جزوه درسی

نظریه زبانها و ماشینها

تالیف: مهندس حمید علمی

عضو هیئت علمی دانشگاه پیام نور مرکز آران و بیدگل

پیشگفتار

نظریه زبانها و ماشینها عنوان یکی از دروس تخصصی رشته مهندسی کامپیوتر (نرم افزار) و مهندسی فناوری اطلاعات می باشد.

آنچه در این مجموعه مشاهده می کنید مطالبی است که تحت عنوان جزوه نظریه زبانها و ماشینها در دانشگاه پیام نور آران و بیدگل تدریس می شود که با تلاش و کوشش انجمن علمی مهندسی کامپیوتر دانشگاه تدوین شده است.

لازم به ذکر است که این جزوه جایگزینی برای کتاب نیست بلکه با ارائه توضیحات بیشتر درمورد مطالب کتاب سعی در آسان سازی مفاهیم آن را دارد. امید است دانشجویان با بهره گیری از آن در کنار کتاب، مطالب را به خوبی درک کنند

از آنجایی که جزوه پیش رو اولین ویرایش جزوه نظریه زبانها و ماشینها می باشد بدیهی است خالی از اشکال نیست. بنابراین از دانشجویان عزیز تقاضا می کنم هرگونه اشکال و انتقاد خود را در جهت هرچه بهتر و پربارتر شدن این جزوه از طریق ایمیل **h_alemi_arani@yahoo.com** با اینجانب در میان بگذارند.

در آخر لازم است از زحمات دانشجوی گرامی، آقای سجاد سمیعی زاده که بنده را در نگارش و ویراستاری این جزوه یاری نمودند تشکر و قدردانی نمایم.

حمید عالمی آرانی

مهرماه ۹۰

اصطلاحات مربوط به زبانها

انواع زبانها عبارت اند از:

- زبانهای محاوره ای
- زبانهای ماشینی(کامپیوتری)
- زبانهای ریاضی

مفاهیم اصلی مربوط به زبانها شامل موارد زیر می باشد:

I. عبارتهای منظم
II. گرامرها
III. ماشینها(آتاماتاها)

الفبا:یک مجموعه متناهی و غیر تهی از سمبلهایی که به هر یک از آنها یک الفبا می گویند. معمولا یک مجموعه الفبا را با نماد $\sum$ نشان می دهند.

مثال) مجموعه الفبای زبان انگلیسی $\sum=\{a,b, ...,z,A,B, ...,Z\}$

مجموعه الفبای زبان برنامه نویسی $\sum=\{if,while,for,...\}$ C

رشته: دنباله ای است متناهی از عناصر الفبای یک زبان. معمولا در تعریف یک زبان از حروف انتهایی الفبای انگلیسی برای نشان دادن نام رشته ها استفاده می شود در بیشتر موارد یک رشته را با نماد w نشان داه می شود.

مثال) $\sum=\{a,b\} \rightarrow w=ab$

طول رشته: به تعداد سمبلهای تشکیل دهنده ی یک رشته طول رشته گفته می شود.طول رشته را معمولا به صورت |w| و گاهی اوقات به صورت n(w) نمایش می دهند.

مثال) $|w|=3$

تعداد الفبای مشخص در رشته: تعداد الفبایی مانند a در یک رشته را با یکی از دو نماد |w|a ویا na(w) نشان می دهند.

عملگرهای روی رشته ها

معکوس کردن یک رشته: معکوس رشته ی w را با نماد w^R نشان می دهند.

مثال) $w=abb \rightarrow w^R=bba$

الحاق دو رشته: الحاق دو رشته، رشته ای است که از پیوند دو رشته (دقیقا به همان ترتیب ذکر شده) به دست می آید.الحاق دو رشته w_1 و w_2 را به صورت $w_1.w_2$ نشان می دهند. (در بسیاری از موارد این الحاق به صورت w_1w_2 نشان داده می شود.)

مثال) $w_1=aa\ ,\ w_2=bb \ \rightarrow \ w_1.w_2=aabb$

$$w_1.w_2 \neq w_2.w_1$$

$$|w_1.w_2|=|w_1|+|w_2|$$

تکرار متناهی یک رشته: یک رشته را که به تعداد متناهی تکرار می شود با w^n نمایش می دهند که n یک عدد صحیح خواهد بود ، w^n به معنی الحاق n مرتبه رشته w با خودش می باشد.

مثال) $w^2=w.w$

$W^3=w.w.w=w^2.w$

$W=ab \ \rightarrow \ w^2=abab$

رشته تهی: رشته ای است به طول صفر که با نماد λ نشان داده می شود.

$$w.\lambda=\lambda.w=w$$

$$\lambda^R=\lambda$$

بستار یک رشته:

W^* : مجموعه ای از رشته هایی است که از تکرار ۰ بار یا بیشتر رشته ی w که با خود الحاق می شود، بدست می آید.

W^+ : مجموعه ای از رشته هایی است که از تکرار 1 بار یا بیشتر رشته ی w که با خود الحاق می شود، بدست می آید.

معمولا بستار برای رشته ها به کار نمی رود، بلکه برای مجموعه الفبایی یک زبان مورد استفاده قرار می گیرد.

مثال) اگر $\sum$ یک مجموعه الفبایی باشد، آنگاه: مجموعه تمامی رشته های روی $\sum$ به همراه $\lambda = \sum^*$

مجموعه تمامی رشته های روی $\sum$ بدون $\lambda = \sum^+$

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

$\sum^*$: فرض کنید $\sum$ یک مجموعه الفبا باشد آنگاه مجموعه $\sum^*$ مجموعه ای است که منطبق بر سه گام زیر باشد حاصل می شود :

۱. $\lambda \in \sum^*$

۲. if $a \in \sum$ then $a \in \sum^*$

۳. if $w \in \sum^*$ & $a \in \sum$ then $w_a \in \sum^*$

مثال) $\sum=\{a,b\}$

$\{\lambda , a , b , aa , ab , ba , bb , aaa , abb , \ldots\}$

مثال) اگر $\sum=\{a,b,c\}$ باشد، آنگاه $\sum^*$ به صورت زیر بدست می آید:

$\sum^{\cdot} = \{\lambda\}$

$\sum^{۱} = \{a,b,c\}$

$\sum^{۲} = \{aa,ab,ac,ba,bb,bc,ca,cb,cc\}$

$\sum^{۳} = .\ldots\ldots$

$\sum^{۴} = .\ldots\ldots$

پس خواهیم داشت:

$$\Sigma^*=\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots \Sigma^i \cup \ldots\ldots = \cup_{i=0}^{\infty} \Sigma^i$$

Σ^* مجموعه ای است که در ساخت آن از دو عملگر اجتماع(بین Σ^* ها) و الحاق(برای ساختن هر Σ^* ها) استفاده می شود.
هر رشته ای که از Σ^* انتخاب شود و با یکی از سمبلهای الفبای Σ الحاق شود رشته ی حاصل نیز در Σ^* خواهد بود.
حاصل الحاق هر دو رشته از مجموعه ی Σ^* در خود Σ^* خواهد بود.

زبان

به مجموعه ای از رشته ها که بر روی یک مجموعه الفبا ساخته می شوند، زبان می گویند.یک زبان می تواند متناهی یا نا متناهی باشد.معمولا یک مجموعه ی زبان را با حرف L نشان می دهند.

مثال)

$\Sigma = \{a,b\}$

$L_1=\{ab,aa,ba,bb\}$ زبان محدود

$L_2=\{\lambda,a,aa,aaa,\ldots..\}$ زبان نامحدود

نشان دادن یک زبان با مجموعه ی اعضایش گاهی ممکن است مبهم باشد، همچنین ممکن است به دلیل تعداد زیاد رشته های یک زبان محدود نشان دادن همه ی رشته هایش امکان پذیر نباشد، از این رو همواره زبانها را تعریف می کنند . تعریف یک زبان باید به گونه ای باشد که نه هیچ عضوی از مجموعه ی رشته ها کم شود ونه هیچ رشته ی اضافی را شامل شود.

مثال) L_1 مجموعه ی تمامی رشته هایی روی $\{a,b\}$ می باشد که دارای طول ۲ هستند.

$$\Sigma=\{a,b\} \ , \ L_1=\{w \in \Sigma^* \mid |w|=2\}$$

L_2 مجموعه ی تمامی رشته هایی روی $\{a,b\}$ است که دارای طول صفر یا بیشتر می باشند که هیچ یک از رشته ها سمبل b را ندارد.

$$L_2=\{a_n \mid n\geq 0\}$$

گاهی اوقات ممکن است رشته های زبان به گونه ای باشند که الفباها دارای شکل و قالب خاصی نباشند، در این گونه موارد باید حتما الفبا به همراه زبان نشان داده شوند.

مثال) L مجموعه تمامی رشته های دارای a,b که در آن تعداد a با تعداد b یکسان است.

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

در بعضی از زبانها ممکن است رشته های زبان دارای شکل و قالب خاصی از سمبل های الفبا باشند. از آنجایی که برای نشان دادن این قالب خاص از خود سمبل های الفبا استفاده می شود، نیازی به ذکر جداگانه مجموعه ی الفبا نیست.

مثال) L مجموعه ی تمامی رشته های دارای تعداد یکسان $\{a,b\}$ که همه ی a ها قبل از b ها باشند.

$$L=\{a_nb_n \mid n\geq 0\}$$

درتعریف زبانها یکی از دو شکل زیر ممکن است به کار رود:

$L1=\{w \in \{a,b\}^* \mid$

تا اینجا این تعریف بیان می کندکه رشته ی λ ممکن است عضو زبان باشد، بودن یا نبودن این رشته به شرط بستگی دارد .

$L2=\{w \in \{a,b\}+ \mid$

تا اینجا این تعریف بیان می کند که رشته ی λ عضو این زبان نخواهد بود.

عملگرهای روی زبان

اجتماع $L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \quad OR \quad w \in L_2\}$

اشتراک $L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \quad And \quad w \in L_2\}$

تفاضل $L_1 - L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ And } w \notin L_2\}$

الحاق $L_1 . L_2 = \{xy \in \Sigma^* \mid x \in L_1 \text{ And } y \in L_2\}$

معکوس $L^R = \{w \in \Sigma^* \mid w^R \in L\}$

بستار $L^* = L.L.L\ldots. = L_0 \cup L_1 \cup L_2 \cup \ldots.$

متمم $\overline{L} = \Sigma^* - L$

تکرار متناهی $L^n = L.L.L\ldots.L$

مثال) با در نظر گرفتن دو زبان $L_1=\{a^nb^n \mid n\geq 0\}$, $L_2=\{w \in \{a,b\}^* \mid n_a(w) = n_b(w)\}$ موارد خواسته شده را بدست آورید.

$L_1 \cup L_2 = L_2$

$L_1 \cap L_2 = L_1$

$L_1 - L_2 = \phi$

$L_2 - L_1 = \{w \in \{a,b\}^* \mid w \neq a^nb^n , n\geq 0 \ , \ n_a(w) = n_b(w) = w\}$

$L_1 . L_2 = L_2$

$L_2 . L_1 = L_2$

$L^R{}_1 = \{b^na^n \mid n\geq 0\}$

$L^R{}_2 = L_2$

$L^*{}_1 = \{(a^nb^m)^i \mid i\geq 0,$ درهرتکرار m می تواند متفاوت از تکرارهای قبلی باشد $\}$

$L^*{}_2 = L_2$

$\overline{L_1} = \{w \in \{a,b\}^* \mid w \neq a^nb^n, n\geq 0\}$

$\overline{L_2} = \{w \in \{a,b\}^* \mid n_a(w) = n_b(w)\}$

$L^2{}_1 = L_1.L_1 = \{a^nb^na^mb^m \mid n,m\geq 0\}$

$L^2{}_2 = L_2.L_2 = L_2$

عبارت های منظم (Regular Expression)

عبارت منظم ابزاری است که برای بعضی از زبانها توصیف می شود تا پذیرش یا عدم پذیرش یک رشته توسط یک زبان از آن استفاده می شود.عبارت منظم را تنها برای زبانهای منظم می توان نوشت.

یک عبارت منظم، از ۳ جزء زیر تشکیل شده است:

۱) الفبای زبان : هر یک از سمبل های الفبا و همچنین سمبل λ می تواند در یک عبارت منظم مورد استفاده قرار گیرد.

۲) عملگرهای زبان : سه نوع عملگر در عبارت های منظم به کار می رود:

اجتماع	$r_1 \mid r_2$ یا $r_1 + r_2$ (انتخاب r_1 یا r_2)
الحاق	$r_1 . r_2$ (که به صورت r_1r_2 نیز نوشته می شود.)
بستار	r^* به معنی تکرار صفر بار یا بیشتر عبارت r است.
	r^+ که به معنی تکرار یک بار یا بیشتر عبارت r است
	$r^?$ که به معنی تکرارصفربار یا یکبار عبارت r است

۳)علامت پرانتز برای دسته بندی

در یک عبارت منظم فقط موارد بالا هست و نه هیچ چیز دیگری.

فرض کنید Σ یک مجموعه الفبایی باشد، آنگاه یک عبارت منظم روی Σ از قوانین زیر ساخته می شود:

۱) ϕ و λ و $a \in \Sigma$ عبارت منظم هستند.

۲) اگر r_1 و r_2 دو عبارت منظم باشند، آنگاه r_1+r_2 ، r_2+r_1 ، r_1r_2 ، r_2r_1 ، r_1^* ، r_1^+ ، (r_1) ، r_2^* ، r_2^+ و (r_2) نیز عبارت های منظم خواهند بود.

۳) r یک عبارت منظم خواهد بود اگر و تنها اگر ا ز قانون ۱ وتکرار متناهی از قانون ۲ بدست آید.

مثال) برای هر یک از عبارت های زیر یک عبارت منظم مناسب بنویسید.

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

$L_2=\{a^n \mid n\geq 1\}$ a^+ یا aa^*

$L_3=\{a^n \mid n\geq 2\}$ aaa^* یا aa^+

$L_4=\{a^n \mid n=2k\}$ $(aa)^*$

$L_5=\{a^n \mid n=2k+1\}$ $a(aa)^*$

$L_6=\{a^n \mid n \bmod 3 = 0\}$ $(aaa)^*$

$L_7=\{a^n \mid n \bmod 3>0\}$ $(aaa)^*a+(aaa)^*aa$ یا $(aaa)^*(a+aa)$

$L_8=\{w \in \{a,b\}^* \mid |w|\geq 0\}$ $(a+b)^*$

$L_9=\{a^nb^m \mid n=2k , m=2k+1\}$ $(aa)^*(bb)^*b$

$L_{10}=\{w \in \{a,b\}^* \mid$ رشته w با a شروع و به b ختم می شود$\}$ $a(a+b)^*b$

$L_{11}=\{w \in \{a,b\}^* \mid$ رشته w با a شروع یا به bb ختم می شود$\}$ $a(a+b)^*+(a+b)^*bb$

$L_{12}=\{w \in \{a,b\}^* \mid$ رشته w دارای حداقل یک a باشد$\}$ $(a+b)^*a(a+b)^*$ یا $b^*a(a+b)^*$ یا $(a+b)^*ab^*$

$L_{13}=\{uwv \mid u,v,w \in \{a,b\}^* \ |u|=|v|=2\}$ $(a+b)(a+b)\ (a+b)^*\ (a+b)(a+b)$ یا

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

$L_{14}=\{vwv \mid v,w \in \{a,b\}^* , |v|=2\}$

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

$L_{15}=\{w \in \{a,b,c\}^* \mid$ طول رشته w زوج بوده ودارای دقیقا یک سمبل a باشد$\}$

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

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

$L_{16}=\{w \in \{a,b\}^* \mid$ {رشته w با a شروع وطول آن زوج باشد

$$a((a+b)(a+b))^*(a+b)$$

$L_{17}=\{w \in \{a,b\}^* \mid$ {رشته w دارای هیچ زیر رشته aa نباشد

$$a(ab+b)^*(a+b+\lambda) \quad \text{یا} \quad (ab+b)^*(a+\lambda) \quad \text{یا} \quad (a+\lambda)(ba+b)^*$$

$L_{18}=\{w \in \{0,1\}^* \mid$ { رشته w دارای دقیقا یک جفت صفر متوالی باشد

$$(01+1)^*00(10+1)^*$$

گرامرها

گرامرها ابزارهایی برای نشان دادن رشته های یک زبان هستند .هر گرامر برای توصیف یک زبان نوشته می شود. و به گونه ای است که تمامی رشته های آن زبان را می پذیرد . گرامر ابزاری است که برای همه ی انواع زبانها می توان طراحی کرد.

هر گرامر یک چهارتایی به فرم $G=(V,\Sigma,P,S)$ یا $G=(V,T,P,S)$ می باشد که در آن :

V : یک مجموعه متناهی و غیر تهی ازنمادهایی است که به هریک از آنها یک متغیر یا نانترمینال یا غیرپایانی می گویند.

Σ : یک مجموعه متناهی و غیر تهی از نمادهایی است که به هر یک از آنها یک الفبا یا ترمینال یا پایانی می گویند.(گاهی در تعریف گرامرها این مجموعه با T نشان داده می شود.)

P : مجموعه قواعد تولید گرامر است که هر قاعده به شکل روبه رو خواهد بود: $x \rightarrow y$

دنباله به طول یک یا بیشتر از الفبا ها یا متغیرها $x \in (V \cup T)^+$

دنباله به طول صفر یا بیشتر از الفبا ها یا متغیرها $y \in (V \cup T)^*$

$S \in V$: متغیر شروع گرامر خواهد بود .

معمولا برای نوشتن گرامر ها از یکسری قراردادها استفاده می شود که عبارتند از :

- معمولا از حروف بزرگ انگلیسی و گاهی از عبارت ها به عنوان متغیر ها استفاده می شود .
- معمولا از حروف کوچک انگلیسی ویا ارقام و یا عملگر های ریاضی به عنوان سمبلهای الفبا استفاده می شود.

- معمولا قاعده یا قواعد مربوط به متغیر شروع در سطر اول قواعد نوشته می شود .

مثال)یک گرامر می تواند به فرم زیر باشد:

$$\{G = (S, A, B) \quad , \quad (a, b) \quad , \quad S = \begin{cases} S \to AB \\ A \to a \\ B \to b \end{cases} \}$$

معمولا هر گرامر را به طور خلاصه و تنها با قواعد تولیدش نشان می دهند.به طور مثال گرامر فوق را به صورت زیر نشان داده می شود:

$S \to AB$
$A \to a$
$B \to b$

قاعده تهی : هر قاعده به فرم $A \to \lambda$ دریک گرامررا قاعده تهی می نامند.معمولا دریک گرامر چنانچه دو قاعده به فرم $A \to \alpha$ و $A \to \beta$ را داشته باشیم ، این دو قاعده را که دارای سمت چپ یکسان و سمت راست های متفاوتی هستند به صورت $A \to \alpha \mid \beta$ نشان می دهند.

رشته در گرامر ها : به دنباله ای از سمبل هایی که در تمامی آنها الفبا یا ترمینال باشد رشته می گویند.

