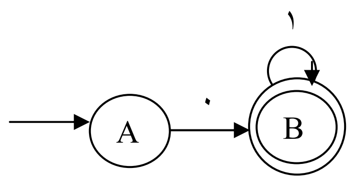
می‌خواهیم برای عبارات پیچیده نظیر $(a \cup b)^*(a^*b^*c^*)^*$ماشین تشکیل دهیم.

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

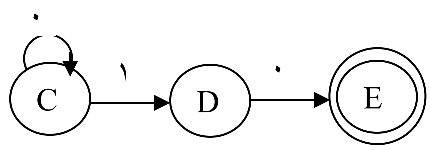
مثال: فرض کنیم می‌خواهیم ماشینی که چنین رشته‌هائی را بپذیرد تشکیل بدهیم: $R_1 = 01^*$

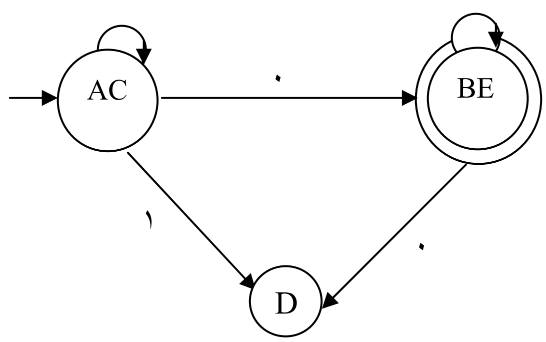
در حقیقت می‌خواهیم ماشین $R_1 \cup R_2$ را تشکیل بدهیم. $R_2 = 0^*10$

R_1:



R_2:

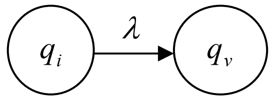




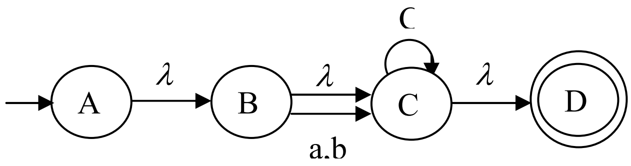
$$0^*(0 \cup 10)1^* = 0^*01^* \cup 0^*101^*$$

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

λ –Transition : 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 یک FSA است بفرم $M=(Q,S,P,q_I,q_F)$ که دارای λ –Transition 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. انتقالات M' شامل تمامی انتقالات M و بعلاوه انتقالاتی بفرم زیر می‌باشد:

$$q_I \xrightarrow{\lambda} q \qquad q\in I \qquad , \qquad q' \xrightarrow{\lambda} q_{F\in} \qquad q'\in F$$

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

تبدیل acceptor – λ 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 نمی‌کند در حقیقت ماشین باز هم یک acceptor – λ است اما بدون حلقه‌ای از λ.

مراحل تبدیل acceptor – λ به FSA :

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

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

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 می‌دهیم:

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

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 نمی‌توان گفت "حتماً می‌توان به یک FSA رسید."

بردار

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

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

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

۳. انتقالات M' بصورت زیر می‌باشد:

a. هر انتقال $q \xrightarrow{s} q'$ $(s \in S)$ 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 acceptor – λ مقابل را به FSA تبدیل کنیم.

بردار

در مرحلهٔ اول Loop– λ 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:

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

$L(M_1) = R_1$ بردار

$L(M_2) = R_2$ بردار

$L(M_3) = R_1 \cup R_2$ بردار

ماشین M_3 از نوع **acceptor** – λ می‌باشد.

بردار

$L(M_4) = R_1 . R_2$

بردار

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

ماشینهای M_5, M_4 نیز از نوع **acceptor** – λمی‌باشند.

قضیه: برای هر عبارت منظم مثل αروی الفبای V می‌توان یک ماشین FSA مثل M ساخت که L(M)=R که در آن R نشان دهندهٔ مجموعه رشته‌های مشخص شده بوسیلهٔ α است.

