
بسم الله الرحمن الرحیم
Joyandeh
online internet shop!!!

فصل اول: زبان‌هـا

۱-۱ الفبا و زبان[1]

تئوری زبان‌ها چیست؟ برای پاسخ به این پرسش ما ابتدا باید بدانیم که زبان چیست؟ وبستر[2] زبان را به این صورت تعریف می‌کند: زبان پیکره ای از کلمات و روش‌های ترکیب کلمات استفاده شده و فهم شده توسط یک جامعه[3] می باشد. هرچند این تعریف بحد کافی برای ساختن یک تئوری ریاضی از زبان کافی نیست و ما باید یک زبان صوری[4] انتزاعی[5] را به عنوان بخشی از یک سیستم یا سامانه تعریف کنیم . این فرمول بندی ما را قادر به ایجاد جمله های دشوار[6] درباره زبانهای صوری و توسعه پیکره ای از دانشی که میتواند برای این زبانها در مدلهای مناسب استفاده شود رهنمون می سازد . این ایده ها بسیار مهم هستند بنابراین لازم است تا مفاهیم کلیدی و اصطلاحاتی را که استفاده میکنیم تعریف کنیم.

فرضیهٔ زبانهای صوری با کوششهای نوام چامسکی[8] در سال‌های ۱۹۵۰هنگامی‌که چامسکی سعی در به‌دست آوردن ویژگیهای مشخصی از ساختار زبانهای طبیعی داشت؛ تکامل یافت. هدف او تعریف نحو[9] زبان با استفاده از قوانین دقیق ریاضی بود و بعدها مشخص شد که نحو زبانهای برنامه نویسی را میتوان با استفاده از مدلهای گرامری چامسکی توصیف کرد. بعدها ریاضی دانانی چون آکسل[10] و پست[11] و کلین[12] سمبلهای باینری را با توجه به ویژگیهای ریاضی رشته ها و مجموعه ها بررسی کردند.

۱-۱-۲ الفبا

الفبا مجموعه ای است متناهی از عناصر ساده ای که تجزیه ناپذیرند که طول هر عنصر آن برابر واحد و یا یک است. پس الفبا بعنوان یک مجموعهٔ متناهی از سمبلها[7] در نظر گرفته شده است . گرچه تعداد نامتناهی غیر قابل شمارش از سمبلها هم وجود دارد ما باید تنها یک زیر مجموعه متناهی قابل شمارش از همگی مجموعه های قابل نمایش را روی کنیم . این زیر مجموعه شامل ارقام ، حروف بزرگ و کوچک و سمبلها و علائم خاصی چون # ،@، . . . هستند .هر تعداد قابل شمارشی از جمله های اضافی که بتوان آنها را مناسب یافت غیر قابل اضافه کردن به این مجموعه است . مجموعه الفبا را معمولا با Σ نشان میدهند.

$$\Sigma = \{0,1, \ldots ,9,a, \ldots ,z,+,*,\%,Div , Mod , If , Then , Else, \ldots \}$$

---

[1] alphabet and language
[2] Webster
[3] Community
[4] Formal Language یعنی زبانهای قراردادی یا رسمی و ظاهری مانند زبانهای طبیعی یا کامپیوتری که از قواعد نحوی خاصی پیروی میکنند .
[5] Abstractly
[6] Rigorous Statements
[8] Noam Chomsky
[9] Syntax
[10] Axel
[11] Emil Post
[12] Stephen Kleene
[7] Symbols

۱-۲ رشته [۱]

دنباله متناهی از عناصر الفبا (متعلق به یک مجموعه الفبا) تشکیل یک رشته را میدهند. طول رشته برابر عناصر الفبای موجود دررشته است.

$\Sigma=\{0,1\}$ $\omega 1=1010$, $Length(\omega 1)=4$

λ رشته نول [۲] یا تهی رشته ای است که دارای هیچ سمبلی نباشدو بـا λ نـشان داده میـشود و دارای طـول صـفر است.و هیچگاه نمیتواند جزو الفبا باشد .