شبه جمله در گرامر ها : به دنباله ای از سمبل های پایانی و غیر پایانی شبه جمله گفته می شود.

رشته و شبه جمله در توصیف اشتقاق به کار می روند.

پذیرش یک رشته توسط یک گرامر

:به عمل جایگذاری متغیر های یک گرامر که از متغیر شروع آغاز شده و با توجه به قوانین گرامر به تولید یک رشته می انجامد، اشتقاق(**derivation**) می گویند.در هر مرحله از اشتقاق تنها می توان یکی از متغیرها را جایگذاری نمود.برای نشان دادن پذیرش یک رشته توسط یک گرامر از اشتقاق استفاده می شود .

مثال) با توجه به گرامر زیر و با استفاده از عمل اشتقاق یک رشته دلخواه از گرامر تولید کنید

G: $S \to AB$
$A \to a$
$B \to b$

$S \to AB \to aB \to ab$

درهر عمل اشتقاق برای تولید یک رشته ممکن است چندین جمله وجود داشته باشد اما تنها یک رشته خواهیم داشت.
به تعداد مراحل جایگذاری برای تولید یک رشته ، تعداد مراحل اشتقاق می گویند.

انواع اشتقاق

دو نوع اشتقاق خاص وجود دارد:

۱) اشتقاق از چپ (سمت چپ ترین اشتقاق) :

نوعی از اشتقاق است که در آن برای تولید یک رشته همواره در هر مرحله از اشتقاق سمت چپ ترین متغیر برای جایگذاری انتخاب می شود.

۲) اشتقاق از راست (سمت راست ترین اشتقاق) :

نوعی از اشتقاق است که در آن برای تولید یک رشته همواره در هر مرحله از اشتقاق سمت راست ترین متغیر برای جایگذاری انتخاب می شود.

مثال)

$S \rightarrow AB \rightarrow aB \rightarrow ab$ چپ از اشتقاق

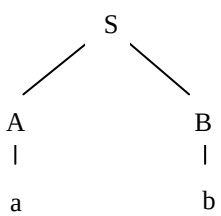
$S \rightarrow AB \rightarrow Ab \rightarrow ab$ راست از اشتقاق

درخت اشتقاق(derivation tree)

به نمایش درختی عمل اشتقاق، درخت اشتقاق می گویند،که در رسم آن باید به موارد زیر توجه داشت:

۱) در هر گره از درخت اشتقاق تنها یک سمبل چه پایانی و چه غیر پایانی قرار می گیرد.
۲) ریشه درخت اشتقاق متغیر شروع گرامر است.
۳) متغیرهای گرامر در گره های داخلی درخت اشتقاق خواهند بود .
۴) پایانی های گرامر و همین طور سمبل λ در برگ های درخت خواهند بود .
۵) برای بسط دادن یک متغیر موجود در یک گره تنها می توانیم یکی از قواعد آن متغیر را استفاده کنیم .
۶) رشته حاصل از پایانی های موجود در برگ ها از چپ به راست همان رشته ای است که توسط گرامرتولید می شود .

مثال) عمل اشتقاق گرامرمثال قبل را با درخت اشتقاق نشان می دهیم.



زبان یک گرامر

زبان یک گرامر مجموعه ی تمامی رشته هایی است که توسط آن گرامر پذیرفته می شود و آن را با L(G) نشان می دهند.به عبارت دیگر $L(G) =\{w \in \Sigma^* \mid S^* \rightarrow w\}$.

قانون بازگشتی : هر قاعده به فرم $A \rightarrow \alpha A \beta$ که در آن $\alpha,\beta \in (V \cup T)^*$ در یک گرامر، یک قاعده بازگشتی می باشد . قواعد بازگشتی باعث می شوندکه زبان یک گرامر نامحدود باشد.

مثال) با توجه به گرامر روبه‌رو

$$G: S \rightarrow AB$$
$$A \rightarrow \alpha A \mid \alpha$$
$$B \rightarrow bB \mid \lambda$$

الف) نشان دهید رشته ی w=aaabb توسط گرامر پذیرفته می شود.

ب) زبان گرامر را بدست آورید.

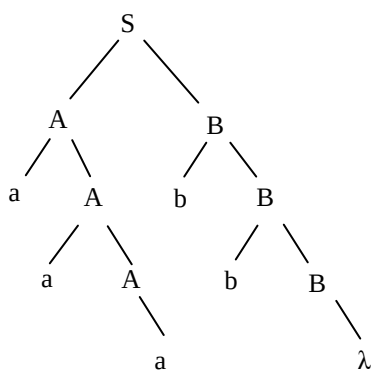
الف) $S \rightarrow AB \rightarrow aAB \rightarrow aaAB \rightarrow aaaB \rightarrow aaabB \rightarrow aaabbB \rightarrow aaabb$

ب)

$$S \rightarrow \underline{A}B \rightarrow a\underline{A}B \rightarrow aa\underline{A}B \rightarrow a\ldots a\underline{A}B \rightarrow a^n\underline{A}B \rightarrow a^naB \rightarrow a^{n+1}\underline{B} \rightarrow a^{n+1}b\underline{B} \rightarrow a^{n+1}b\ldots.b\underline{B}$$
$$\rightarrow a^{n+1m}B \rightarrow a^{n+1}b^m$$

$$L=\{a^{n+1}b^m \mid n \geq 0 , m>0\}$$

و درخت اشتقاق این مثال:



مثال) زبان گرامر زیر را بدست آورید

$$G: S \rightarrow aSa \mid A$$
$$A \rightarrow bbA \mid b$$

$$S \rightarrow aSa \rightarrow a\ldots aSa\ldots a \rightarrow a^nSa^n \rightarrow a^nAa^n \rightarrow a^nbbAa^n \rightarrow a^nbbbbAa^n \rightarrow a^n(bb)^mAa^n \rightarrow a^n(bb)^mba^n \rightarrow a^nb^{2m}ba^n \rightarrow a^nb^{2m+1}a^n$$

$L=\{a^n b^{2m+1} a^n \mid m,n\geq 0\}$

مثال) زبان گرامر زیر را بدست آورید.

G: $S\rightarrow A|B$

$A\rightarrow aAb|C$

$C\rightarrow aC|a$

$B\rightarrow aBbD$

$D\rightarrow bD|b$

$S\rightarrow A\rightarrow aAb \rightarrow a^nAb^n\rightarrow a^nCb^n \rightarrow a^naCb^n \rightarrow$

$a^na^mCb^n \rightarrow a^na^mab^n \rightarrow a^na^mb^n \rightarrow a^pb^q$ ($m>=0$, $m>=1$, $p>q$)

$B\rightarrow aBb \rightarrow aaBbb\rightarrow a^iBb^i\rightarrow a^iDb^i \rightarrow a^ibDb^i\rightarrow a^ib\ldots bDb^i$

$\rightarrow a^ib^jDb^i\rightarrow a^ib^jbb^i \rightarrow a^ib^jb^i\rightarrow a^xb^y \quad x<y$ ($J>=0$, $J>=1$)

$a^pb^q \cup a^xb^y \rightarrow L=\{a^nb^m \mid n\neq m\}$

انواع گرامرها

گرامرها بر طبق دسته بندی چامسکی در چهار گروه طبقه بندی می شوندکه عبارت اند از:

۱) گرامرهای منظم یا باقاعده (Regular):

یک گرامر منظم یا باقاعده خواهد بود اگر و تنها اگر خطی از راست یا خطی از چپ باشد.

گرامرهای خطی از راست: یک گرامر خطی از راست خواهد بود اگر و تنها اگرهر قاعده آن به یکی از دو شکل زیر باشد:

$$\begin{cases} A \rightarrow \alpha B & A,B \in V \\ A \rightarrow \alpha & \alpha \in T^* \end{cases}$$

گرامرهای خطی از چپ: یک گرامر خطی از چپ خواهد بود اگر و تنها اگر هر قاعده آن به یکی از دو شکل زیر باشد:

$$\begin{cases} A \rightarrow B\alpha & A,B \in V \\ A \rightarrow \alpha & \alpha \in T^* \end{cases}$$

مثال) کدامیک از گرامرهای زیر منظم است؟

G_1: $S\rightarrow AB$, $A\rightarrow a$, $B\rightarrow b$

G_2: $S\rightarrow aSb|a$

G_3: $S\rightarrow aA$, $A\rightarrow Bb$, $B\rightarrow b$

G_4: $S\rightarrow aA$, $A\rightarrow aB|b$, $B\rightarrow \lambda$

G4 خطی از راست است، پس گرامری منظم می باشد.

برای منظم بودن یک گرامر باید تمامی قواعد آن، خطی از راست ویا خطی از چپ باشند.

۲) گرامر های مستقل از متن (Context free) :

یک گرامر مستقل از متن خواهد بود، اگر و تنها اگر هر قاعده ی آن به شکل زیر باشد:

$A \rightarrow \alpha \qquad A \in V$
$\alpha \in (V \cup T)^*$

با توجه به تعاریف گفته شده می توان نتیجه گرفت هر گرامر منظم یک گرامرمستقل از متن خواهد بود یا به عبارت دیگر گرامر های منظم زیر مجموعه ی گرامر های مستقل از متن هستند.

۳) گرامر های حساس به متن (Context Sensitive) :

یک گرامر حساس به متن خواهد بود اگر وتنها اگر هر قاعده آن به شکل زیر باشد:

$\alpha \rightarrow \beta \qquad \alpha \in (V \cup T)^+$
$\beta \in (V \cup T)^+$
$|\alpha| \leq |\beta|$

قواعد تهی جزء قواعد گرامرهای حساس به متن نیستند.بنابراین نمی توان ادعا کرد که همه ی گرامرهای مستقل از متن گرامرهایی حساس به متن نیز هستند، بلکه می توان گفت هرگرامر مستقل از متن بدون قاعده ی تهی یک گرامر حساس به متن است.

۴) گرامرهای بدون محدودیت (Unrestrained) :

یک گرامر بدون محدودیت، گرامری است که هر یک از قواعدش به فرم زیر باشد:

$\alpha \rightarrow \beta \qquad \alpha \in (V \cup T)^+$
$\beta \in (V \cup T)^*$

مثال) برای هریک از زبان های زیر یک گرامر مناسب طراحی کنید.

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

$S \rightarrow aS \mid \lambda$

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

$S \rightarrow aS \mid a$

$L_3=\{a^n \mid n=2k\}$

$S \rightarrow aaS \mid \lambda$

$L_4=\{a^n \mid n=2k+1\}$

$S \rightarrow aaS \mid a$

$L_5=\{a^n b^m \mid n \geq 0 \ , m \geq 1\}$

$S \rightarrow AB$

$A \rightarrow aA \mid \lambda$

$B \rightarrow bB \mid b$

$S \rightarrow aS \mid A$

$A \rightarrow bA \mid b$

$L_6=\{a^nb^m \mid$ فرد و m زوج باشد n$\}$

$S\rightarrow AB$

$A\rightarrow aaA|a$

$B\rightarrow bbB|\lambda$

$S\rightarrow aaS|aA$

$A\rightarrow bbA|\lambda$

$L_7=\{a^nb^m \mid n\geq 0, m\geq 4\}$

$S\rightarrow AB$

$A\rightarrow aA|\lambda$

$B\rightarrow bB|bbbb$

$L_8=\{a^nb^m \mid n+m=2k\}$

$S\rightarrow AB|CD$

$A\rightarrow aaA|\lambda$

$B\rightarrow bbB|\lambda$

$C\rightarrow aaC|a$

$D\rightarrow bbD|b$

$L_9=\{a^nb^n \mid n\geq 0\}$

$S\rightarrow aSb|\lambda$

$L_{10}=\{a^nb^n \mid n\geq 1\}$

$S\rightarrow aSb|ab$

$L_{11}=\{a^{2n}b^{2n} \mid n\geq 0\}$

$S\rightarrow aaSbb|\lambda$

$L_{12}=\{a^{2n+1}b^{2n+1} \mid n\geq 0\}$

$S\rightarrow aaSbb|ab$

$L_{13}=\{a^nb^{2n} \mid n\geq 0\}$

$S\rightarrow aSbb|\lambda$

$L_{14}=\{a^nb^m \mid n<m\}$

$S\rightarrow AB$

$A\rightarrow aAb|\lambda$

$B\rightarrow bB|b$

$L_{15}=\{a^nb^m \mid n\leq m\}$

$S\rightarrow aSb|A$

$A\rightarrow bA \mid \lambda$

$L_{16}=\{a^nb^m \mid n\neq m\}$

$S\rightarrow aSb|A|B$

$A\rightarrow aA|a$

$B\rightarrow bB|b$

$L_{17}=\{ a^nb^mC^n \mid n,m>0 \}$	$S\rightarrow aSc \mid A$ $A\rightarrow bA \mid \lambda$
$L_{18}=\{ a^nb^mC^k \mid k=n+m \}$	$S \rightarrow aSc \mid A$ $A\rightarrow bAc \mid \lambda$
$L_{19}=\{ a^nb^mC^k \mid m=n+m \}$	$S\rightarrow AB$ $A\rightarrow aAb \mid \lambda$ $B\rightarrow bBc \mid \lambda$
$L_{20}=\{ a^nb^mC^k \mid n+m=2k \}$	$S\rightarrow aaSc \mid A \mid abAc$ $A\rightarrow bbAc \mid \lambda$
$L_{21}=\{ a^nb^mC^k \mid m=2q+1 ,\ n+k=2q' \}$	$S\rightarrow ABC \mid A'B\ C'$ $A\rightarrow aaA \mid \lambda$ $B\rightarrow bbB \mid b$ $C\rightarrow ccC \mid \lambda$ $A'\rightarrow aaA' \mid a$ $C'\rightarrow ccC' \mid c$
$L_{22}=\{ ww^R \mid w \in \{a,b\}^*\}$	$S\rightarrow aSa \mid bSb \mid \lambda$
$L_{23}=\{ ww^R \mid w \in \{a,b\}^+\}$	$S\rightarrow aSa \mid bSb \mid aa \mid bb$
$L_{24}=\{ w=w^R \mid w \in \{a,b\}^*\}$	$S\rightarrow aSa \mid bSb \mid a \mid b \mid \lambda$
$L_{25}=\{ a^nwc^{2k+1}w^Rb^n \mid n,k\geq 0 ,\ w \in \{a,b\}^+\}$	$S\rightarrow asb \mid A$ $A\rightarrow aAa \mid bAb \mid aBa \mid bBb$ $B\rightarrow ccB \mid c$
$L_{26}=\{ w \in \{a,b\}^* \mid n_a(w)=n_b(w)\}$	$S\rightarrow aSb \mid bSa \mid SS \mid \lambda$

نرمال سازی گرامرها

برای گرامرهای مستقل از متن دو فرم نرمال به نام های فرم نرمال چامسکی(CNF) و فرم نرمال گریباخ توصیف می شود.برای نرمال سازی یک گرامر به هر یک از این دو شکل لازم است بعضی از محدودیت ها در مورد قواعد تولید یک گرامر رعایت شود. این محدودیت ها در قالب ساده سازی گرامرها مطرح می شوند، بنابراین ساده سازی یک

گرامر به معنی اعمال محدودیت هایی روی آن خواهد بود، که هر چند ممکن است تعداد قواعد تولید را افزایش دهد اما غالبا عمل اشتقاق برای تولید یک رشته را کوتاهتر می کند.

ساده سازی گرامر ها

۱) حذف متغیر بازگشتی آغازین :

اگر متغیر S در یک گرامر متغیر شروع باشد و برای آن قاعده ای مانند $S\rightarrow\alpha S\beta$ که در آن $\alpha , \beta \in (V \cup T)^*$ را داشته باشیم، آنگاه متغیر شروع به صورت بازگشتی خواهد بود.
برای حذف متغیر بازگشتی آغازین تنهاکافی است یک متغیر مانند S' تعریف کنیم و قاعده ای مانند $S\rightarrow S'$ به گرامر اضافه کنیم و در گرامر جدید متغیر S' را متغیر شروع در نظر بگیریم.

مثال)

$S\rightarrow aSb|A$
$A\rightarrow aA|\lambda$

حذف متغیر بازگشتی آغازین

$\longrightarrow$

$S\rightarrow S$
$S\rightarrow aSb|A$
$A\rightarrow aA|\lambda$

۲) حذف قوانین تولید λ :

هر قاعده به فرم $A\rightarrow\lambda$ که در آن $A \in V$ باشد را یک قاعده تولید λ (قانون تهی) می نامند.حذف قوانین تولید λ در یک گرامر باعث می شود، تا اشتقاق کوتاهتری برای رسیدن به یک رشته داشته باشیم. به متغیرهایی که در یک گرامر مستقل از متن می توانند λ تولید کنند اصطلاحا متغیرهای میرا یا nullable گفته می شوند.حذف قوانین تولید λ در یک گرامر مستقل از متن شامل دو مرحله است :

- یافتن متغیرهای nullable
- حذف متغیرهای nullable

یافتن متغیر های nullable :

متغیرهایی که در یک گرامر می توانند λ تولید کنند بر دو نوع اند؛ دسته اول متغیرهایی که به طور مستقیم λ تولید می کنند و دسته دوم متغیرهایی که به طور غیرمستقیم λ تولید خواهند کرد.

برای یافتن متغیرهای nullable یک گرامر می توانیم از الگوریتم زیر استفاده کنیم:

۱. $NULL=\{A|A\rightarrow\lambda \in P\}$, $PREV=\{\}$
۲. اگر $PREV \neq NULL$ برو به مرحله بعدی وگرنه برو به مرحله ۶ .

۳. PREV← NULL

۴. تمامی متغیرهایی که شرایط روبه رو را دارند به NULL اضافه کن : $\{X \mid X \rightarrow \alpha \in P, \alpha \in \{PREV\}^*\}$

۵. برو به مرحله ۲ .

۶. پایان .

مثال) در گرامر زیر متغیرهای nullable را بیابید.

	PREV	NULL	
$S \rightarrow aA \mid BC$	{}	{B}	
$A \rightarrow Aa \mid Bb$	{B}	{B,C}	nullable={B,C,S}
$B \rightarrow a \mid \lambda$	{B,C}	{B,C,S}	
$C \rightarrow b \mid B$	{B,C,S}	{B,C,S}	

حذف متغیرهای nullable :

برای حذف nullable هایی که یک گرامر تمامی قوانین آن بازنویسی می شوند (قوانین تهی ذکر نخواهند شد) بدین صورت که: $A \rightarrow x_1, x_2, \ldots\ldots, x_n \longrightarrow A \rightarrow \alpha_1, \alpha_2, \ldots\ldots, \alpha_n$

با در نظر گرفتن این شرایط که :

۱. اگر $x_i \in$ nullable باشد که آنگاه $\alpha_i = x_i$ یا $\alpha_i = \lambda$.

۲. اگر $x_i \notin$ nullable باشد که آنگاه $\alpha_i = x_i$.

۳. تمامی α_i ها نباید با هم λ شوند.