اثبات: ثابت می‌کنیم که می‌توان برای α یک **acceptor** – λ تشکیل داد.

اثبات با استقرار روی طول α($|\alpha|$):

پایه: $|\alpha| = 1$

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

فرض: فرض میکنیم نحوهٔ تشکیل acceptor – λ برای $1\le|\alpha|<K$ را می‌دانیم.

حکم: می‌خواهیم نحوهٔ تشکیل acceptor – λ برای $|\alpha|=K$ را پیدا می‌کنیم.

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

$$q_I\overset{W}{\Rightarrow}q_F$$

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

$$q_{I1}\overset{W}{\Rightarrow}q_{F1} \qquad W\in\alpha_1$$

$$q_{I2}\overset{W}{\Rightarrow}q_{F2} \qquad W\in\alpha_2$$

مثال: میخواهیم 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 $F:\ell\to\ell$ بسته است اگر برای هر $L=\ell$ ، $F(L)\in\ell$. هم چنین ℓ تحت یک عمل باینری مثل $G:\ell\times\ell\to\ell$ بسته است اگر برای هر $L_1,L_2\in\ell$ ، $G(L_1,L_2)\in\ell$.

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

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

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

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

$$L_1^C=A^*-L1=(a/b)^*-a^*b$$

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

ابهام (Ambiguity)

ابهام در رابطه با زبانهای منظم:

تعریف: گرامری مبهم است که برا یک جملهٔ ورودی چندین اشتقاق داشته باشد.

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

$$B \to 0B \quad C \to 1C$$

$$B \to 0C \quad C \to 1$$

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

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

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

تشخیص وجود ابهام:

برای گرامر یک FSA میسازیم: $M=(Q,S,P,I,F)$

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

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

فرض کنیم : $W = \gamma\psi$

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

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

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

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

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

اگر حاصل این اشتراک مجموعه‌ای شامل حداقل ۲ حالت باشد آنگاه ماشین دارای حالاتی مثل q',q بوده که برای رشته‌ها دو دنبالهٔ 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.

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

$$B \to 0B \quad C \to 1C$$

$$B \to 0C \quad C \to 1$$

بردار

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

$$A \overset{1}{\Rightarrow} B \overset{0}{\Rightarrow} B \overset{0}{\Rightarrow} B \overset{0}{\Rightarrow} C \overset{1}{\Rightarrow} D \qquad ۱۰۰۰۱$$

$$A \overset{1}{\Rightarrow} B \overset{0}{\Rightarrow} B \overset{0}{\Rightarrow} B \overset{0}{\Rightarrow} C \overset{1}{\Rightarrow} D$$

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 $P : \ell \rightarrow \{ture,false\}$ 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 پروسیجری در صورت وجود یک الگوریتم موثر برای پیشگوئی P است.

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

۱. آیا $L_1 = L_2$؟

۲. آیا $L_1 = \phi$؟

۳. آیا L_1 محدود یا نامحدود است؟

۴. آیا $L_1 \cap L_2 = \phi$ ؟

۵. آیا $L_1 \subseteq L_2$ ؟

(آیا $L_1 \cap L_1^C = \phi$؟)

۶. آیا G مبهم است؟

فصل ششم :محدودیتهای Final Automata

در این فصل می‌خواهیم جوابگوی سئوالاتی باشیم از قبیل:

۱. آیا رشتهٔ $w \in (0 \cup 1)^*$، even parity است؟

۲. آیا رشتهٔ $w \in (0 \cup 1)^*$، 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 رشتهٔ w عضوی از مجموعهٔ A می‌باشد؟

تعریف: گوئیم اتوماتای M نمونه‌ای از مسئلهٔ 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 مسئلهٔ 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 زبانهائیکه 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.