۴. اگر $S \in$ nullable باشددرانتها باید $S \rightarrow \lambda$ به گرامر اضافه شود.

مثال) درگرامر مثال قبل متغیرهای nullable را حذف کنید.

$S \rightarrow aA \mid BC$		$S \rightarrow aA \mid BC \mid C \mid B \mid \lambda$
$A \rightarrow Aa \mid Bb$	nullable={B,C,S}	$A \rightarrow Aa \mid Bb \mid b$
$B \rightarrow a \mid \lambda$	$\longrightarrow$	$B \rightarrow a$
$C \rightarrow b \mid B$		$C \rightarrow b \mid B$

مثال) قوانین λ را در گرامر زیر حذف کنید.

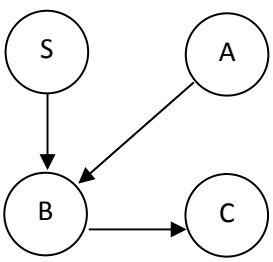
	PREV	NULL	
S→ABaC	{}	{B,C}	
A→BC	{B,C}	{B,C,A}	nullable={B,C,A}
B→d | λ	{B,C,A}	{B,C,A}	
C→D | λ			
D→d			

S→ABaC | BaC | AaC | ABa | aC | Ba | Aa | a
A →BC | C | B
B →d
C→D
D→d

۳) حذف قوانین زنجیر :

به قوانینی مانندA→Bکه در آن A,B ϵ V باشد در اصطلاح قوانین زنجیر یا واحد یا یکه می گویند. از آن جایی که در قوانین زنجیر نه الفبایی در سمت راست قانون وجود دارد ونه اینکه تعداد متغیرها را تغییر می دهد، لذا استفاده از آنها در عمل اشتقاق تنها باعث افزایش تعداد مراحل اشتقاق می شود . بنابراین برای کوتاهتر شدن اشتقاق قوانین زنجیر یک گرامر را حذف می کنیم.

روند حذف زنجیر: برای حذف قوانین زنجیر در یک گرامر ابتدا تمامی زنجیر های آن را شناسایی می کنیم و با استفاده از متغیرهای به کار رفته در قوانین زنجیر یک گراف وابستگی رسم می کنیم، به گونه ای که متغیر های به کار رفته در قواعد زنجیر راس های گراف و خود قواعد زنجیر یال های گراف باشند.سپس با توجه به گراف تمامی مسیرها از یک راس به راس دیگر را شناسایی می کنیم. برای بدست آوردن گرامر حاصل ابتدا گرامر اولیه را بدون هیچ یک از قواعد زنجیرش می نویسیم، سپس با توجه به هر یک از مسیرها و با استفاده از گرامر جدید اگر مسیری به صورتX→Y (از X به Y) وجود دارد هر آنچه Y تولید می کند، برای Xاضافه می‌کنیم.



مثال) در گرامر زیر قوانین زنجیر را حذف کنید.

S→aA | B
A→a | B
B→bB | C

زنجیرها :
S →B
A→B

C→b

B→ C

مسیرها :

S →B

S→ C

A→ B

A→ C

B →C

S→aA | bB | b

A→a | bB | b

B→bB | b

C→b

مثال) با توجه به گرامر مثال حذف λ ، که متغیر λ در آن حذف شد قوانین زنجیر را حذف کنید.

S→ABaC | BaC | AaC | ABa | aC | Ba | Aa | a

A→BC | C | B

B→d

C→D

D→d

زنجیرها :

A→B

A→C

C→D

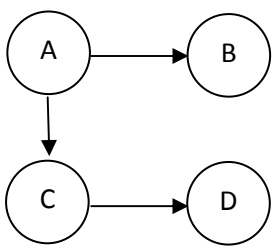
مسیرها :

A→B

A→C

A→D

C→D



S→ABaC | ABa | AaC | BaC | Aa | aC | Ba | a

A→BC | d

B→d

C→d

D→d

۴) حذف متغیرهای بدون استفاده (غیر مفید یا Useless) :

متغیر X را در یک گرامر مفید یا قابل استفاده می گوییم، هرگاه دو شرط زیر را داشته باشد:

۱. $X \in V , w \in \Sigma^*$ $X \rightarrow w$(از طریق متغیر X و طی چند مرحله ی اشتقاق بتوانیم به یک رشته الفبا برسیم.)

۲. $S \rightarrow \alpha X \beta \; S \in V , \alpha,\beta \in (V \cup \Sigma)^*$ (متغیر X از متغیر شروع قابل دسترس باشد.)

بنابراین در یک گرامر مستقل از متن متغیری را که به هیچ وجه به یک رشته از الفبا نرسد یا از متغیر شروع قابل دسترس نباشد متغیر بدون استفاده یا غیر مفید می نامیم. متغیرهایی را که به هیچ وجه نتوان از آنها به یک رشته از الفبا رسید متغیرهای بدون استفاده از جنبه اول و آنهایی را که از متغیر شروع نتوان به آنها رسید متغیر بدون استفاده از جنبه دوم می نامند.

در حذف متغیر های غیر مفید یک گرامر ابتدا باید متغیرهای غیرمفید از جنبه اول حذف شوند وسپس در گرامر حاصل متغیرهای غیرمفید از جنبه دوم حذف می شوند.

الگوریتم یافتن متغیرهای غیرمفید از جنبه اول :

۱. PREV={} و $TERM=\{A \mid A\rightarrow w \in P, w \in \Sigma^*\}$ (متغیرهایی که به طور مستقیم یک دنباله از الفبا را تولید می کنند.)
۲. اگر TERM≠PREV برو به مرحله بعدی و گرنه برو به مرحله ۶ .
۳. PREV← TERM .
۴. متغیرهایی که شرایط زیر را دارند به TERM اضافه کن :

$\{X \mid X\rightarrow \alpha \in P, \alpha \in (\Sigma \cup PREV)^*\}$

۵. برو به مرحله ی ۲ .
۶. متغیرهای غیر مفید از جنبه اول : {V}-{TERM}
۷. پایان .

مثال) متغیرهای غیرمفید از جنبه اول را در گرامر زیر پیدا کنید.

	PREV	TERM	
S→aS | A | C			
A→a	{}	{A,B}	
B→aa	{A,B}	{A,B,S}	{C}=متغیر غیرمفید از جنبه اول
C→aCb	{A,B,S}	{A,B,S}	

پس از یافتن یک متغیر غیرمفید چه از جنبه اول و چه از جنبه دوم هم تمامی قواعد مربوط به آن متغیر حذف می شوند و هم تمامی دنباله هایی که در سمت راست متغیرهای دیگر که دارای این متغیر باشند به طور کامل حذف می شوند .

مثال) متغیرهای غیرمفید از جنبه اول را در گرامر مثال قبل حذف کنید.

S→aS | A | C	S→aS | A
A→a	A→a
B→aa	B→aa
C→aCb	

الگوریتم یافتن متغیرهای غیرمفید از جنبه دوم :

۱. PREV={}و REACH={S} .

۲. اگر REACH≠PREV برو به مرحله بعدی و گرنه برو به مرحله ۷.

۳. NEW=REACH - PREV .

۴. PREV←REACH .

۵. برای تمامی متغیرهای موجود در NEW مانند X و تمامی قوانین به فرم X→α درگرامر متغیرهای جدید موجود در دنباله را به REACH اضافه کن .

۶. برو به مرحله ۲.

۷. متغیرهای غیر مفید از جنبه دوم : {V}-{REACH}

۸. پایان .

مثال) در گرامر حاصل از مثال قبل متغیرهای غیرمفید از جنبه دوم را بیابید و سپس حذف کنید.

S→aS | A	PREV	REACH	NEW
A→a	{}	{S}	_
B→aa	{S}	{S,A}	{S}
	{S,A}	{S,A}	{A}

{B}=متغیر غیرمفید از جنبه دوم

S→aS | A
A→a

مثال) در گرامر زیر متغیرهای غیر مفید را حذف کنید و سپس زبان گرامر را بدست آورید .

S→AC | BS | B	PREV	TERM
A→aA | aF	{}	{B,F}
B→CF | B	{B,F}	{B,F,S,A}
C→cC | D	{B,F,S,A}	{B,F,S,A,E}
D→aD | BD | C	{B,F,S,A,E}	{B,F,S,A,E}
E→aA | BSA		
F→bB | b		

{C,D}=متغیرهای غیرمفید از جنبه اول

پس از حذف متغیرهای غیرمفید از جنبه اول خواهیم داشت :

S→BS | B	PREV	REACH	NEW
A→aA | aF	{}	{S}	_

B→B	{S}	{S,B}	{S}	{A,E,F}=متغیرهای غیرمفید از جنبه دوم
E→aA | BSA	{S,B}	{S,B}	{B}	
F→bB | b				

و پس از حذف این متغیرها نیز خواهیم داشت :

S→BS | B

B→b

بنابراین زبان گرامر برابر است با :

$L(G)=\{b^n \mid n\geq 1\}$

۵) حذف بازگشتی از چپ :

هر قاعده یک گرامر مستقل از متن می تواند به یکی از دو شکل زیر باشد :

$A\rightarrow\alpha_1 \mid \alpha_2 \ldots\ldots\ldots \mid \alpha_n$ $\quad A\subset V$

$A\rightarrow A\beta_1 \mid A\beta_2 \ldots\ldots\ldots \mid A\beta_m$ $\quad \alpha_i,\beta_j \in (V , \Sigma)^*$

شکل فوق بیان می کند که دنباله های سمت راست یک متغیر در یک گرامر مستقل از متن یا با همان متغیر شروع می شود و یا نه اگر در یک گرامر مستقل ازمتن برای متغیری مانند A حداقل یک قاعده ی تولید به شکلی که با A شروع شود داشته باشیم، می گوییم قواعد A دارای بازگشتی از چپ است .

پس از یافتن برای متغیری که دارای بازگشتی از چپ است با تعریف یک متغیر کمکی برای آن متغیر و با استفاده از فرم زیر بازگشتی از چپ را حذف می کنیم :

$A\rightarrow\alpha_i \mid \alpha_i A'$ $\quad 1\leq i \leq n$

$A\rightarrow\beta_j \mid \beta_j A'$ $\quad 1\leq j \leq m$

مثال) بازگشتی از چپ را در گرامر زیر حذف کنید .

S→aSb | aAB	S→aSb | aAB
A→aA | ABa | aAb	A→aA | aAA´ | aAb | aAbA´
B→Bab | BBa | bb	A´→Ba | BaA´
	B→bb | bbB´
	B´→ab | abB´ | Ba | BaB´

فرم نرمال چامسکی

یک گرامر به فرم نرمال چامسکی خواهد بود اگر وتنها اگر هر قاعده آن به یکی از دو شکل زیر باشد :

$A\rightarrow BC$ $\quad A,B,C \in V \quad , \quad a \in \Sigma$

$A\rightarrow a$

برای تبدیل یک گرامر مستقل از متن به فرم نرمال چامسکی لازم است ابتدا قوانین λ، قوانین زنجیر و متغیرهای بدون استفاده حذف شوند .

مثال) گرامر زیر را به فرم نرمال چامسکی بنویسید .

$S \rightarrow aSb \mid aA$
$A \rightarrow aA \mid a$
$S \rightarrow aSb$

$S \rightarrow x_1x_2 \mid x_3A$
$x_1 \rightarrow x_3S \mid a$
$x_2 \rightarrow b$
$x_3 \rightarrow a$
$A \rightarrow x_3A \mid a$

مثال) پس از حذف قوانین λ و زنجیر،گرامر زیر را به فرم نرمال چامسکی درآورید .

$S \rightarrow A \mid B$
$A \rightarrow aA \mid \lambda$
$B \rightarrow bB \mid b$

حذف λ ⟶

$S \rightarrow A \mid B \mid \lambda$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid b$

حذف زنجیر ⟶

زنجیرها
$S \rightarrow A$
$S \rightarrow B$

$S \rightarrow aA \mid a \mid bB \mid b \mid \lambda$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid b$

→

$S \rightarrow x_1A \mid a \mid x_2B \mid b \mid \lambda$
$x_1 \rightarrow a$
$x_2 \rightarrow b$
$A \rightarrow x_1A \mid a$
$B \rightarrow x_2B \mid b$

فرم نرمال گریباخ

یک گرامر به فرم نرمال گریباخ خواهد بود اگر و تنها اگر هر قاعده آن به فرم زیر باشد :

$A \rightarrow a\alpha$

$A \in V$
$a \in \Sigma$
$\alpha \in V^*$

برای تبدیل یک گرامر به فرم نرمال گریباخ باید قوانین λ، قوانین زنجیر ، متغیرهای بدون استفاده و بازگشتی از چپ را درآن حذف کنیم .

مثال) گرامر زیر را به فرم نرمال گریباخ در آورید .

$S \rightarrow aSb \mid AB$
$A \rightarrow aA \mid a$
$B \rightarrow bBb \mid bb$

→

$S \rightarrow aSx_1 \mid aAB \mid aB$
$x_1 \rightarrow b$
$A \rightarrow aA \mid a$
$B \rightarrow bBx_1 \mid bx_1$

الگوریتم عضویت برای گرامرهای مستقل از متن (CYK)

برای گرامرهای مستقل از متن الگوریتمی به نام الگوریتم عضویت وجود دارد که با دریافت گرامر در رشته ای مانند w پذیرش رشته توسط گرامر را بررسی می کند . برای اجرای این الگوریتم لازم است حتما گرامرمستقل از متن ورودی به فرم نرمال چامسکی باشد . همچنین الگوریتم از مرتبه $O(|w|3)$ می باشد .

فرض کنید رشته ی ورودی یک رشته ی w دارای سمبل های i ام تا j ام به شکل زیر باشد به طوری که :

$w_{ij}=a_i a_{i+1}.....a_j \qquad a_i \in \Sigma$

در اجرای الگوریتم یکسری مجموعه های V_{ij} محاسبه می کنیم که هر V_{ij} به صورت زیر تعریف می شود :

$$V_{ij} = \{A \mid A \overset{*}{\rightarrow} w_{ij}\}$$

یعنی هر Vij یک مجموعه شامل متغیرهایی از گرامر است که می توانند زیررشته ی شامل سمبل های i ام رشته ی مورد نظر را تولید کنند . برای تعیین مجموعه های Vij از قانون زیر استفاده می کنیم :

$$V_{ij} = \begin{cases} x \rightarrow w_{ij} \\ \\ \cup \{x \rightarrow yz \mid y \in V_{i\,k} \ , z \in V_{k+1\,j} \ , i \leq k < j\} \end{cases}$$

برای اجرای الگوریتم CYK چناچه رشته ای به طول n مانند w داده شود آن را به صورت رشته ی w1n (از سمبل اول تا n ام) در نظر می گیریم . سپس باتوجه به قانون گفته شده مجموعه های Vij را تعیین می کنیم . در تعیین مجموعه های Vij ابتدا Vij های تک عضوی (i=j) سپس Vij های دو عضوی و به همین ترتیب تا V1n را محاسبه می کنیم . در نهایت اگر متغیر شروع گرامر عضوی از مجموعه ی V1n باشد رشته ی موردنظر توسط گرامر پذیرفته می شود، در غیر این صورت رشته پذیرفته نخواهد شد .

مثال) با توجه به گرامر زیر و با استفاده از الگوریتم CYK بررسی کنید که آیا رشته ی w=aabb در زبان تولید شده گرامر وجود دارد یا خیر ؟

$S \rightarrow AB$
$A \rightarrow BB \mid a$
$B \rightarrow AB \mid b$

$V_{11}=\{A\} \quad V_{22}=\{A\} \quad V_{33}=\{B\} \quad V_{44}=\{B\}$

$$V_{12} \rightarrow \quad \overset{k=1}{x \rightarrow yz} \quad \begin{cases} y \in V_{11}=\{A\} \\ \\ z \in V_{22}=\{A\} \end{cases} \qquad V_{12}= \phi$$

$$V_{23} \rightarrow \quad \overset{k=2}{x \rightarrow yz} \quad \begin{cases} y \in V_{22}=\{A\} \\ \\ z \in V_{33}=\{B\} \end{cases} \qquad V_{23}= \{S,B\}$$

$$V_{34}\rightarrow \quad \overset{k=3}{x\rightarrow yz}\begin{cases} y\in V_{33}=\{B\} \\ z\in V_{44}=\{B\} \end{cases} \qquad V_{34}=\{A\}$$

$$V_{13}\begin{cases} \overset{k=1}{x\rightarrow yz}\begin{cases} y\in V_{11}=\{A\} \\ z\in V_{23}=\{S,B\} \end{cases} & \{S,B\} \\ \overset{k=2}{x\rightarrow yz}\begin{cases} y\in V_{12}=\phi \\ z\in V_{33}=\{B\} \end{cases} & \phi \end{cases} \qquad V_{13}=\{S,B\}$$

$$V_{24}\begin{cases} \overset{k=2}{x\rightarrow yz}\begin{cases} y\in V_{22}=\{A\} \\ z\in V_{34}=\{A\} \end{cases} & \phi \\ \overset{k=3}{x\rightarrow yz}\begin{cases} y\in V_{23}=\{S,B\} \\ z\in V_{44}=\{B\} \end{cases} & \{A\} \end{cases} \qquad V_{24}=\{A\}$$

$$V_{14}\begin{cases} \overset{k=1}{x\rightarrow yz}\begin{cases} y\in V_{11}=\{A\} \\ z\in V_{24}=\{A\} \end{cases} & \phi \\ \overset{k=2}{x\rightarrow yz}\begin{cases} y\in V_{12}=\phi \\ z\in V_{34}=\{A\} \end{cases} & \phi \\ \overset{k=3}{x\rightarrow yz}\begin{cases} y\in V_{13}=\{S,B\} \\ z\in V_{44}=\{B\} \end{cases} & \{A\} \end{cases} \qquad V_{14}=\{A\}$$

رشته ی w توسط گرامر پذیرفته نمی شود یا به عبارت دیگر : $S\in V_1\rightarrow w\in$ grammar

گرامر های مبهم

گرامری مبهم است که در آن حداقل یک رشته برای پذیرش توسط گرامر بیش از یک اشتقاق داشته باشد.

مثال) گرامر زیر یک گرامر مبهم است .

$S \rightarrow SS \mid ab \mid \lambda$

مثلا برای رشته w=ab دو اشتقاق خواهیم داشت :

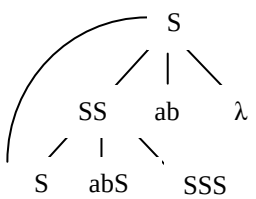
$S \rightarrow SS \rightarrow S \rightarrow ab$

$S \rightarrow SS \rightarrow SSS \rightarrow SabS \rightarrow abS \rightarrow ab$

یکی از روش های تشخیص گرامرهای مبهم استفاده از درخت یا گراف تجزیه است. اگر در تشکیل گراف تجزیه دور ایجاد شود آن گرامر مبهم خواهد بود.گراف با قوانین زیر رسم می شود :

۱. ریشه ی گراف متغیر شروع گرامر خواهد بود .
۲. در هر گره از گراف تجزیه یک دنباله ی دارای سمبل های پایانی و غیرپایانی خواهیم داشت .
۳. برای بست هر گاه سمت چپ ترین متغیر موجود در آن انتخاب می شود .
۴. برای بست متغیر انتخاب شده قطعی سمت راست های هر قاعده ی آن متغیر در گرامر جایگزین می شود .
۵. درخت تجزیه یک درخت پایانی ناپذیر است .

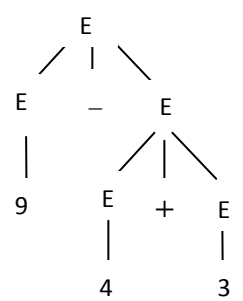
مثال) $S \rightarrow SS \mid ab \mid \lambda$

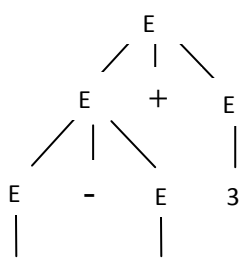


مثال) گرامری برای تولید عبارت های ریاضی شامل عملگرهای + و − و عملوند های تک رقمی داریم که به صورت زیر است :

$E \rightarrow E+E \mid E-E \mid 0 \mid 1 \mid 2 \mid \ldots. \mid 9 \mid$

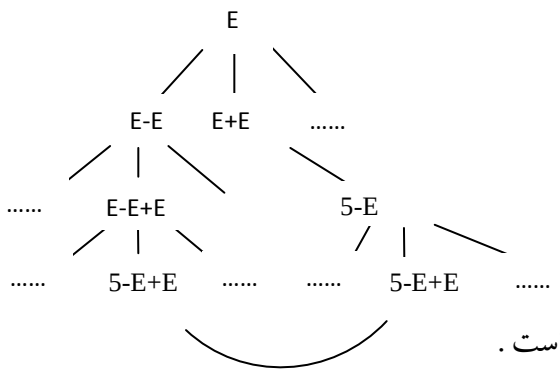
برای تولید رشته ۳+۴−۹





و خروجی دو درخت متفاوت می باشد.

مثال) آیا گرامر زیر مبهم است



چون دور حاصل می شود پس گرامر مبهم است .

ابهام یکی از ویژگی گرامر است، یعنی شاید بتوان برای زبانی که گرامر مبهم برایش نوشته شده است گرامر غیرمبهمی نیز ارایه کرد . اما بعضی از زبان ها هستند که به هیچ عنوان نمی توان گرامر غیر مبهمی برایشان ارایه داد که به آن ها زبانهای ذاتا مبهم می گویند.

گرامرهای خطی

یک گرامر خطی گرامر مستقل از متنی است که در سمت راست همه ی قواعد تولیدش حداکثر یک متغیر وجود داشته باشد.هر زبانی را که بتوان آن را حداقل با یک گرامر خطی توصیف کرد، زبان مستقل از متن خطی می نامند.

مثال)

$L_1=\{ww^R \mid w \in \{a,b\}^*\}$ $\quad S\rightarrow aSa \mid bSb \mid \lambda$

$L_2=\{a_nb_n \mid n \geq 0\}$ $\quad S\rightarrow aSb \mid \lambda$

$L_3=\{w \in \{a,b\}^* \mid n_a(w)=n_b(w)\}$ $\quad S\rightarrow aSb \mid bSa \mid S \mid \lambda$

زبان مستقل ازمتن خطی نیست

گرامرهای ساده

گرامر مستقل از متنی است که هریک از قواعد آن به شکل زیر باشد :

$A\rightarrow a\alpha \qquad A \in V , a \in \Sigma , \alpha \in V^*$

و همچنین هر زوج (A,a) حداکثر یکبار در گرامر تکرار شود .

مثال) در مورد گرامر های G_1 و G_2 کدام گزینه است ؟

$G_1=S\rightarrow aS \mid bSS \mid c$

$G_2=S\rightarrow aS \mid bSS \mid aSS \mid c$

الف) G_1 ساده ولی G_2 غیرساده است .

ب) G_1 غیر ساده و G_2 ساده است .

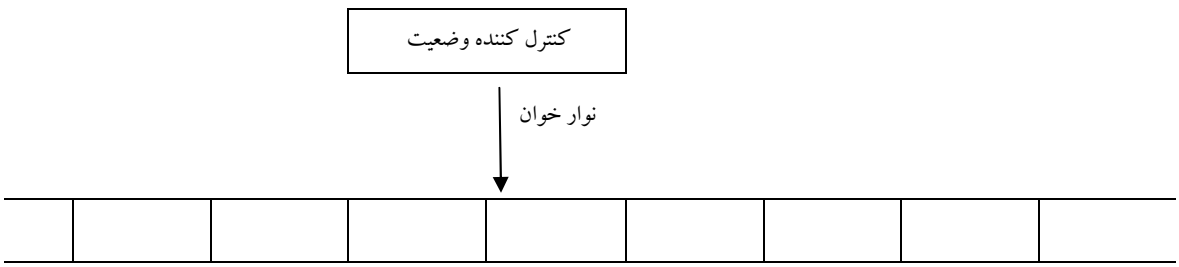
ج) هر دو گرامر ساده اند .

د) هر دو گرامر غیر ساده اند .

پاسخ : گزینه الف .

ماشین های متناهی (Final Automata)

ماشین های حالت متناهی، ماشین هایی هستند که برای پذیرش زبان های منظم طراحی می شوند . مدل انتزاعی هر ماشین متناهی دارای یک نوار ورودی، یک نوارخوان و یک کنترل کننده ی وضعیت به شکل زیر است :



رشته ی ورودی بر روی نوار ورودی قرار می گیرد و کنترل کننده ی وضعیت که تنها قادر به خواندن سمبل ها و تغییر وضعیت است با تغییر وضعیت های متوالی بررسی می کند که آیا رشته ی ورودی نوار توسط ماشین پذیرفته خواهد شد که در نوار خوان در انتهای رشته ی ورودی باشد و وضعیت ماشین یک وضعیت پایانی باشد .

$w\rightarrow$ | FA | $\rightarrow$ پذیرش یا عدم پذیرش رشته توسط ماشین

تعریف استاندارد ماشین متناهی قطعی

یک ماشین متناهی قطعی یا DFA یک پنج تایی به فرم $M=(Q,\Sigma,q,\delta,F)$ می باشد که در آن :

Q : یک مجموعه ی متناهی و غیرتهی از حالت ها یا وضعیت های ماشین می باشد .

Σ : یک مجموعه ی متناهی و غیرتهی از الفبای زبان ماشین می باشد .

$q \in Q$: بیانگر حالت شروع است .

F : یک مجموعه ی غیر تهی از حالت های پایانی ماشین است .

δ : مجموعه ی توابع انتقال ماشین است که هر تابع آن به شکل زیر توصیف می شود :

$\delta : Q * \Sigma \rightarrow Q$

هر تابع انتقال بیان می کند که هر حالت از ماشین با هر یک از سمبل های الفبا به چه حالت دیگری از ماشین منتقل خواهد شد . برای

مثال $\delta(qi,a)=qj$ بیان می کند که حالت qi با سمبل های الفبایی a به حالت qj منتقل می شود .

مثال) یک ماشین می تواند به صورت زیر باشد :

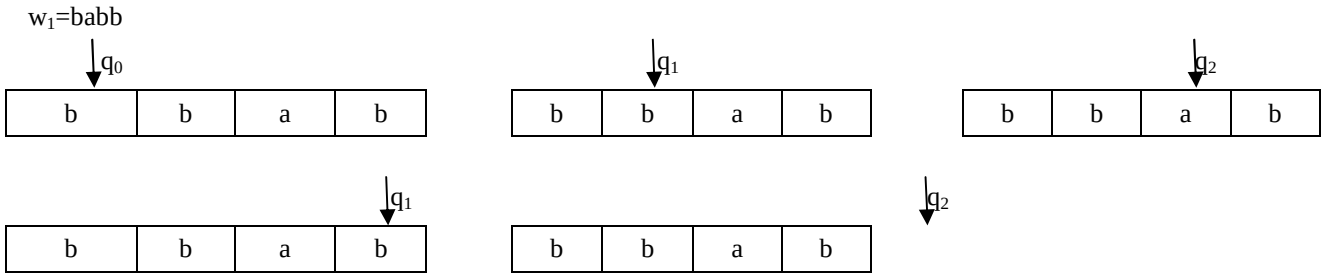
$M=(Q,\Sigma,q,\delta,F)$ $Q=\{q_0,q_1,q_2\}$ $\Sigma=\{a,b\}$ $q=q_0$ $F=\{q_2\}$

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

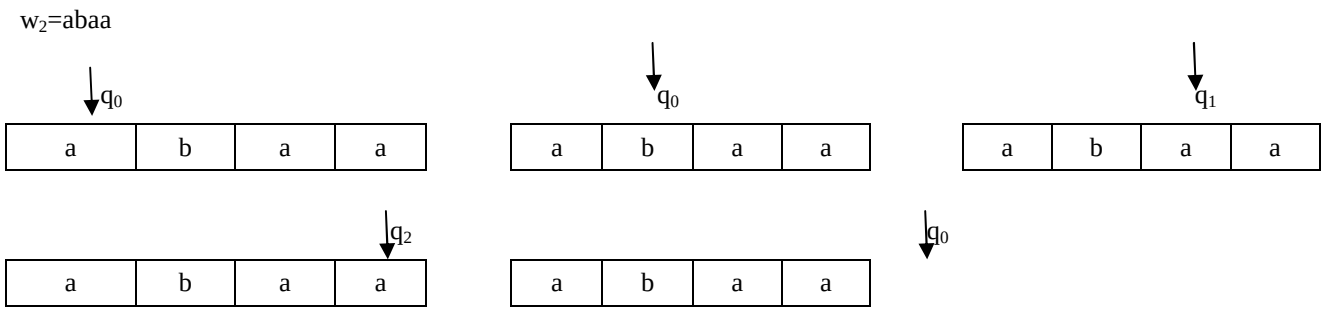
شرایط پذیرش

یک رشته زمانی توسط یک ماشین پذیرفته می شود که با در نظر گرفتن آن رشته به عنوان ورودی و با شروع و اعمال توابع انتقال زمانی که به پایان رشته می رسیم ماشین در یک حالت پایانی باشد .

مثال) با توجه به مثال قبل پذیرش رشته های w را بررسی کنید .



چون q_2 حالت پایانی است w_1 توسط ماشین پذیرفته می شود.



چون q_0 حالت پایانی نیست پس رشته ی w_2 توسط ماشین پذیرفته نمی شود .

پیکربندی

هر پیکربندی وضعیت جاری یک DFA را توصیف می کند.هر پیکربندی در ماشین های متناهی به صورت $[q_i,aw]$ می باشد که در آن q_i حالت فعلی ماشین بوده و aw دنباله رشته روی نوار است که نوار خوان به a اشاره دارد.

پیکر بندی شروع : پیکر بندی $[q_0,w]$ را پیکر بندی شروع می نامیم اگر q_0 حالت شروع ماشین بوده و wرشته ی ورودی اولیه باشد.

پیکر بندی پذیرش : پیکر بندی $[q_f,w]$ را پیکر بندی پذیرش می نامیم اگر q_f حالت پایانی ماشین بوده و λ به معنی اتمام رشته ی ورودی می باشد .

تابع حرکت : بر روی دو پیکربندی تعریف می شود، بدین صورت که برای هر $a \in \Sigma$, $w \in \Sigma^*$ و هر تابع انتقالی مانند $\delta(q_i,a)=q_j$ این تابع به شکل زیر عمل می کند :

$[q_i \quad aw] \vdash [q_j \; w]$

به دنباله ی حرکت روی پیکربندی ها شرح آنی یا توصیف لحظه ای می گویند .

برای نشان دادن پذیرش یک رشته توسط یک ماشین از شرح آنی که با پیکر بندی شروع آغاز شود استفاده می کنیم ، بدیهی است زمانی یک رشته توسط یک ماشین متناهی پذیرفته خواهد شد که شرح آنی آن به پیکربندی پذیرش ختم شود .

مثال) با توجه به ماشین متناهی مثال قبل پذیرش دو رشته ی w_1 , w_2 را با استفاده از شرح آنی بررسی کنید.

$M=(Q,\Sigma,q,\delta,F) \qquad Q=\{q_0,q_1,q_2\} \qquad \Sigma=\{a,b\} \qquad q=q_0 \qquad F=\{q_2\}$

$\delta(q_0,a) = q_0 \quad \delta(q_0,a) = \delta(q_1,a) = \delta(q_1,a) = \delta(q_2,a) = \delta(q_2,a) = q_1$

$q_1 \qquad q_2 \qquad q_2 \qquad q_0$

$w_1=babb$

$[q_0 , babb] \vdash [q_1 , abb] \vdash [q_2 , bb] \vdash [q_1 , b] \vdash [q_2 , \lambda]$ رشته پذیرفته می شود.

$W_۲=abaa$

$[q_0 , abaa] \vdash [q_0 , baa] \vdash [q_1 , aa] \vdash [q_2 , a] \vdash [q_0 , \lambda]$ رشته پذیرفته نمی شود.

زبان ماشین

اگر $M=(Q,\Sigma,q,\delta,F)$ یک ماشین متناهی باشد مجموعه ی تمامی رشته های روی Σ که توسط M پذیرفته می شود را زبان ماشین می گویند و آن را با $L(M)$ نمایش می دهند .به عبارت دیگر :

$$L(M) = \{w \in \Sigma^* \mid \delta^*(q0,w) \in F\}$$

متمم زبان ماشین مجموعه ی رشته هایی است که توسط ماشین پذیرفته نمی شود و به صورت زیر نمایش داده می شود :

$$\overline{L(M)} = \{w \in \Sigma^* \mid \delta^*(q0,w) \notin F\}$$

جدول توابع انتقال : توابع انتقال یک ماشین را می توان در جدولی به نام جدول توابع انتقال نشان داد که حالت های ماشین بر روی سطرهای این جدول و الفبای ماشین بر روی ستون های آن قرار می گیرند.عنصر روی هر سطر و ستون نشان می دهد که هر حالت از ماشین با هر سمبل الفبا به چه حالتی منتقل شده است .

مثال) توابع انتقال ماشین متناهی مطرح شده در مثال قبل را با جدول توابع انتقال نمایش دهید

.

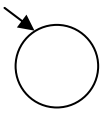
δ	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_0	q_1

نمودار وضعیت DFA

نمودار وضعیت یک DFA گراف بر چسب دار جهت داری است که با توجه به قوانین زیر ساخته می شود :

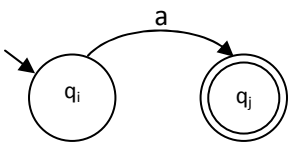
۱) راس های گراف اعضای مجموعه ی Q هستند .

۲) برچسب روی یال های گراف اعضای مجموعه ی Σ یا همان الفبای ماشین هستند .

۳) حالت شروع ماشین در راسی از گراف به صورت  نمایش می دهیم .

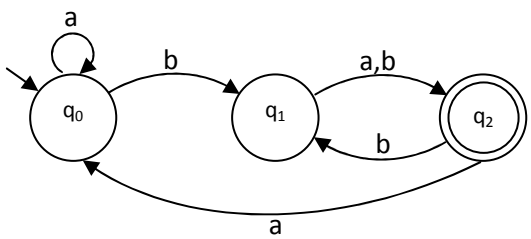
۴) حالت های پایانی ماشین در راس هایی به صورت  نشان داده می شود

۵) هر تابع انتقال $\delta(q_i,a)=q_j$ به صورت زیر نمایش داده می شود :



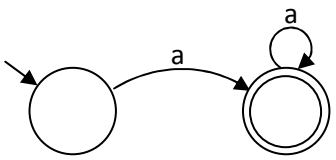
۶) برای هر حالت $q_i \in Q$ و هر الفبای $a \in \Sigma$ تنها یک یال با بر چسب a از q_i خارج می شود .

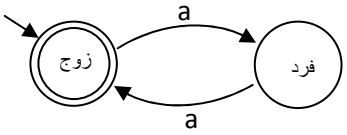
مثال) نمودار وضعیت DFA مثال قبل را رسم کنید .



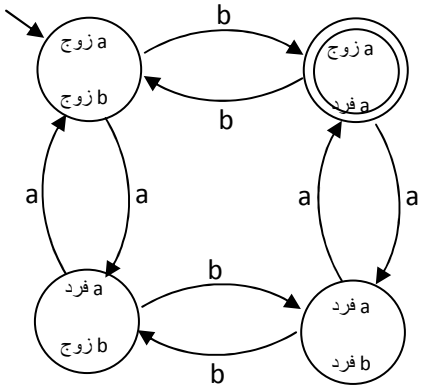
مثال) برای هر یک از زبان های زیر DFA طراحی کنید

$L_1=\{a_n \mid n\geq 1\}$ $L_2=\{a_n \mid n=2k\}$

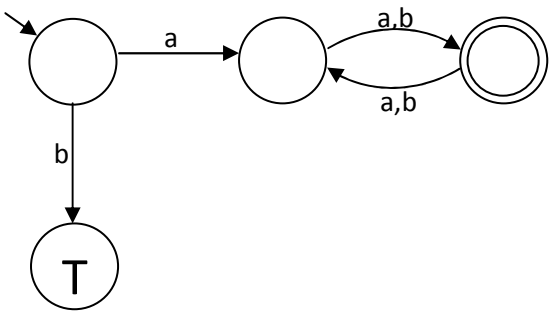




$L_3=\{w \in \{a,b\}^* \mid n_a(w)=2k\ ,n_b(w)=2k'+1\}$

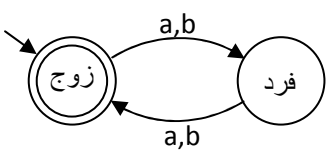


$L_4=\{w \in \{a,b\}^* \mid$ {رشته ی w با a شروع شود و طول آن زوج باشد

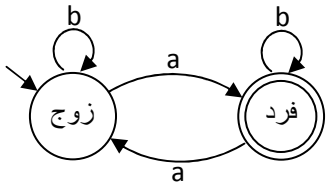


حالت Trap یا دام حالتی است غیرپایانی که باتمامی سمبل های الفبا به خودش انتقال دارد . یعنی هرگاه در مسیر بررسی پذیرش یک رشته به این حالت منتقل شویم دیگر آن رشته پذیرفته نخواهد شد.از این حالت زمانی استفاده می کنیم که بعضی ازحالت های دیگر ماشین با بعضی از سمبل های الفبا نتواند یک انتقال واقعی داشته باشند.در این صورت این انتقال را به حالت Trap رسم می کنیم تا شکل ظاهری ماشین یک DFA باشد .