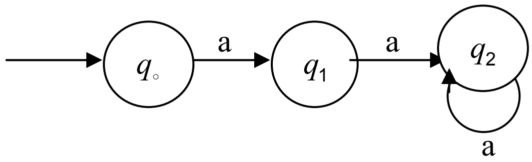
$$G : \Sigma \to S \qquad S \to aSb$$

$S \to ab$

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

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

فرض کنیم n تعداد حالات ماشین M باشد و فرض کنیم $r \geq n$. ثابت می‌کنیم که ماشین M علاوه بر پـذیرش رشتهٔ $a^r b^2$، رشته‌های دیگری از مجموعهٔ $a^* b^*$ 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 رشته‌هائی که دارای تعداد مساوی b,a هسـتند، رشـته‌هـائی از $a^* b^*$ را مـی‌توانـد پذیرش کند که دارای تعداد یکسانی از b,a نمی‌باشد.

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

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

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

قضیه: فرض کنیم L یک زبان منظم باشد و فرض کنیم عدد صـحیحی مثـل $p \geq 0$ وجـود دارد بـه قسـمی کـه $X = \{\alpha_1 (\alpha_2)^k \alpha_3 (\alpha_4)^k \alpha_5 \mid k \geq P\}$ در L وجود دارد. $\alpha_5, \cdots \alpha_2, \alpha_1$ رشته‌هائی رانشان میدهند و α_4, α_2 غیر تهی می‌باشند. آنگاه رشته‌هائی مثل $\beta_5, \cdots \beta_2, \beta_1$ وجود دارد. بقسمی که:

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

زیر مجموعه‌ای از L است:

$\beta_3 \in \alpha^*{}_2 \alpha_3 \alpha_4^*$ $\qquad$ $\beta_1 \in \alpha_1 \alpha^*{}_2$

$\beta_4 \in \alpha^*{}_4 - \lambda$ $\qquad$ $\beta_2 \in \alpha^*{}_2 - \lambda$

$$\beta_5 \in {\alpha^*}_4\alpha_5^*$$

اثبات: فرض کنیم M یک FAS با n حالت باشد و M تشخیص دهندهٔ زبان L باشد. فرض کنیم زبان L شامل مجموعهٔ 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:

$$q_I = q_0 \overset{\alpha_1}{\Rightarrow} q_I \overset{\alpha_2}{\Rightarrow} \cdots \overset{\alpha_2}{\Rightarrow} q_{r+1} \overset{\alpha_3}{\Rightarrow} q_{r+2} \overset{\alpha_4}{\Rightarrow} \cdots \overset{\alpha_4}{\Rightarrow} q_{2r+2} \overset{\alpha_5}{\Rightarrow} q_{2r+3}$$

$$w = \alpha_1\,\overbrace{\alpha_2..\,\underbrace{.\alpha_2\ \alpha_2...\alpha_2\ \alpha_2.}_{x\text{بار}}\,..\alpha_2}^{r}\,\alpha_3\alpha_4$$

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

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

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

$$\beta_1 \in \alpha_1{\alpha^*}_2 \qquad \beta_4 \in {\alpha^*}_4 - \lambda$$

$$\beta_2 \in {\alpha^*}_2 - \lambda \qquad \beta_5 \in {\alpha^*}_4\alpha_5^*$$

$$\beta_3 \in {\alpha^*}_2\alpha_3\alpha_4^*$$

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

$$1.\,(\ ) \in L_P$$

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

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

گرامر context free مقابل زبان L_p را تولید می‌کند.

$$G: \Sigma \to S \qquad S \to (\ )$$

$$S \to (S)$$

$$S \to SS$$

$$X = \{(^{k})^{k} \mid k \geq 1\} \subset L_p$$

$$\beta_1(\beta_2)^{*}\beta_3(\beta_4)^{*}\beta_5 \subset L_p$$

$$\alpha_1 = \lambda, \alpha_2 = (\ , \alpha_3 = \lambda, \alpha_4 = \ ), \alpha_5 = \lambda$$

$\text{v}, \text{y} > 0$ $\qquad$ $*\underbrace{(^{*}}_{\beta_1}(\underbrace{(^{v}}_{\beta_2})^{*}\underbrace{(^{*})^{*}}_{\beta_3}(\ \underbrace{)^{y}}_{\beta_4})\underbrace{^{*})^{*}}_{\beta_5} \subset L_p$

چون تعداد پرانتز باز و بسته‌ها در رشتۀ × یکسان نیست و این با فرض تناقض دارد پـس L_p یـک زبـان مـنظم نیست.