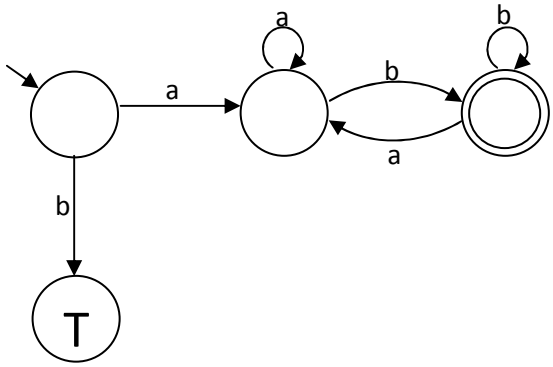
$L_5=\{w \in \{a,b\}^* \mid |w| \bmod 2 = 0\}$



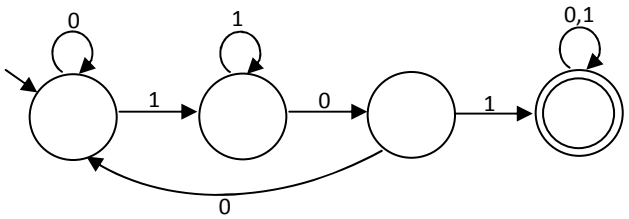
$L_6=\{w \in \{a,b\}^* \mid n_a(w) = 2k+1\}$



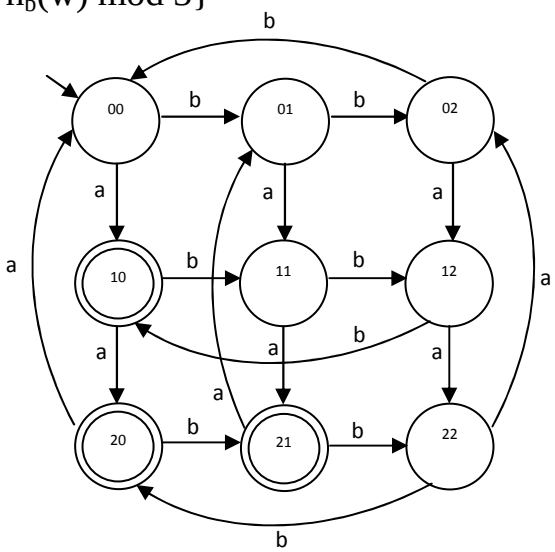
$L_7=\{w \in \{a,b\}^* \mid$ {رشته ی w با a شروع شود و به b ختم شود



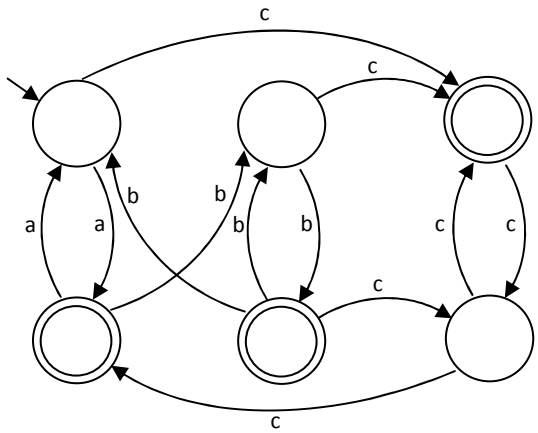
$L_8=\{w \in \{0,1\}^* \mid$ {رشته ی w دارای حداقل یک زیر رشته ی ۱۰۱باشد



$L_9=\{w \in \{a,b\}^* \mid n_a(w) \bmod 3 > n_b(w) \bmod 3\}$



$L_{10}=\{a^n b^m c^k \mid (n+m+k)$ فرد باشد $\}$



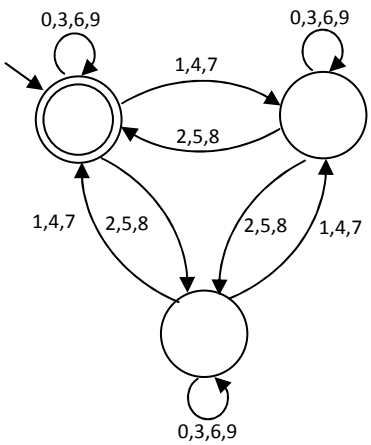
اجتماع و اشتراک ماشین های DFA

برای بدست آوردن اجتماع یا اشتراک دو ماشین DFA که اغلب الفبای آنها یکسان است، ابتدایک حالت جدید به عنوان حالت شروع ماشین حاصل رسم می کنید و در هر دو حالت DFA ماشین اولیه را در این جدید می نویسیم . سپس بررسی می کنیم که این حالت جدید با هر یک از سمبل های الفبا به چه حالتی انتقال دارد.این روند را تا زمانی که هیچ حالت جدیدی به ماشین اضافه نشود ادامه می دهیم .

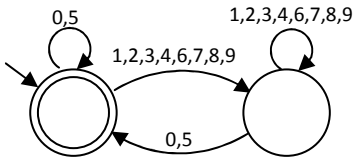
در انتها اگر هدف بدست آوردن اجتماع دو ماشین است، حالت هایی در ماشین حاصل پایانی خواهد بود که حداقل یکی از حالات پایانی دو ماشین اولیه را در خود داشته باشد.اما اگر هدف بدست آوردن اشتراک دو ماشین باشد حالت هایی در ماشین حاصل پایانی خواهد بود که تمامی حالت های موجود در آن در دو ماشین اولیه نیز پایانی باشد.

مثال) برای زبان های زیر DFA مناسب طراحی کنید و اشتراک این ماشین ها را نیز بدست آورید .

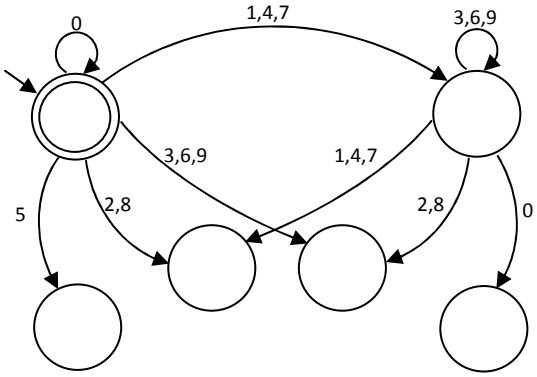
$L_1=\{w \in \{0,1,\ldots,9\}^* \mid$ {رشته ی w بر ۳ بخش پذیر باشد



$L_۲=\{w \in \{0,1,\ldots,9\}^* \mid$ {رشته ی w بر ۵ بخش پذیر باشد



$L=\{w \in \{0,1,\ldots,9\}^* \mid$ {رشته ی w بر ۱۵ بخش پذیر باشد

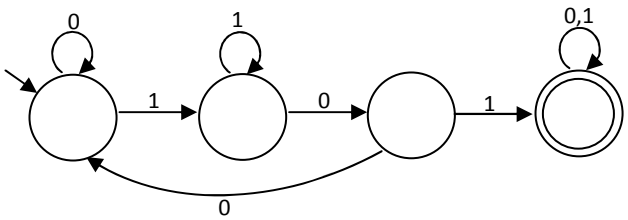


متمم DFA

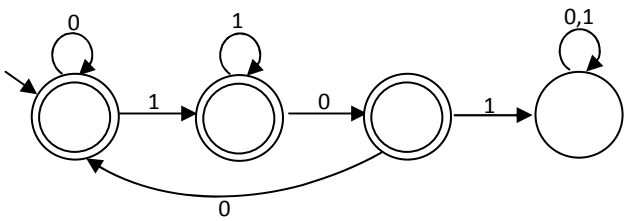
برای بدست آوردن یک DFA تنها کافی است حالت پایانی به غیر پایانی و بالعکس تبدیل شوند .

مثال)

$L=\{w \in \{0,1\}^* \mid$ {رشته ی w دارای هیچ زیر رشته ای از ۱۰۱ نباشد



$\overline{L}=\{w \in \{0,1\}^* \mid$ {رشته ی w دارای حداقل یک زیر رشته از ۱۰۱ باشد

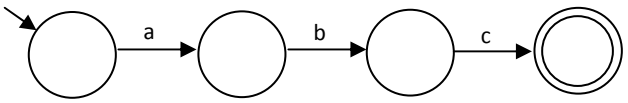


معکوس یک DFA

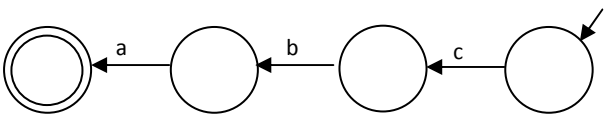
برای بدست آوردن معکوس یک DFA علاوه بر اینکه جهت تمامی انتقال ها را معکوس می کنیم باید حالت شروع را به پایان تبدیل شود.(دیگر شروع نخواهد بود).و همه ی حالت های پایانی به شروع تبدیل شوند .(دیگر پایانی نخواهند بود .)

مثال)

$L=\{abc\}\ \ \Sigma=\{a,b,c\}$



معکوس L : $L=\{cba\}$



ماشین متناهی غیر قطعی (NFA)

یک ماشین متناهی قطعی ممکن است با داشتن بعضی از شرایط عدم قطعیت به یک ماشین متناهی غیر قطعی تبدیل شود.شرایط عدم قطعیت که در یک ماشین متناهی می تواند رخ دهد شرایطی است که با بروز آنها در بعضی از موارد ماشین بیش از یک تصمیم گیری خواهد داشت .این شرایط عدم قطعیت عبارت اند از :

۱. وجود بیش از یک حالت شروع در ماشین .
۲. وجود بیش از یک انتقال با یک سمبل الفبا برای بعضی از حالت های ماشین .
۳. عدم انتقال بعضی از حالت ها با بعضی از سمبل های الفبا .
۴. وجود انتقال λ در ماشین .

یک NFA می تواند یک یا چند شرط عدم قطعیت فوق را داشته باشد .

تعریف استاندارد NFA :

هر پنج تایی به فرم $M=(Q,\Sigma,\delta,q_0,F)$ می باشد که در آن Q , Σ , q_0 , F همان مجموعه های تعریف شده در DFA هستند و δ مجموعه توابع انتقال است که هر تابع انتقال به صورت زیر توصیف می شود :

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

زبان یک NFA : اگر $M=(Q,\Sigma,\delta,q_0,F)$ نشان دهنده یک NFA باشد به مجموعه ی رشته های قابل پذیرش توسط آن، زبان NFA می گویند . به عبارت دیگر :

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

تبدیل NFA به DFA

برای هر ماشین NFA یک DFA معادل وجود دارد .منظور از یک DFA معادل، DFA ای است که زبان آن دقیقا با زبان NFA برابر یا معادل باشد . برای اثبات این ادعا کافی است روش تبدیل NFA به DFA مطرح شود .دوروش برای تبدیل NFA به DFA وجود دارد که روش اول معمولا در NFA هایی است که انتقال λ ندارند و از روش دوم بیشتر برای تبدیل NFA هایی که انتقال λ دارند استفاده می شود .

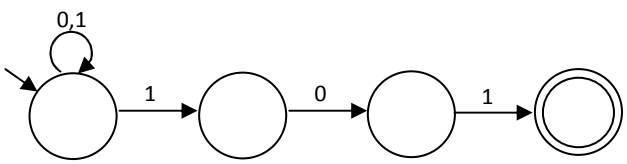
- روش اول تبدیل NFA به DFA :

در این روش ابتدا یک حالت به عنوان حالت شروع ماشین جدید (ماشین DFA) رسم می کنیم و تمامی حالت های شروع ماشین اولیه (NFA) را در آن می نویسیم.سپس مانند آنچه در بحث اجتماع و اشتراک گفته شد انتقال این حالت را با هر یک از سمبل های الفبا بررسی می کنیم.اگر چناچه حالتی با یک سمبل الفبا نتواند به هیچ حالت دیگری منتقل شود این انتقال را در ماشین جدید به حالت Trap قرار می دهیم.روند فوق را تا زمانی که هیچ حالت دیگری به ماشین اضافه نشود ادامه می دهیم .در پایان هر حالتی از ماشین DFA که حداقل یک حالت پایانی از NFA را در خود دارد در ماشین DFA پایانی خواهد بود .

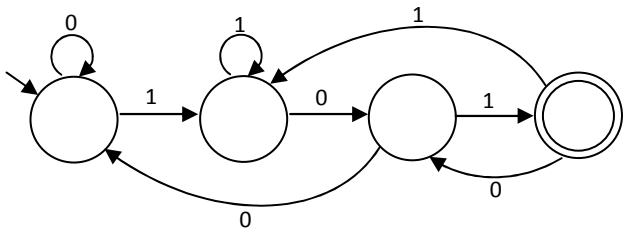
مثال) برای زبان زیر یک DFA مناسب طراحی کنید .

$$L=\{w \in \{0,1\}^* \mid \text{رشته ی } w \text{ به زیر رشته ی ۱۰۱ ختم شود}\}$$

NFA :



DFA معادل :



- روش دوم تبدیل NFA به DFA :

معمولا از این روش دوم در تبدیل NFA هایی که انتقال λ دارند استفاده می شود.برای استفاده از این روش لازم است دو تابع را به صورت زیر تعریف کنیم :

Closure (X) : اگر X یک حالت از ماشین باشد، Closure (X) مجموعه ای از حالت های ماشین خواهد بود که از حالت X و بدون دیدن هیچ سمبلی از الفبا به آنها منتقل می شویم Closure (X) را با نماد Λ (X) نمایش می دهند .

دنباله ای از انتقالات λ یک مسیر بدون سمبل الفباست .
اگر X و Y دو حالت از ماشین باشند آنگاه Closure (X,Y) = Closure (X) ∪ Closure (Y) .

Move(X,a) : اگر X یک حالت از ماشین و a یک سمبل الفبا باشد Move(X,a) باشد آنگاه Move(X,a) برابر است با :

$$Move(X,Y,a) = Move(X,a) \cup Move(Y,a)$$

برای تبدیل یک NFA به DFA توابع انتقال ماشین DFA را در جدولی به نام D_{transe} بدست خواهیم آورد.جدول D_{transe} جدولی است که سمبل های الفبای ماشین بر روی ستون های آن قرار دارند و پس از تکمیل شدن حالت های ماشین بر روی سطرهای آن خواهند بود.این جدول با الگوی زیر تکمیل می شود :

۱. اگر q_0 حالت شروع NFA باشد، Closure (q_0) به عنوان اولین را به اولین سطر جدول اضافه می کنیم.این حالت را با نامی مانند A نام گذاری می کنیم که حالت شروع DFA خواهد بود.
۲. یک حالت علامت نخورده از جدول مانند T را انتخاب می کنیم . (اگر چنین حالتی وجود ندارد به مرحله ی ۶ می رویم.)
۳. حالت T را علامت می زنیم.
۴. برای حالت T و هر سمبل الفبا مانند $a \in \Sigma$ مراحل زیر را انجام می دهیم :

الف) $U = \text{Closure}(\text{Move}(T,a))$ (U نام یک مجموعه است.)

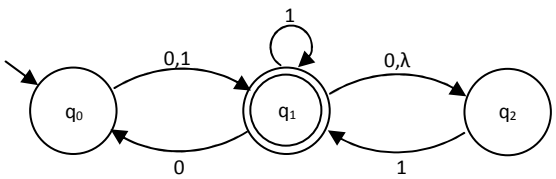
ب) اگر V یک حالت جدید است با نام گذاری آن را به سطر جدیدی از جدول اضافه می کنیم.

ج) $D_{transe}[T,a] = V$

۵. برو به مرحله ی ۲ .

۶. در پایان هر حالتی از DFA که حداقل یک حالت پایانی NFA را در خود دارد در DFA نیز پایانی خواهد بود.

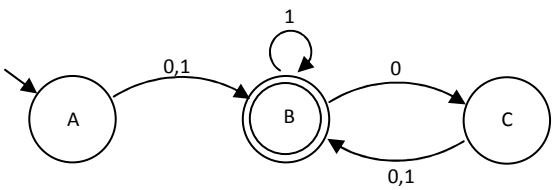
مثال) NFA زیر را به DFA تبدیل کنید.



$\text{Closure}(q_0) = \{q_0\}$ $\text{Closure}(q_1) = \{q_1 q_2\}$ $\text{Closure}(q_2) = \{q_2\}$

$\text{Closure}(\text{Move}(A,0)) = \text{Closure}(q_1) = \{q_1 q_2\}$
$\text{Closure}(\text{Move}(A,1)) = \text{Closure}(q_1) = \{q_1 q_2\}$
$\text{Closure}(\text{Move}(B,0)) = \text{Closure}(q_0 q_2) = \{q_0 q_2\}$
$\text{Closure}(\text{Move}(B,1)) = \text{Closure}(q_1) = \{q_1 q_2\}$
$\text{Closure}(\text{Move}(C,0)) = \text{Closure}(q_1) = \{q_0 q_2\}$
$\text{Closure}(\text{Move}(C,1)) = \text{Closure}(q_1) = \{q_1 q_2\}$

D_{transe}	0	1
A	B	B
B	C	B
C	B	B



بهینه ساختن ماشین های DFA

منظور از بهینه ساختن ماشین های DFA کاهش تعداد حالت های آن به حداقل حالت ممکن است به گونه ای که در ماشین کاهش یافته هیچ تغییری در زبان ماشین ایجاد نشود.برای بهینه سازی ماشین ها از یک روند به نام Partitioning استفاده می شود.این روند شامل دو مرحله است :

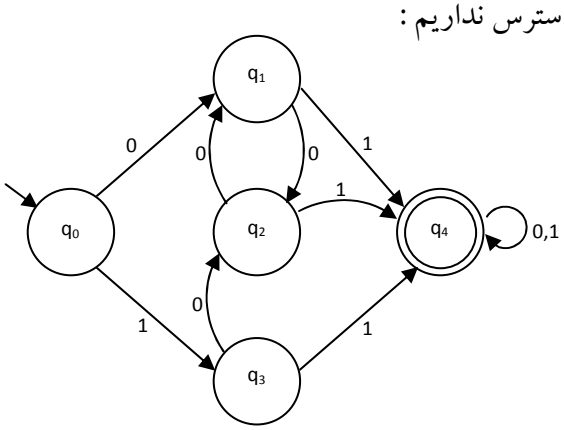
۱. در ابتدا حالتهای غیرقابل دسترس ماشین را یافته و حذف می کنیم .(در مرحله ی دوم آنها را به حساب نمی آوریم .)

یک حالت غیر قابل دسترس حالتی است که به هیچ عنوان نتوان از حالت شروع به آن رسید .

۲. در این مرحله یک مجموعه شامل دو پارتیشن تعریف می شود، یکی پارتیشن حالت های غیرپایانی و دیگری پارتیشن حالتهای پایانی .سپس رفتار حالتهای درون یک پارتیشن را به ازای سمبل های الفبایی یکسان بررسی می کنیم .منظور از رفتار یکسان دوحالت به ازای یک سمبل الفبا انتقال آن دو حالت با آن سمبل به یک پارتیشن یکسان است و منظور از رفتار متفاوت انتقال به پارتیشن های متفاوت است.پس از اینکه رفتارتک تک حالت ها با هریک از سمبل های الفبا بررسی شد حالت هایی از یک پارتیشن را که رفتار یکسانی دارند در مرحله ی بعد نیز در یک پارتیشن نگه می داریم و آنهایی که رفتار متفاوتی دارند جدا می کنیم.این روند را تا جایی ادامه می دهیم که نتیجه ی یک مرحله با مرحله ی قبلی یکسان شود و یا همه ی پارتیشن ها تک عنصری شوند.در پایان حالتهایی که درون یک پارتیشن باشند حالتهای معادل یا تمایزناپذیر می نامیم.در ماشین بهینه می توانیم به جای چند حالت معادل آن که یک حالت است قرار دهیم .ماشین بهینه از روی ماشین اولیه و با کاهش حالت های معادل بدست می آید .