مثال: زبان تصویر آینه (Mirror Image) $\qquad L_{mi} = \{ww^{R} \mid w \in (a \cup b)^{*}\}$

$$X = \{(ab)^{k}(ba)^{k} \mid k \geq 1\} \subset L_{mi}$$

$\alpha_1 = \lambda, \alpha_2 = ab, \alpha_3 = \lambda, \alpha_4 = ab, \alpha_5 = \lambda$ $\qquad$ $\beta_1(\beta_2)^{*}\beta_3(\beta_4)^{*}\beta_5$

$$(ab)^{*}\left((ab)^{v}\right)^{*}(ab)^{*}(ba)^{*}\left((ba)^{y}\right)^{*}(ba)^{*}$$

چون همه رشته‌هائی که تولید می‌شوند عضو L_{mi} نیستند پس L_{mi} یک زبان منظم نیست.

مثال: $\quad L_1 = \{a^{n}b^{m} \mid n \geq m \geq 1\}$

فرض کنیم L_1 منظم باشد.

بنا به قضیه L_1 منظم است پس L_1^{R} هم منظم است. $\qquad L_1^{R} = \{a^{m}b^{n} \mid n \geq m \geq 1\}$

چون L_1^{R} منظم است پس زبان L_2 زیر که فقط جای nonterminalهای b,a عوض شده منظم است:

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

$$L_1 \cap L_2 = \{a^{n}b^{n} \mid n \geq 1\}$$

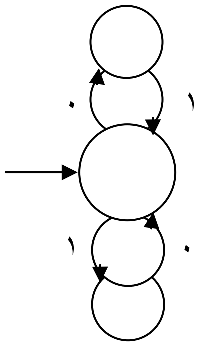
قبلاً ثابت شده که زبانی مثل $L_1 \cap L_2$ منظم نیست پس خود L_1 نیز نمی‌تواند منظم باشد.

محدودیت‌های تولید کننده‌های حالت محدود:

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

.

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



این FSG رشته‌هائی بفرم $(01 \cup 10)^*$را تولید می‌کند.

اگر نشان دهنده یک تولید کننده باشد آنگاه رشته‌هائی بفرم $(01 \cup 10)^*$ را تولید می‌کند.

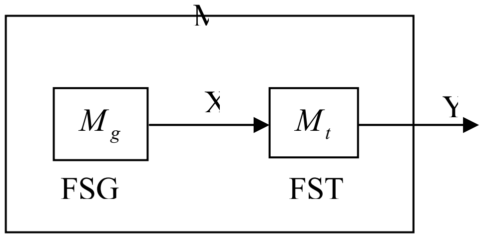
اگر نشان دهنده یک پذیرنده باشد آنگاه رشته‌هائی بفرم $(01 \cup 10)^*$ را پذیرش می‌کند.

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

محدودیت‌های مترجم‌های حالت محدود:

تعریف: فرض کنیم $M = (Q, S, R, f, g, q_I)$ یک مترجم معین باشد . اگر X هر مجموعه از رشته‌های ورودی باشد، ترجمهٔ X بوسیلهٔ M { φپاسخ ماشین M در برابر تحریک $W \in X$ باشد | $Y = \{\varphi \in R^*$