مثال) تعداد حالت های آتوماتای زیر را کمینه نموده و زبان آن را بدست آورید .

حالت غیر قابل دسترس نداریم :



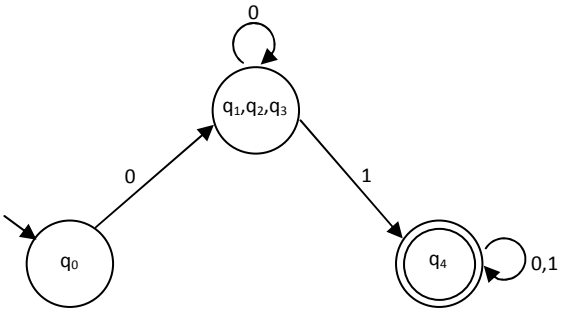
$P=\{ \underbrace{\{q_0,q_1,q_2,q_3\}}_{\text{پارتیشن۱}},\underbrace{\{q_4\}}_{\text{پارتیشن۲}} \}$

$P=\{ \underbrace{\{q_0\}}_{\text{پارتیشن۱}},\underbrace{\{q_1,q_2,q_3\}}_{\text{پارتیشن۲}},\underbrace{\{q_4\}}_{\text{پارتیشن۳}} \}$

$P=\{ \{q_0\},\{q_1,q_2,q_3\},\{q_4\} \}$

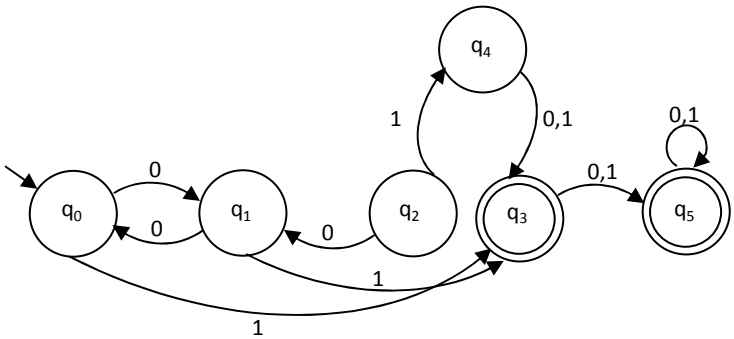
حالتهای q_1,q_2,q_3 با هم معادل اند و تمایزناپذیر می باشند.

وماشین نهایی برابر خواهد بود با :



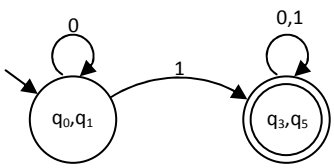
$L=\{w \in \{0,1\}^* \mid$ {w دارای حداقل یک سمبل ۱ بوده و طول آن بزرگتر یا مساوی ۲ می باشد

مثال) تعداد حالت های DFA زیر را به حداقل برسانید .



حالت های q_4 , q_2 غیرقابل دسترس هستند. پس خواهیم داشت :

$P=\{ \{q_0,q_1\} , \{q_3,q_5\}\} \rightarrow P=\{ \{q_0,q_1\} , \{q_3,q_5\}\}$



$L=\{w \in \{0,1\}^* \mid$ {w دارای حداقل یک سمبل ۱ باشد

ارتباط بین عبارت های منظم، گرامرهای منظم و ماشین های DFA

یکی از دسته زبانهایی که در نظریه مطرح می شود زبان های منظم هستند.یک زبان منظم خواهد بود اگر وتنها اگر برای پذیرش رشته های آن بتوان یک DFA و یا NFA طراحی کرد .(توانایی DFA ها و NFA ها با هم برابر است زیرا هر NFA قابل تبدیل به یک DFA است .)

قضیه:

طبق یک تعریف دیگر می توان گفت یک زبان منظم خواهد بود اگر وتنها اگر بتوانیم برای تولید رشته های آن یک عبارت منظم یا یک گرامر منظم و یا DFA طراحی کنیم .

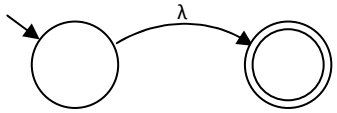
برای اثبات این ادعا تنها کافی است روش تبدیل عبارت های منظم یا گرامر های منظم به DFA وبالعکس مطرح شود.این روش ها به صورت زیر خواهند بود :

روش تبدیل عبارت های منظم به DFA

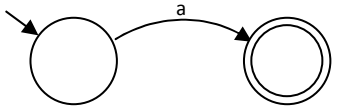
برای تبدیل هر عبارت منظم به یک DFA معادل آن از روشی به نام روش غیر مستقیم استفاده می شود.در این روش ابتدا عبارت منظم به یک NFA تبدیل می شود، سپس NFA به DFA تبدیل خواهد شد.برای تبدیل عبارت منظم به NFA باید روشی برای ارائه NFA برای عبارت های منظم پایه ونیز برای عملگر های عبارت های منظم داشته باشیم .طبق تعریف عبارت های منظم؛ ϕ، λ وهر $a \in \Sigma$ عبارت های منظم هستند.برای این سه عبارت منظم می توان ماشین های زیر را در نظرگرفت :

ϕ :

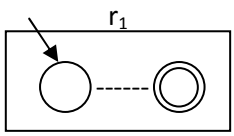
λ:

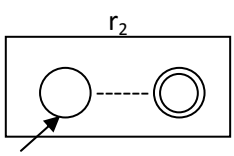


$a \in \Sigma$:

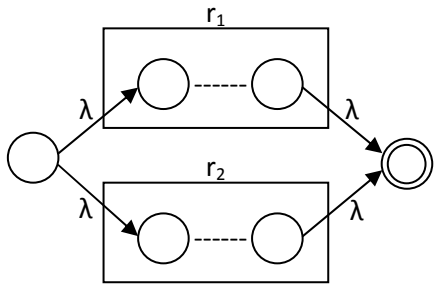


برای هر عملگر در عبارت منظم می توان یک روش در DFA ها مطرح کرد.طبق تعریف عبارت های منظم اگر r_1 و r_2 دو عبارت منظم باشند، آنگاه r_1+r_2 ، r_1r_2 و r^*_1 نیزعبارت های منظم خواهند بود .

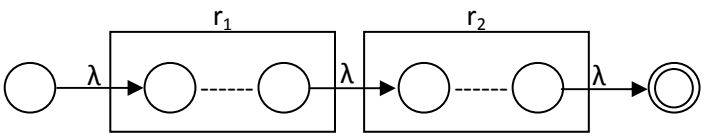




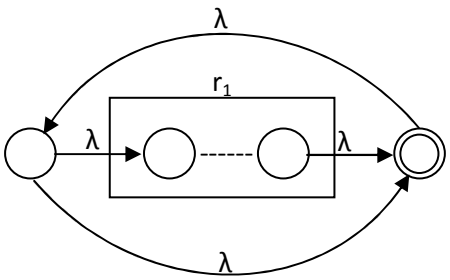
r_1+r_2



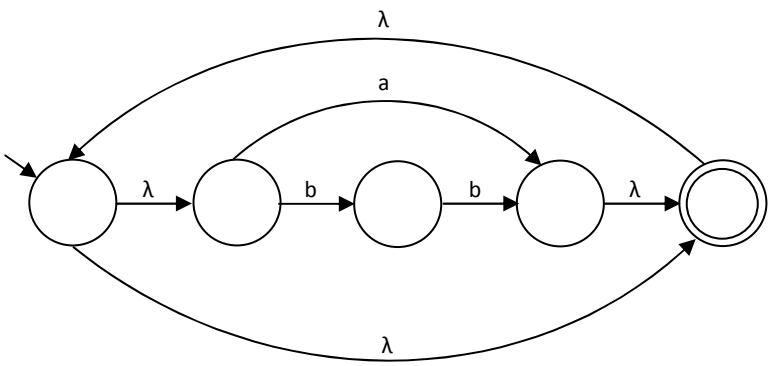
r_1r_2



r_1^*



مثال) برای عبارت منظم $(a+bb)^*$ یک ماشین DFA طراحی کنید .



ماشین فوق را می توان با روش های تبدیل NFA به DFA به یک ماشین DFA تبدیل کرد .

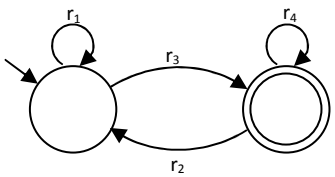
روش تبدیل ماشین های متناهی به عبارت منظم

برای بدست آوردن عبارت منظم یک ماشین متناهی (NFA یا DFA) می توان از یکی از دو روش زیر استفاده کرد :

۱. استفاده از گراف انتقال عمومی (GTG) :

یک گراف انتقال عمومی دقیقا مانند یک ماشین متناهی خواهند بود با این تفاوت که بر روی یال های آن می تواند عبارت منظم قرارگیرد.یک گراف انتقال عمومی کامل نوعی گراف است که در آن از هر راس به راس دیگر (حتی خود آن راس) یک یال مستقیم وجود دارد.برای تبدیل یک DFA به عبارت منظم ابتدا ماشین را به یک گراف انتقال عمومی کامل تبدیل می کنیم.در این تبدیل تنها کافی است از هر راس به راسی دیگر که یال مستقیم نداریم، یک یال با برچسب تهی اضافه کنیم.سپس گراف انتقال کامل را تا رسیدن به یک گراف انتقال عمومی دو حالته (یک حالت شرع و یک حالت پایان) کاهش حالت می دهیم.روند کاهش حالت باید به گونه ای باشد که در هر مرحله یک حالت از این گراف کاسته شود.نکته ی دیگر اینکه این روند باید به گونه ای باشد که در انتها گراف عمومی دو حالته حتما دارای یک حالت شروع و یک حالت پایانی باشد.هرگاه می خواهیم یک حالت را کاهش دهیم باید تمامی انتقالها بین حالت های دیگر را با عبور از این حالت کاسته شده نیز در نظر بگیریم و عبارت منظم لازم را اضافه کنید.در پایان اگر به گراف انتقال عمومی دو حالته مانند زیر رسیدیم عبارت منظم آن به شکلی که گفته می شود بدست می آید .

$$r^*_1 r_2 (r_4 + r_3 r^*_1 r_2)^*$$



برای ساده سازی عبارت های منظم باید به روابط زیر دقت کنیم :

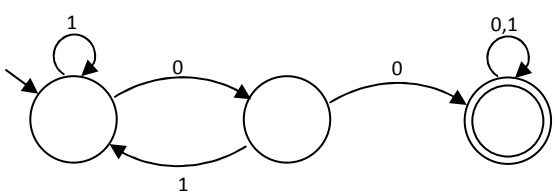
$r + \phi = r$

$r.\phi = \phi$

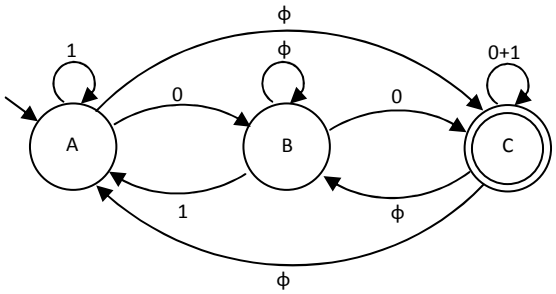
$\phi^* = \lambda$

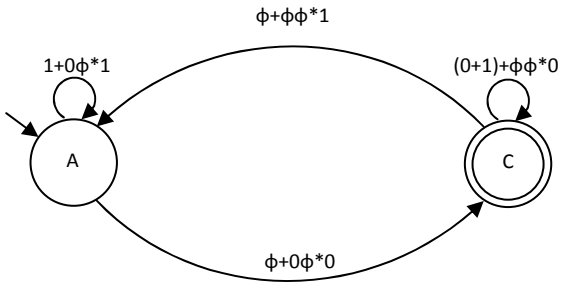
مثال) یک عبارت منظم برای زبان زیر بنویسید .

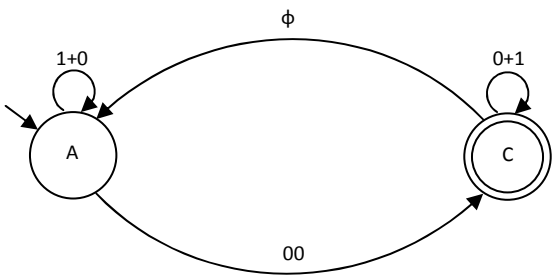
$L=\{w \in \{0,1\}^* \mid$ {رشته ی w دارای حداقل یک زوج صفر متوالی باشد



گراف کامل انتقال عمومی





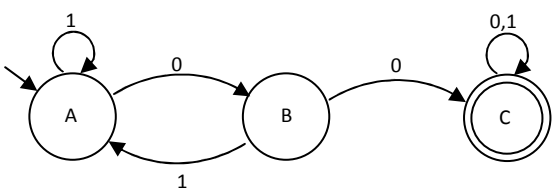


$$(1+01)^{*}00[(0+1)+\phi(1+01)^{*}00]^{*}=(1+01)^{*}00(0+1)^{*}$$

۲. استفاده از معادلات حالت :

دراین روش ابتدا معادله ی حالت هر حالت از ماشین را با توجه به انتقال های خروجی از آن حالت می نویسیم.سپس هرمعادله حالت به فرم $X=pX+q$ را به معادله ی حالت$X=p^{*}q$ تبدیل می کنیم درنهایت با جایگذاری های مختلف باید به جایی برسیم که معادله حالت متناظر با حالت شروع فاقد هر گونه نام حالتی باشد.این عبارت همان عبارت منظم ماشین است .

مثال) عبارت منظم متناظر با ماشین زیر را بدست آورید .



برای نوشتن معادلات حالت ماشین هر تابع انتقال به فرم $\delta(q_i,a)=q_j$ به معادله حالت $q_i,a=q_j$ تبدیل می شود.همچنین هر حالتی که به عنوان یک حالت پایانی مطرح شود یک عبارت λ به صورت اجتماع به معادله ی حالتش اضافه می شود .

$A=1A+0B$
$B=0C+1A$
$C=0C+1C+\lambda$ $C=0C+1C+\lambda \rightarrow C=(0+1)C+\lambda \rightarrow C=(0+1)^*\lambda \rightarrow C=(0+1)^*$
$B=0C+1A \rightarrow B=0(0+1)^*+1A \rightarrow$
$A=1A+0B \rightarrow A=1A+0[0(0+1)^*+1A] \rightarrow A=1A+01A+00(0+1)^*$
$A=(1+01)A+00(0+1)^* \rightarrow A=(1+0)^*00(1+0)^*$

روش تبدیل گرامر های منظم به ماشین های متناهی و بالعکس

هر گرامر منظم را می توان به یک ماشین متناهی و نیز هر ماشین متناهی را می توان به یک گرامر منظم تبدیل کرد.در این تبدیل لازم است گرامر منظم یک گرامر منظم خطی از راست باشد .

۱. روش تبدیل گرامر های خطی از راست به ماشین متناهی :

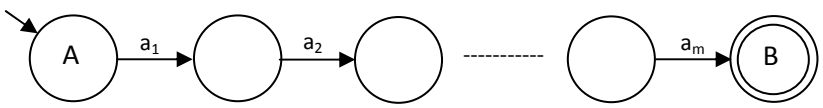
طبق تعریف گرامرهای خطی از راست خواهد بود که هر قاعده ی آن به یکی از دو شکل زیر باشد :

این دو فرم را می توان به شکل فرم زیر به ماشین تبدیل نمود :

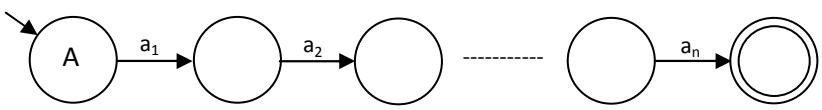
$$\begin{cases} A\rightarrow \alpha B \\ A\rightarrow \alpha \end{cases}$$

$$\begin{cases} A\rightarrow \alpha_1 a_2 \ldots a_m B & A,B \in V \\ A\rightarrow \alpha_1 a_2 \ldots a_n & a_i \in T \end{cases}$$

برای فرم اول می توان آن را به شکل زیر به ماشین تبدیل نمود :

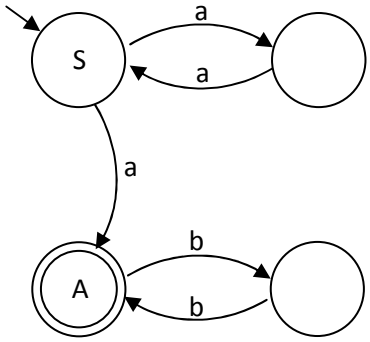


وبرای فرم دوم خواهیم داشت :



مثال) برای گرامرمنظم زیر یک ماشین متناهی بدست آورید .

$S\rightarrow aaS \mid aA$
$A\rightarrow bbA \mid \lambda$

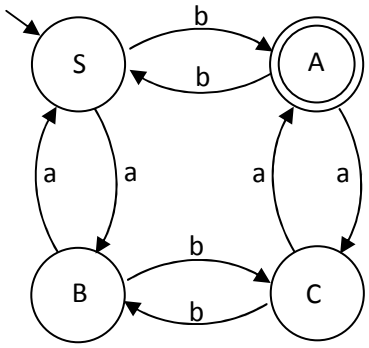


۲. روش تبدیل ماشین های متناهی به گرامر های خطی از راست :

برای بدست آوردن گرامر منظم یک ماشین متناهی تنها کافی است تمامی معادلات حالتهای ماشین را برای هر حالت به شکل گرامرها (علامت | به جای +) بنویسید.حالت شروع ماشین همان متغیر شروع گرامر خواهد بود.

مثال) یک گرامر منظم برای زبان زیر بنویسید .

$L=\{w \in \{a,b\}^* \mid n_a(w)=2k , n_b(w)=2k'+1\}$



$S \rightarrow aB \mid bA$
$A \rightarrow aC \mid bS \mid \lambda$
$B \rightarrow aS \mid bC$
$C \rightarrow aA \mid bB$

لم تزریق برای زبان های منظم

یک زبان منظم خواهد بود اگر وتنها اگر برای رشته های آن بتوان یک ماشین متناهی (NFA یاDFA) یا یک گرامر منظم و یا یک عبارت منظم ارائه داد.درمورد زبان های منظم دو نکته ی زیر وجود دارد :

۱. هر زبان متناهی منظم خواهدبود .
۲. اگر در یک زبان حداقل دوتا از سمبل های الفبا از نظر تعداد به هم وابسته باشند و این تعداد مقدار نامحدودی باشد؛آن زبان منظم نخواهد بود .

مثال)

$L=\{a^n b^m c^{2n} \mid n,m \geq 0\}$ نامنظم

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

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

$L=\{a^n b^n \mid n \leq ۲^{1000}\}$ منظم

۳. اگر در یک زبان تعداد یک سمبل الفبایی مقدار خاصی باشد (مانند فاکتوریل،عدد اول،مربع کامل،توانی از ۲ و...) به شکلی که این اعداد همگی مضرب یک عدد مشخصی نباشد یا اگر مضرب آن عدد هستند اعداد دیگری هم که این ویژگی را ندارند مضرب آن عدد باشند و این مقدار نامحدود باشد آن زبان منظم نخواهد بود .

مثال)

$L=\{a^n \mid n$ یک عدد اول است$\}$

$L=\{a^n \mid n == m!\}$

$L=\{a^n \mid n$ عددی اول کوچکتر از $۲^{۱۰۰۰}$ است$\}$

لم تزریق

برای زبان های منظم یک شرط لازم وجود دارد که هر زبانی اگر منظم باشد با این شرط لازم را داشته باشد.این شرط لازم به صورت زیر مطرح می شود :

قضیه تزریق : فرض کنید زبان L یک زبان منظم نامتناهی باشد آنگاه ثابتی مثل n وجود دارد به طوریکه برای هررشته‌ی w که $w \in L$ بوده واندازه ی $w \geq n$ باشد، می توانیم w را به سه زیر رشته به صورت $w=xyz$ تقسیم کنیم که $|xy| \leq x$ و $|y| \geq 1$ باشد، آنگاه باید به ازای هر $i \geq 0$ داشته باشیم :

$xy^iz \in L$

شرط کافی برای منظم بودن یک زبان همان ارائه ماشین متناهی یا عبارت منظم یا گرامر منظم است.معمولا از لم تزریق برای اثبات منظم بودن یک زبان استفاده نمی شود زیرا هم بیان آن برای همه ی رشته های زبان و همه ی مقادیر i غیرممکن است و هم اینکه این شرط یک شرط لازم است نه کافی .

کاربرد لم تزریق

از لم تزریق معمولا برای اثبات منظم بودن یک زبان استفاده می شود.اگر بخواهیم ثابت کنیم زبانی منظم نیست باید ابتدا رشته ای از زبان را که طول آن نامحدود است مثال بزنیم.یعنی طول رشته ی $w \geq n$ باشد.سپس رشته را به سه زیر رشته تقسیم می کنیم .به طور مثال اگر w را به سه زیر رشته ی xyz تقسیم کردیم؛ اولا باید اندازه ی $|xy| \leq n$ و ثانیا $|y| \geq 1$ باشد.در انتها کافی است یک i معرفی کنیم که $xy^iz \in L$ نباشد.

مثال) ثابت کنید که هر یک از زبان های زیر منظم نیستند .

$w = a^nb^n$

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

$w = xy$

$x = a^{n-1}$ $\quad |xy|=n \leq n$

$y = a$ $\quad |y| \geq 1$

$z = b^n$

$xy^iz = a^{n-1}a^ib^n$ $\quad i=2 \quad a^{n-1}a^2b^n = a^{n+1}b^n \notin L$.

زبان منظم نیست

$w = a^nb^{n+1}c^{2n+1}$ $\quad w \in L$, $|w|=an+2 \geq n$

$L_2=\{a^nb^mc^k \mid k=n+m\}$

$w = xyz$

$x = a^{n-1}$ $\quad |xy|=n \leq n$

$y = a$ $\quad |y| \geq 1$

$z = b^{n+1}c^{2n+1}$

$xy^iz = a^{n-1}a^ib^{n+1}c^{2n+1}$ $\quad i=4 \quad a^{n-1}a^4b^{n+1}c^{2n+1} = a^{n+3}b^{n+1}c^{2n+1} \notin L$

زبان منظم نیست.

$\Sigma=\{a,b\}$ $\quad w = a^nb^{n+2}$ $\quad w \in L$, $|w|=2n+2 \geq n$

$L_3=\{w \in \Sigma^* \mid n_a(w) < n_b(w)\}$

$w = xyz$

$x = a^{n-1}$ $\quad |xy|=n \leq n$

$y = a$ $\quad |y| \geq 1$

$z = b^{n+2}$

$xy^iz = a^{n-1}a^ib^{n+2}$ $\quad i=4 \quad a^{n-1}a^4b^{n+2} = a^{n+3}b^{n+2} \notin L$

زبان منظم نیست.

$w = a^nb^nb^na^n$ $\quad w \in L$, $|w|=4n \geq n$

$L_4=\{ww^R \mid w \in \{a,b\}^*\}$

$w = xyz$

$x = a^{n-1}$ $\quad |xy|=n \leq n$

$y = a \qquad |y| \geq 1$

$z = b^n b^n a^n$

$xy^i z = a^{n-1} a^i b^n b^n a^n \qquad i=2 \qquad a^{n+1} b^n b^n a^n = a^{n+3} b^{n+2} \notin L$

زبان منظم نیست.

$L_5 = \{a^n \mid n \geq 2 \text{ , } n \text{ عددی اول است}\}$ $\qquad w = a^n b^n b^n a^n \qquad w \in L \text{ , } |w| = n \geq n$

$w = xyz$

$x = a^{n-2} \qquad |xy| = n-1 \leq n$

$y = a \qquad |y| \geq 1$

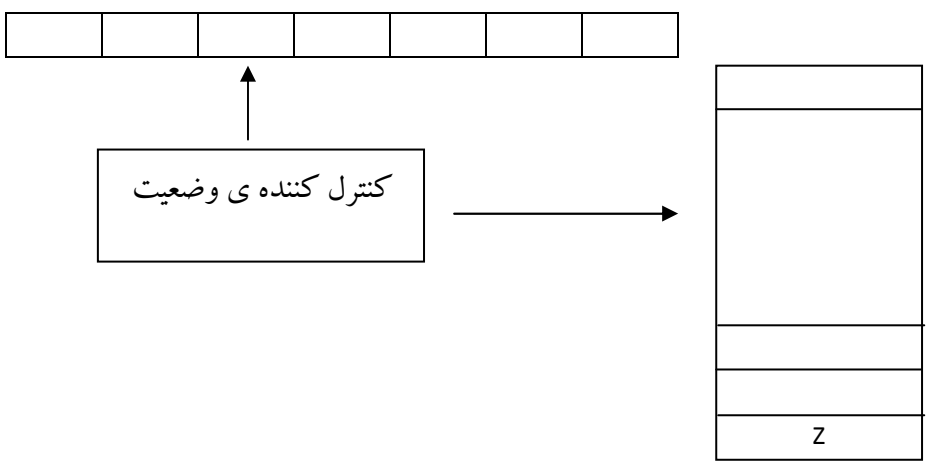
$z = a$

$xy^i z = a^{n-1} a^i a \qquad i=2 \qquad a^{n-1} a^2 a = a^{n+1} \notin L$

اگر n یک عدد اول است پس یک عدد فرد است درنتیجه یک عدد زوج خواهد بود که به هیچ عنوان نمی تواند اول باشد ، پس زبان منظم نیست .

ماشین های پشته ای (Push Down Automata)

بسیاری از زبان ها جزء دسته ی زبان های منظم نیستند و به هیچ عنوان نمی توان برای آنها ماشین متناهی طراحی کرد.مانند زبان $L=\{a^n b^n \mid n \geq 0\}$ زیرا در ماشین های متناهی هیچ ابزاری برای کنترل تعداد سمبل های a و b و موازنه ی آنها وجود ندارد. ماشین های پشته ای برای دسته ای از زبان ها به نام زبان های مستقل از متن توصیف می شوند.در مدل این ماشین علاوه بر نوار ورودی و کنترل کننده ی وضعیت از یک پشته نیز استفاده می شود.این پشته باعث می شود که بتوانیم در یک لحظه تعداد ۲ تا از سمبل های الفبا را موازنه یا کنترل کنیم.مدل یک ماشین پشته ای به شکل زیر خواهد بود :



پشته

تعریف ماشین پشته ای : هرماشین پشته ای یک هفت تایی به فرم $M=(Q,\Sigma,\Gamma,\delta,q_0,Z,F)$ که در آن :

Q :یک مجموعه ی متناهی وغیر تهی از حالت های ماشین.

Σ : یک مجموعه ی متناهی وغیر تهی از الفبای ماشین.

Γ : یک مجموعه ی متناهی وغیر تهی از سمبل هایی که به آنها الفبای پشته می گویند.

δ : مجموعه ی توابع انتقال است که هر تابع به صورت زیر تعریف می شود :

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

$q_0 \in Q$: نشان دهنده ی حالت شروع ماشین است.

$Z \in \Gamma$: نشان دهنده ی سمبل انتهای پشته است.

$F \in Q$: مجموعه ی غیرتهی حالت های پایانی ماشین است.

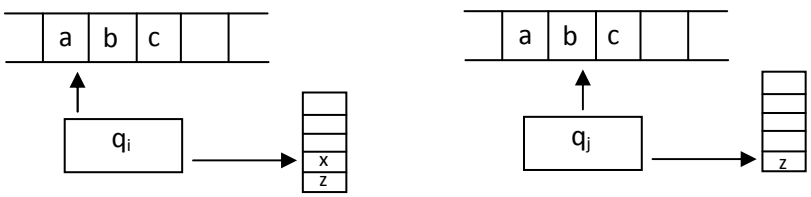
در ماشین پشته ای توابع انتقال بدین شکل توصیف می شوند که هر حالت از ماشین با توجه به سمبل جاری رشته ی ورودی وسمبل بالایی پشته به چه حالت دیگری منتقل شده و چه تغییری در پشته ایجاد می کند.در مورد اعمالی که در پشته های ماشین های PDA صورت می گیرد، می توان حالت های زیر را در نظر گرفت :

۱. اضافه کردن یک سمبل به بالای پشته با دیدن سمبل جاری رشته ی ورودی (Push) :

$\delta (q_i,a,x)=(q_j,yx)$

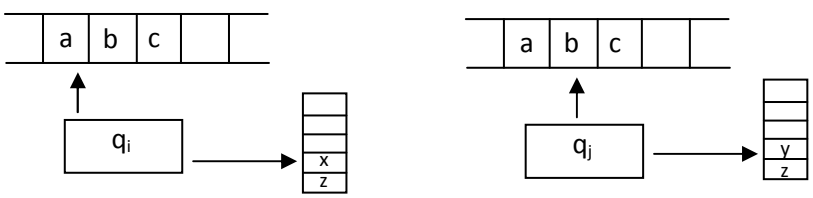
۲. برداشتن یک سمبل از بالای پشته با دیدن سمبل جاری رشته ی ورودی (Pop) :

$\delta (q_i,a,x)=(q_j,\lambda)$



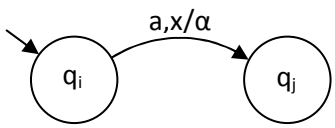
۳. جایگزین کردن یک سمبل به جای سمبل بالای پشته با دیدن سمبل جاری رشته ی ورودی :

$\delta (q_i,a,x)=(q_j,y)$



در ماشین پشته ای به ازای اجرای هر بار یک تابع انتقال هد رشته ی ورودی تنها یک واحد به سمت جلو حرکت می کند.
در ماشین های پشته ای فرض می شود که در انتهای رشته ی ورودی سمبل λ قرار دارد.
الفبای پشته می تواند با الفبای ورودی ماشین یکسان باشد و یا از آن متفاوت باشد هر چند بهتر است حتی الامکان متفاوت در نظر گرفته شود.

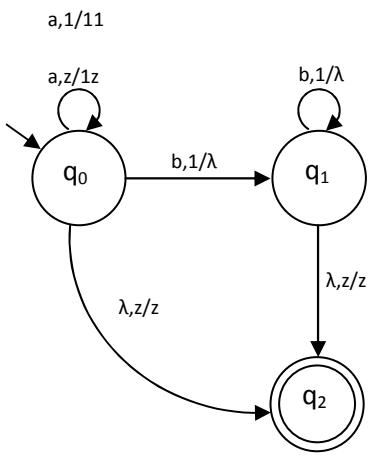
در ماشین های پشته ای هنگامی یک رشته توسط ماشین پذیرفته خواهد شد که ماشین در یکی از حالت های پایانی بوده و هد ورودی در انتهای رشته ی ورودی باشد.در این وضعیت محتوای داخل پشته اهمیتی ندارد.در موردماشین های پشته ای نیز می توان هر ماشین را با یک گراف وضعیت توصیف کرد به طور مثال تابع انتقال $\delta(q_i,a,x)=(g_j,\alpha)$ به شکل زیر نمایش داده می شود :



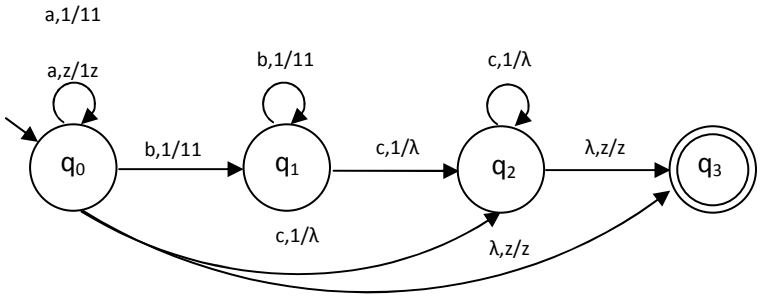
برای طراحی یک ماشین پشته ای برای یک زبان همواره فرض می کنیم در ابتدای کار ماشین در حالت شروع q_0 قراردارد و در این حالت هد پشته به سمبل z (سمبل آخر پشته) اشاره می کند.با تنظیم توابع انتقال و استفاده از پشته ماشین طراحی می کنیم که در تمامی رشته های عضو زبان را پذیرفته و هیچ رشته ای ر اکه عضو زبان نیست نپذیرد.

مثال) برای زبان زیر یک PDA طراحی کنید.

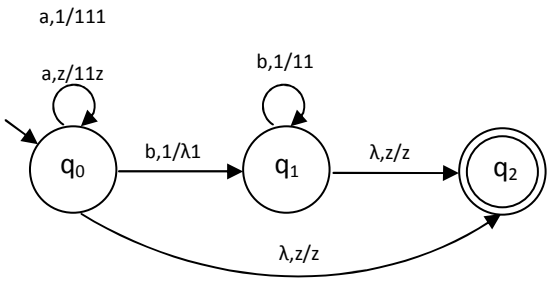
$L_1=\{a^n b^n \mid n \geq 0\}$



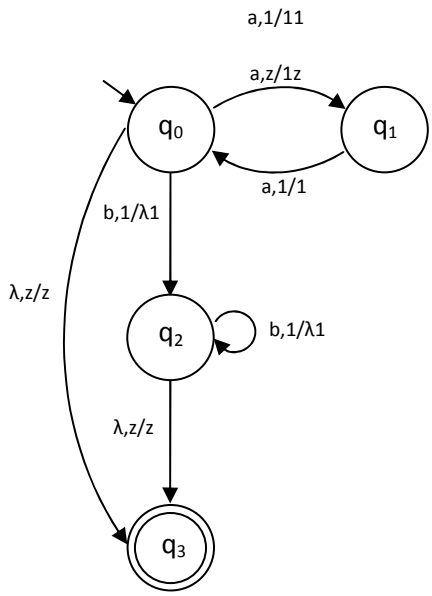
$L_2=\{a^n b^n c^k \mid k=n+m\}$



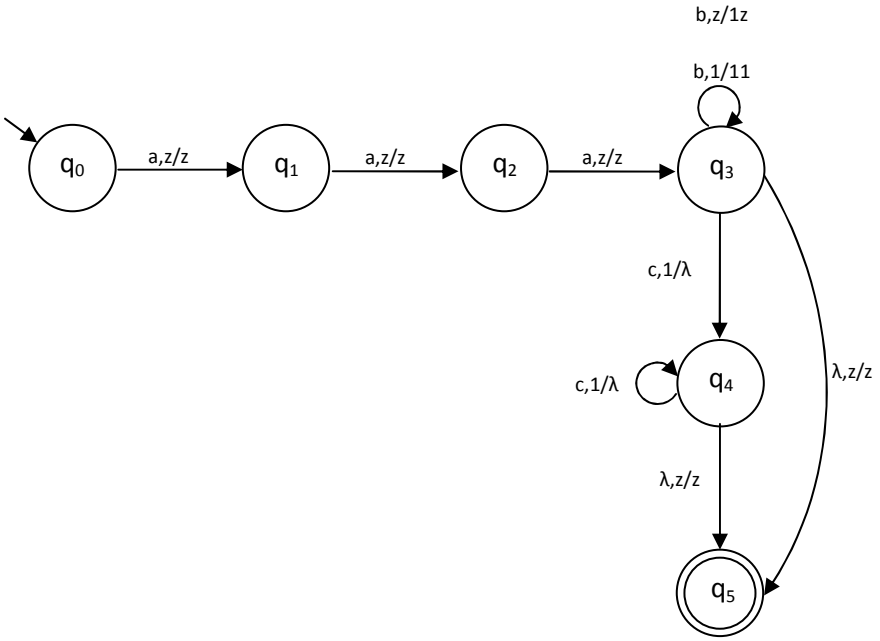
$L_3=\{a^n b^{2n} \mid n\geq 0\}$



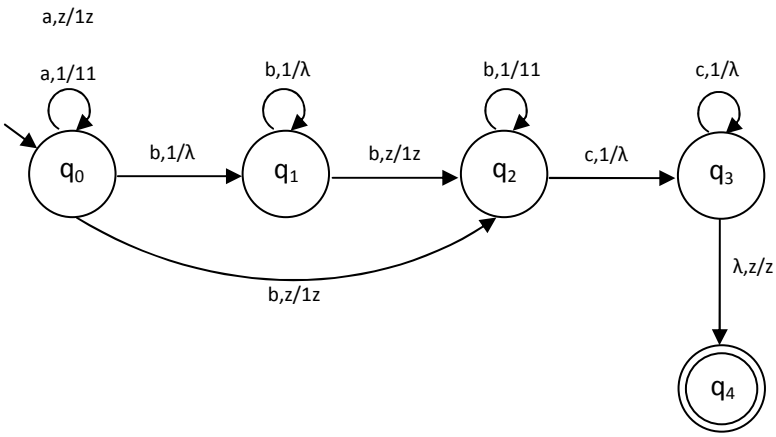
$L_4=\{a^n b^{2n} \mid n\geq 0\}$



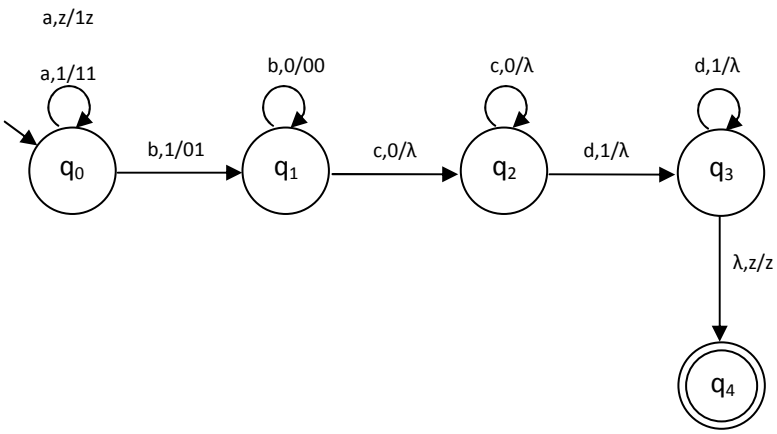
$L_5=\{a^3b^nc^n \mid n\geq 0\}$



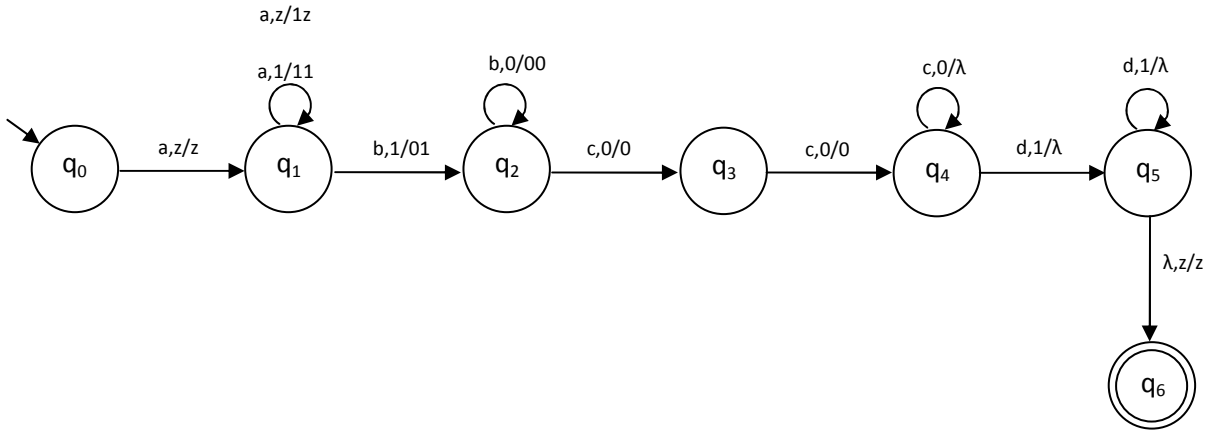
$L_6=\{a^nb^{n+m}c^m \mid n\geq 0 , m\geq 1\}$