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

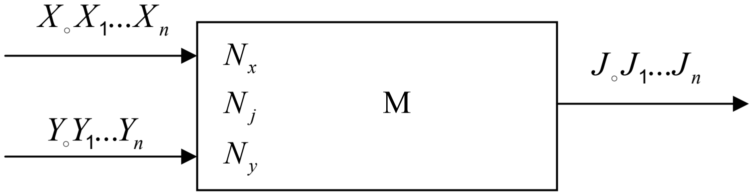


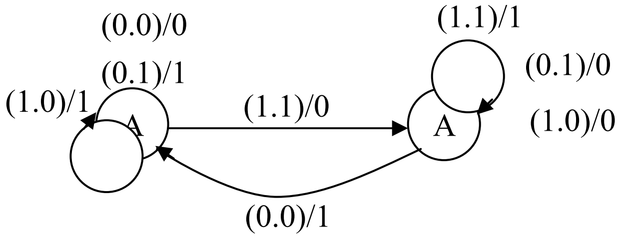
$$M : \underbrace{P}_{PowerSet}\left(S^*\right) \to P\left(R^*\right)$$

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

قضیه: ترجمهٔ یک مجموعهٔ منظم بوسیلهٔ یک FST یک زبان منظم است. یعنی اینکه کلاس مجموعه‌های مـنظم تحت عمل ترجمه به یک FST بسته است.

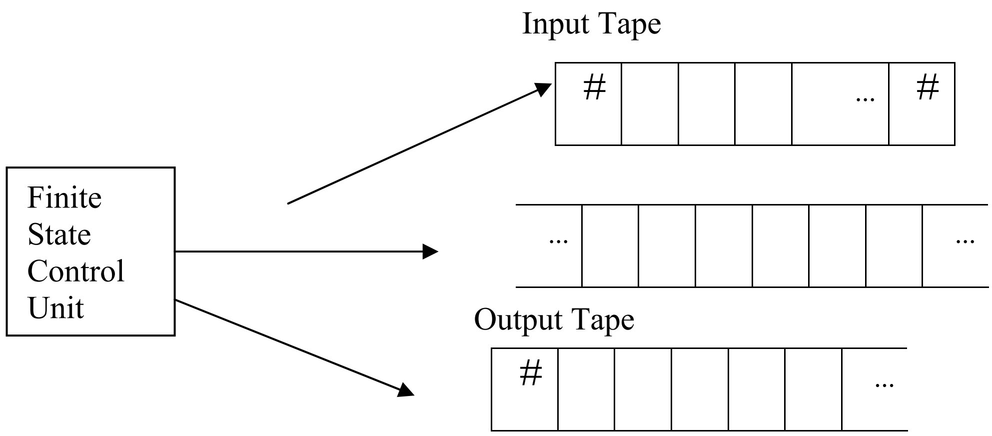
- ماشین M را که کار آن جمع زدن است بصورت زیر تشکیل میدهیم:





فصل هفتم

Tape Automata



مدل فوق یک .T.A را نشان میدهد. فرق اساسی این نوع ماشین بـا FSM، داشـتن حافظـه – 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 سلول‌هـای نـوار ورودی محـدود است. از طرف دیگر طول نوار خروجی میتواند نامحدود باشد. البته ابتدای آن با # مشخص میگردد. طـول 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: $q]more(t,q')$

- 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)
- Accepter (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 براست)
- 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 نامحدودبی

Type of grammar	Type of Accepter	characteristics
3 (Regular)	Finite – State	Storage tape ندارد
2 (Context Free)	Push down	یک نوار حافظهٔ 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

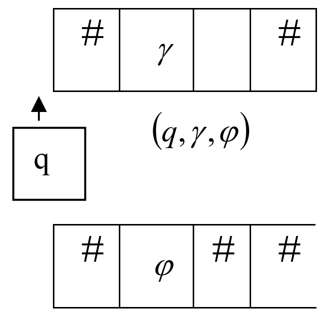
P برنامه GST: = M(Q,S,R,P,I,F)

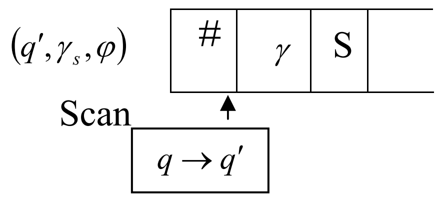
Instructionها:

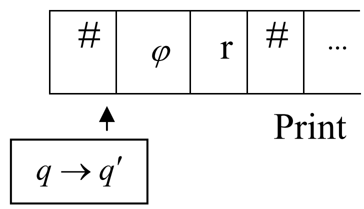
$$\left.\begin{array}{l} q]\,Scan(s,q') \\ q]\,\mathrm{Print}(r,q') \end{array}\right\} \begin{cases} q,q' \in Q \\ s \in S \\ r \in R \end{cases}$$

Machine configuration (پیکربندی ماشین):

1. حالت واحد کنترل
2. رشته‌ای از سمبلها که از نوار ورودی پیمایش کرده
3. رشته‌ای از سمبلها که بر روی نوار خروجی نوشته







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

عملیات از برنامهٔ 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 Configurationای 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 می‌شود.

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

۲. یک عمل $qJ\Pr int(r,q')$ 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' می‌شود.

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

$$(q_\circ,\gamma_\circ,\varphi_\circ)\rightarrow(q_1,\gamma_1,\varphi_1)\rightarrow\cdots(q_k,\gamma_k,\varphi_k)$$

$$(q_\circ,\gamma_\circ,\varphi_\circ)\Rightarrow(q_k,\gamma_k,\varphi_k)$$

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

Conf ابتدائی $\left(\underbrace{q}_{q\in I},\lambda,\lambda\right)$

$(q,\lambda,\lambda)\Rightarrow(q',\gamma,\varphi)$ $\left(\underbrace{q'}_{q'\in F},\gamma,\varphi\right)$

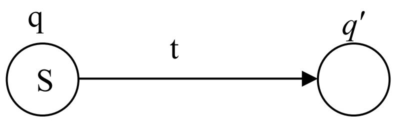
$q\in I,q'\in F$

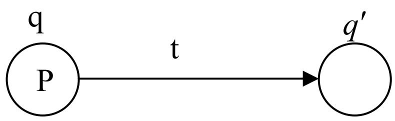
گوئیم φپاسخ ماشین M به تحریک γمی‌باشد.

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

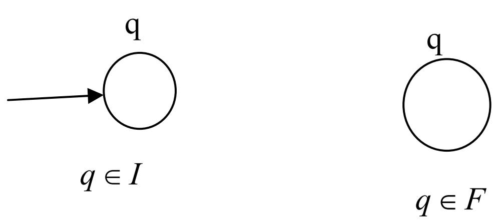
اگر $Y\subset R^*$، آنگاه $X=\{\varphi\in S^*\mid$ پاسخ ماشین M به تحریک می‌باشد $\varphi\in y$ $\}$

$q]Scan(t,q')$





$q]\Pr int(t,q')$

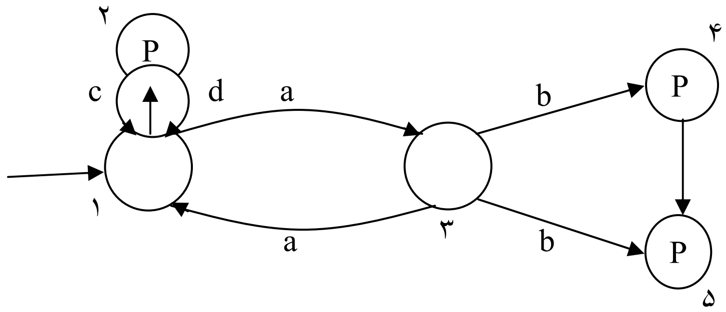


مثال:

I={1}
F={1.3}

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

Confهای نهائی   Conf ابتدائی



1]Scan(b,2)

S={a,b}   1]Scan(a,3)

R={c,b,d}   2]Print(c,1

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

bab   3]Scan(a,1)

$(1,\lambda,\lambda)\xrightarrow{S}(2,b,\lambda)\xrightarrow{P}(1,b,c)\xrightarrow{S}(3,ba,c)$ 3]Scan(b,4)