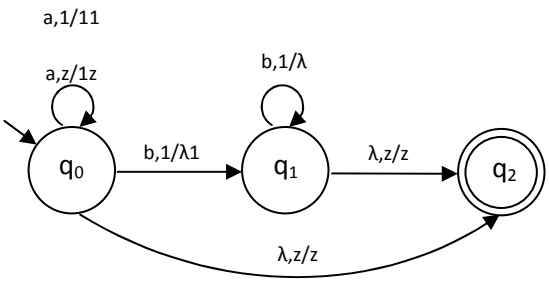
$L_7=\{a^nb^mc^md^n \mid n,m>0 \}$



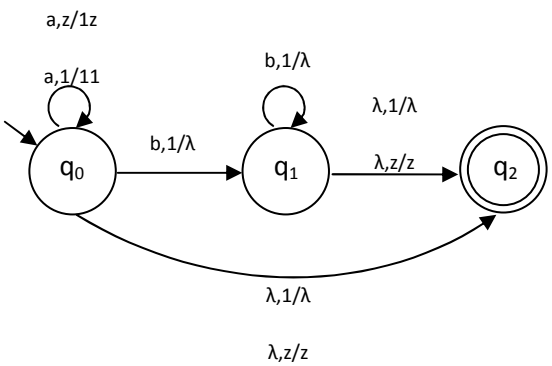
$L_8=\{a^{n+1}b^mc^{m+2}d^n \mid n,m>0\}$



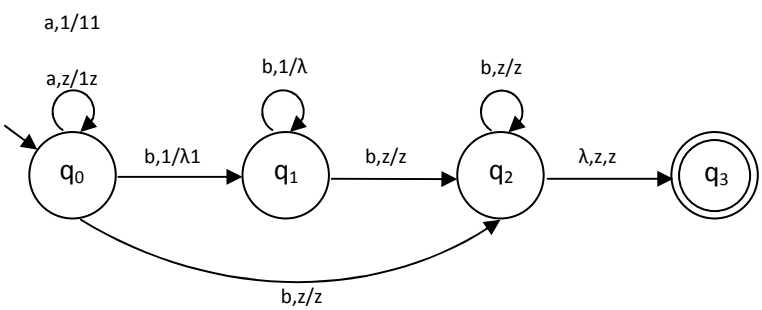
$L_9=\{a^nb^m \mid n>m\}$



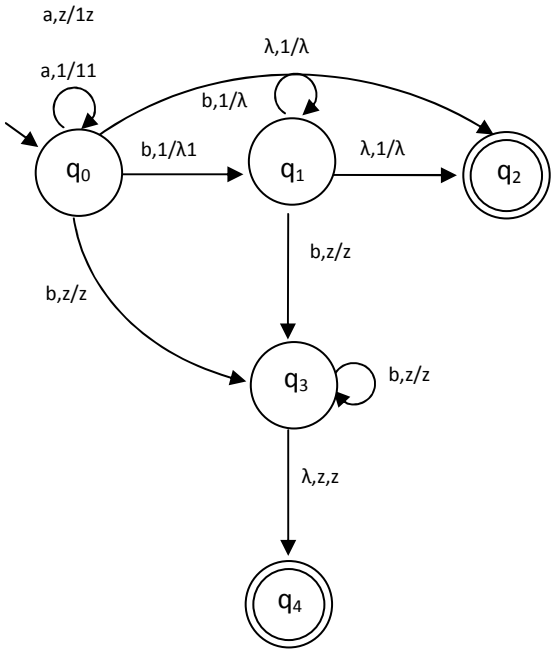
$L_{10}=\{a^nb^m \mid n\geq m\}$



$L_{11}=\{a^nb^m \mid n<m\}$



$L_{12}=\{a^n b^m \mid n \neq m\}$



ماشین های پشته ای غیر قطعی (NPDA)

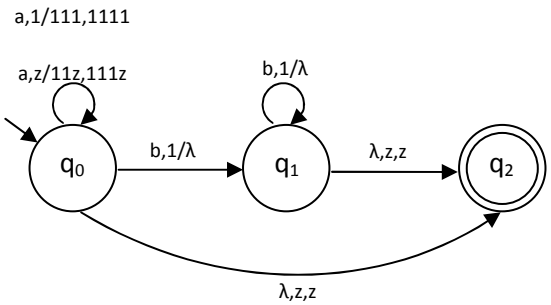
در ماشین های پشته ای نیز می توان شرایط عدم قطعیت وجود داشته باشد اما شرایط عدم قطعیت این ماشین ها با شرایط عدم قطعیت ماشین های متناهی متفاوت است.به طور مثال اگر یک حالت به ازای سمبل a بیش از یک انتقال داشته باشد یا اینکه اصلا انتقال نداشته باشد نمی توان گفت که ماشین حتما غیر قطعی است.یک ماشین پشته ای زمانی غیر قطعی خواهد بودکه در یک حالت با شرایط یکسان (سمبل ورودی و سمبل بالای پشته) بیش از یک مورد برای تصمیم گیری وجود داشته باشد و یا اینکه در ماشین انتقال λ داشته باشیم که این λ به معنی λ انتهای رشته نباشد.

کاربرد این انتقالات λ در ماشین های پشته ای مربوط به زمانی است که می خواهیم بدون توجه به سمبل جاری رشته ی ورودی و فقط با در نظر گرفتن بالای پشته تصمیم مناسب را اتخاذ کنیم که در این صورت از انتقال λ استفاده می کنیم.در این نوع انتقال هد رشته ی ورودی هیچ حرکتی نمی کند.در این صورت ماشین پشته ای غیر قطعی خواهد بود.

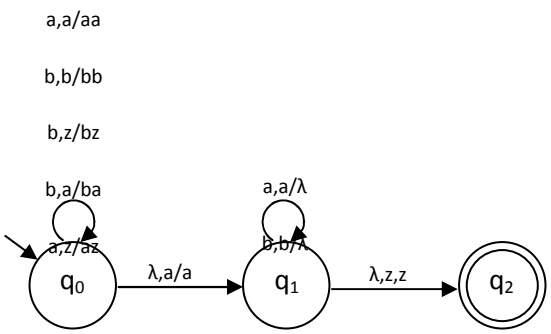
ماشین های پشته ای غیر قطعی را نمی توان به ماشین های پشته ای قطعی تبدیل نمود پس می توان گفت توانایی NPDA ها بیش تر از PDA ها می باشد .

مثال)

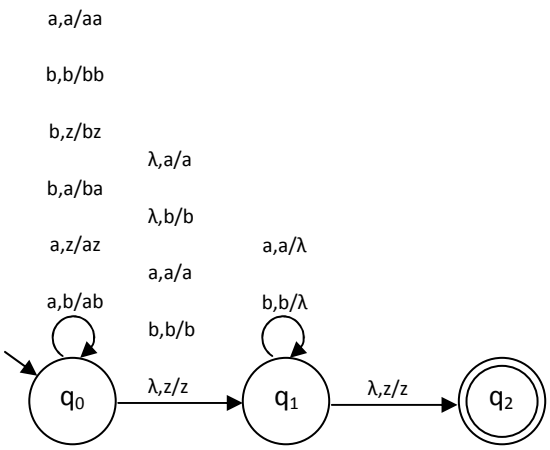
$L_1=\{a^n b^m \mid 2n \le m \le 3n\}$ n=3 m=6 ,7,8,9



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



$L_3=\{w=w^R \mid w \in \{a,b\}^*\}$



ماشین های پشته ای برای گرامر های مستقل از متن

برای هر گرامر مستقل از متن می توان یک ماشن پشته ای غیر قطعی ارائه کرد.برای این کار لازم است که حتما گرامر به فرم نرمال گریباخ باشد.ماشین پشته ای که برای یک گرامر در فرم نرمال گریباخ ارائه می شود یک ماشین ۳ حالته خواهد بود که در آن سه حالت q_0,q_1,q_2 خواهیم داشت.حالت q_0 حالت شروع تابع و حالت q_2 حالت پایانی ماشین به صورت زیر تعریف می شود :

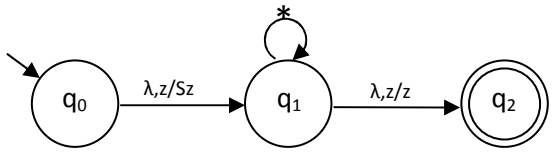
$\delta(q_0,\lambda,z)=(q_1,S_z)$

با این تابع بدون توجه به رشته ی ورودی متغیر شروع گرامر در پشته قرار می گیرد.پس برای هر قاعده تولید گرامر یک تابع انتقال روی همان حالت q_1 نوشته می شود.از آنجایی که گرامر به فرم نرمال گریباخ است، هر یک از قواعد آن به صورت زیر نوشته می شود :

$$\delta (q_1,a,A)=(q_1,\alpha)$$

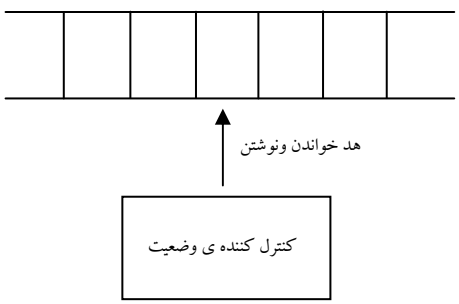
در نهایت برای پذیرش رشته های گرامر در صورتی که q_1 به انتهای رشته رسیده و پشته خالی باشد به حالت پایانی ماشین می رود یعنی آخرین تابع انتقال به صورت زیر خواهد بود :

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



ماشین تورینگ

بسیاری از زبان ها هستند که جزء دسته زبان های مستقل از متن محسوب نمی شوند، مانند زبان $L=\{a^nb^nc^n \mid n\geq 0\}$.برای این زبانها به هیچ عنوان نمی توان ماشین پشته ای استاندارد (قطعی یا غیر قطعی) ارائه داد.برای پذیرش رشته های این زبان ها از ماشین تورینگ استفاده می شود.مدل انتزاعی ماشین تورینگ دارای یک نوار ورودی نامحدود و یک کنترل کننده ی وضعیت است که این کنترل کننده ی وضعیت از طریق یک هد خواندن ونوشتن به نوار دسترسی دارد :



ماشین تورینگ دارای دو ویژگی خاص است :

۱. یکی اینکه هد خواندن علاوه بر خواندن سمبل های روی نوار می تواند روی نوار نیز بنویسد.
۲. با هر بار اجرای یک تابع انتقال هد می تواند یک واحد به سمت راست یا چپ حرکت کند.

تعریف ماشین های تورینگ :

هرماشین تورینگ یک هفت تایی به فرم $M=(Q,\Sigma,\Gamma,\delta,q_0,\square,F)$ که در آن :

Q :یک مجموعه ی متناهی وغیر تهی از حالت های ماشین.

Σ : یک مجموعه ی متناهی وغیر تهی از الفبای ماشین.

Γ : یک مجموعه ی متناهی وغیر تهی از سمبل هایی که بر روی نوار نوشته می شود.

δ : مجموعه ی توابع انتقال است که هر تابع به صورت زیر تعریف می شود :

$$\delta : Q * \Gamma \rightarrow Q * \Gamma^* \{R,L\}$$

$q_0 \in Q$: نشان دهنده ی حالت شروع ماشین است.

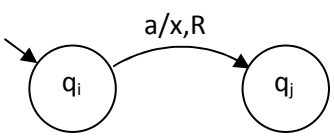
$\Gamma \ni \square$: نشان دهنده ی یک خانه ی خالی نوار است.

$F \in Q$: مجموعه ی حالت های پایانی ماشین است.

نکته: در اکثر ماشین های تورینگ فرض می کنیم : $\Sigma \subset \Gamma - \{\square\}$

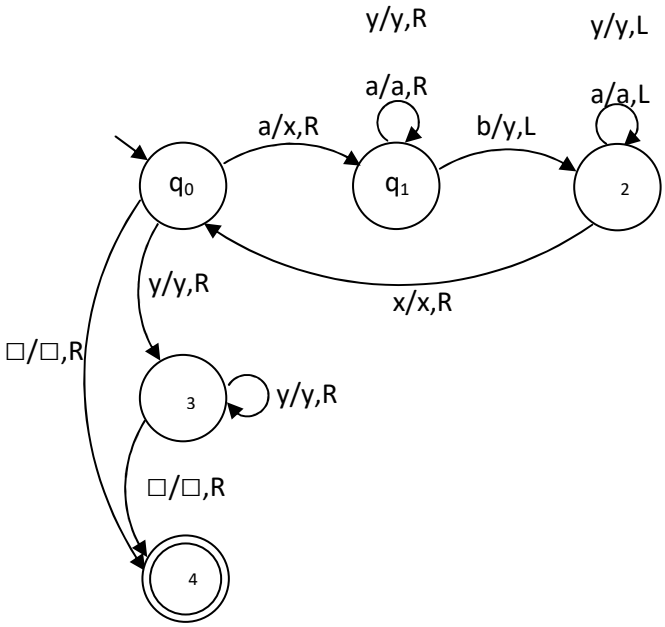
هر تابع انتقال در یک ماشین تورینگ بیان می کند که یک حالت از ماشین به ازای یک سمبل روی نوار به چه حالت دیگری منتقل شده و چه سمبلی را روی نوار جایگزین کند و در چه جهتی حرکت کند .جهت حرکت می تواند به راست(R) و یا چپ(L) باشد.

یک تابع انتقال دریک ماشین تورینگ می تواند تابعی مانند $\delta(q_i,a)=(q_j,x,R)$ که این تابع به معنی آن است که اگر ماشین درحالت q_i بوده و سمبل a از رشته ی ورودی را ببیند به حالت q_j رفته و x را جایگزین a نموده و هد به سمت حرکت کند .برای نمایش ماشین های تورینگ نیز می توان از گراف وضعیت استفاده کرد که در این صورت تابع فوق به صورت زیر نمایش داده می شود :



مثال)

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



تورینگ محاسباتی

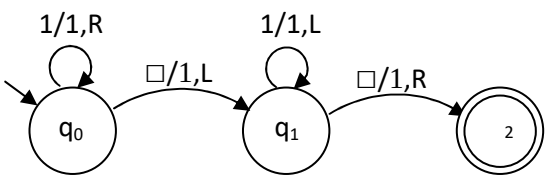
منظور از طراحی ماشین تورینگ محاسباتی، طراحی تورینگی است که توابع ریاضی را پیاده سازی می کند یعنی پس از طراحی اگر ماشین عددی را به عنوان ورودی دریافت کند باید عدد صحیحی را به عنوان خروجی تولید کند.در ماشین های تورینگ هر عدد با تعداد یک هایی که روی نوار قرار می گیرد و مشخص می شود. مثلا برای اینکه یک ماشین تورینگ عدد ۳ را دریافت کند ودنباله ۱۱۱ را روی نوار قرار می دهیم.

اگر توابع ریاضی دو پارامتری باشند ($f(x,y)$) برای دادن دو عدد به ماشین تورینگ، این دو عدد با یک سمبل صفر از هم جدا می شوند.برای مثال دنباله ی ۱۱۱۰۱۱۱۱ به معنی ورود اعداد ۳ و ۴ خواهد بود.

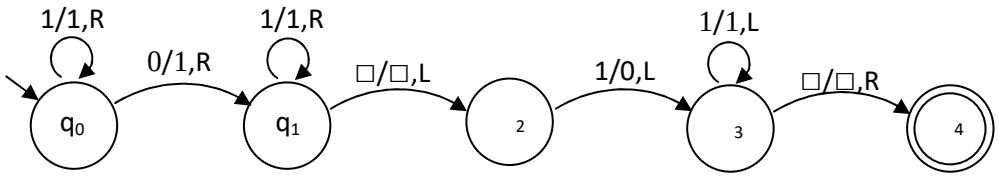
در پایان طراحی ماشین تورینگ محاسباتی، هد در هر حالت پایانی حتما باید به اولین ۱ از دنباله ی جواب اشاره کند.

مثال)

$f(x,y)=x+1$



f(x,y)=x+y



اگر w=uvz نشان دهنده ی یک رشته باشد به هر دنباله ی u یک پیشوند و به هر دنباله z یک پسوند از رشته ی w گویند.هر دنباله یv می تواند یک زیر رشته از رشته ی w باشد.در تعیین پیشوند و پسوند اگر یک رشته دارای طولn باشد، n+1 پیشوند و n+1 پسوند خواهد داشت.

w=abc

مجموعه ی پیشوند ها ={λ,a,ab,abc}

مجموعه ی پسوند ها ={λ,c,bc,abc}

مجموعه ی زیررشته ها ={λ,a,b,c,ab,bc,abc}