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

$\xrightarrow{P}(3,bab,cbb)$ 5]Print(b,3)

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

Two-Way-Accepter

اهداف مطالعهٔ T.W.A

1. نشان دادن اینکه قدرت ماشین‌های نواردار بدون داشتن نوار حافظه به اندازه FSA است.
2. نمایش وجود رابطهٔ 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.

$$\begin{rcases} q]\mathrm{left}(s,q') \\ q]\mathrm{Right}(s,q') \end{rcases} \begin{cases} q,q' \in Q \\ s \in S \cup \{\#\} \end{cases}$$

q

...	#	S_n	...	S_k	...	S_2	S_1	#	...

(q,k)

مکان هد نوار ورودی

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

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

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

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

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

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

مواقعی که یک رشته مورد پذیرش ماشین واقع نمی‌شود:

۱. وقتی که ماشین به حالت مرده (dead stste) برسد.

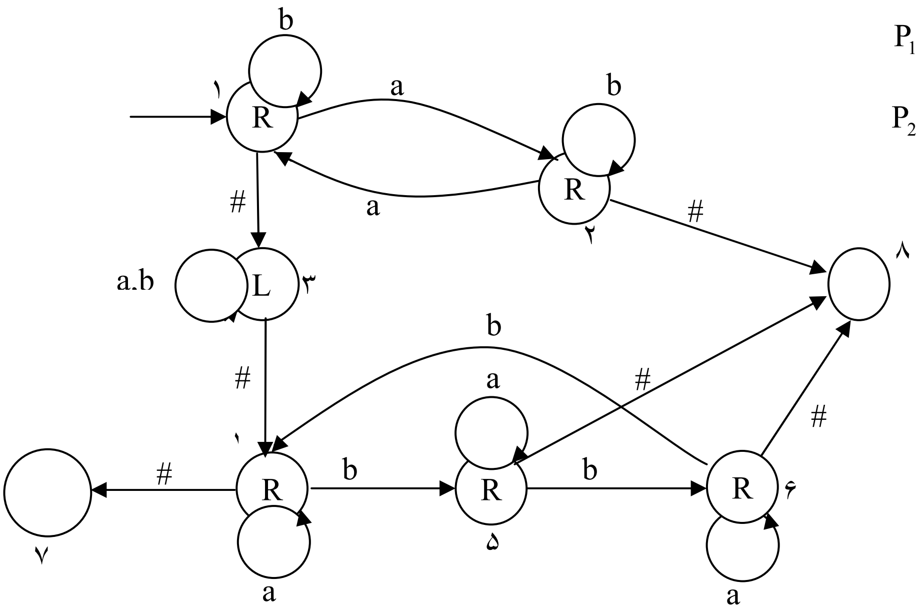
۲. ماشین در حلقه بیافتد. یعنی اینکه یک سیکل از حرکات را بصورت بینهایت بدون رسیدن به یک حالت پایانی تکرار کند.

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

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

$P_1 . N_a(w) \text{ mad } 2 = 0$

$P_2 . N_b(w) \text{ mad } 3 = 0$



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	$N_a(w)$ mad 2	$N_b(w)$ mad 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-$N_a(w)$ mod n=0

P1-$N_b(w)$ mod 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 رشتهٔ W که |w|=r، برابر 3r+3 است.

تعداد قدم‌ها در F.S.A برای پذیرش رشتهٔW که |w|=r، برابر r است.

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

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

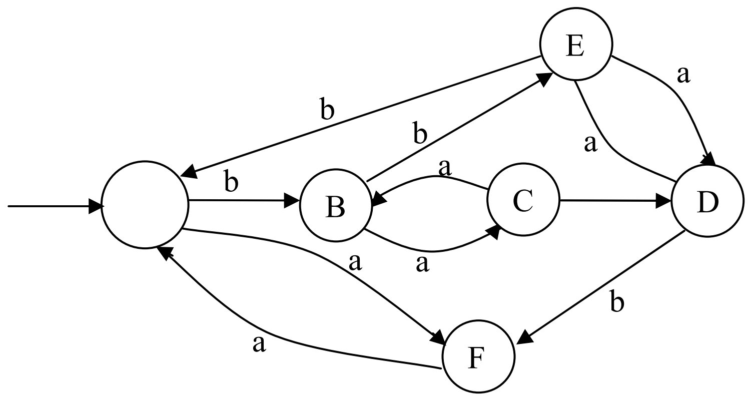
هیچ F.S.A با کمتر از 2r حالت برای L(r) وجود ندارد. (ثابت کنبد).

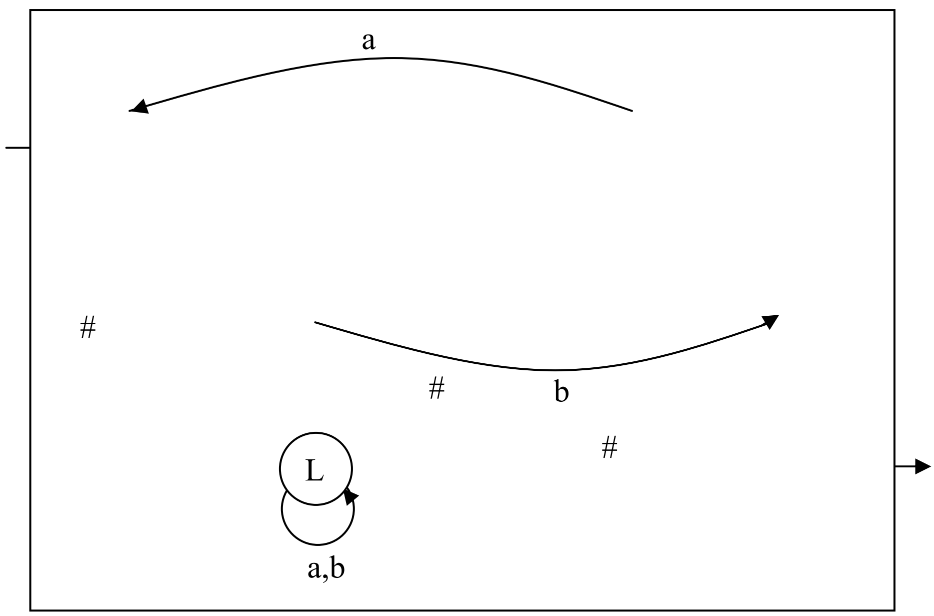
برای مقادیر مشخصی از r می‌توان یک T.W.A با تعداد حالات خیلی کمتر از 2r ساخت:

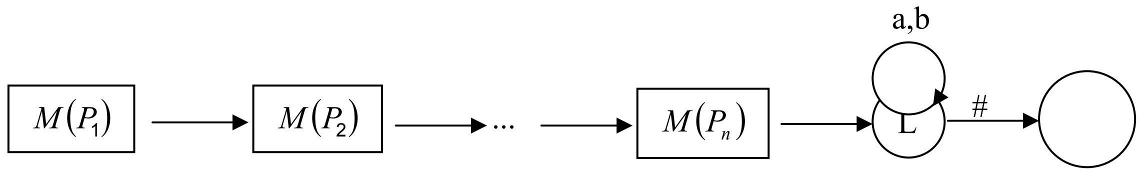
فرض کنیم $P_n,...,P_2,P_1$ اولین n عدد اول باشند. ماشینی مثل M می‌سازیم که برنامهٔ آن شامل n روتین مثل $M(P_n),...,M(P_2),M(P_1)$ باشد. هر روتین دو عمل انجام می‌دهد:

۱. یک پیمایش از چپ براست 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.


E
a
b
b
a
a
b
B
C
D
a
a
b
a
F


a
#
#
b
#
L
a,b


$M(P_1)$
$M(P_2)$
...
$M(P_n)$
a,b
L
#