عناصر الفبا را سمبلهای پایانی [۳] مینامند . در زبانهای طبیعی ، کلمات الفبای زبان را تشکیل میدهند و در زبـان کامپیوتر الفبای زبان معمولاً Token نامیده میشود . مثلاً در زبان پاسکال شامل کلمـات کلیـدی [۳]، شناسـه گرهـا [۴] و سمبلهای خاص [۵] مانند /,&,%,$,#,@,! و ... میباشد .

$\lambda \in \Sigma^*$

Σ : مجموعه کلیه رشته های قابل تولید از الفبای Σ که نامتناهی میباشد.

$\Sigma=\{1\}$ $\Sigma^*=\{1,11,111, \ldots, \underbrace{1\ldots1}_{n}, \ldots\}$

Σ^* شامل λ هم میباشد و Σ^+ مجموعه $\Sigma^*-\{\lambda\}$

۱-۳ زبان

هر مجموعه از رشته ها روی الفبا یک زبان می باشد. اغلب زبانهای مورد بررسی شامل تعداد نامتناهی جمله هستند. سه پرسش بسیار مهم در اینجا قابل طرح است:

۱ – چگونه میتوانیم یک زبان را نمایش دهیم؟ اگر زبان تنها شامل تعداد متناهی جمله باشد پاسخ ساده است: لیستهای ساده ای از مجموعه های متناهی از رشته ها. به‌عبارت دیگر اگر زبان نامحدود باشد ما با مساله چگونگی پیدا کردن نمایش متناهی برای زبان مواجه هستیم. این نمایش محدود بخودی خود معمولاً یک رشته از سمبلها روی الفبا باشد و همراه با برخی تفسیرهای قابل فهم که مرتبط با یک نمایش خاص از زبان مفروض میباشد.

۲ – آیا یک نمایش متناهی برای هر زبان وجود دارد؟ از یک جنبه شاید پاسخ منفی باشد. ما باید ببینیم که مجموعه همگی جمله ها روی یک الفبا نامتناهی و قابل شمارش باشد. یک زبان زیر مجموعه همگی جمله ها و رشته ها میباشد و این جنبه معین و مشخص در تئوری مجموعه هاست که مجوعه همگی زیرمجموعه های یک مجموعه نامتناهی شمارش پذیر ، شمارش پذیر نامتناهی نیست . هرچند ما تعریف نکردیم که چه چیز جانشین یک نمایش محدود و متناهی است. ما میدانیم که هر تعریف با معنی از نمایش متناهی تنها در یک تعداد شمارش پذیر از نمایشهای متناهی نتیجه خواهد داد چون باید قادر به نوشتن چنین نمایشهایی به رشته ای از سمبلها باشیم. بنابراین، تعداد زیادی زبان با نمایش‌های متناهی وجود دارند.

۳ – درباره ساختار کلاس‌های زبان که نمایش‌های متناهی دارند چه میتوان گفت؟

---

[۱] String
[۲] Null string
[۳] Terminal symbol
[۳] Keyword
[۴] Identifier
[۵] Special symbol

۱-۳-۱ روش تولید رشته های متعلق به Σ

عملگرها
- اتحاد دومجموعه از رشته ها
- اتحاد دو مجموعه از رشته ها
- ستاره (*)
- بعلاوه (+)

❑ مثال ۱-۱ : $\Sigma=\{a,b\}$

$X=\{aa,ba\}$　　　　$Y=\{bba,a\}$

$X\cup Y = Y\cup X=\{aa,ba,bba,a\}$ اتحاد

$XY=\{aabba,aaa,babba,baa\}$ اتصال

$YX=\{bbaaa,bbaba,aaa,aba\}$

بنابر این اگر X,y دو مجموعه از رشته ها از مجموعه الفبای Σ باشند:

I: $X\cup Y=\{\omega | \omega\in X \text{ or } \omega\in Y\}$
II: $XY=\{\omega | \omega=\alpha 1\alpha 2 \ ; \alpha 1\in X, \alpha 2\in Y\}$

- باید توجه داشته باشیم که X,Y هر دو از یک Σ ساخته شده اند.

❑ مثال ۱-۲ : $\Sigma=\{a,b\}$　$X=\{\lambda\}$　$Y=\{aa,bb,\lambda\}$

$XY=\{\lambda\}\{aa,bb,\lambda\}=\{\lambda aa,\lambda bb,\lambda\lambda\}=\{aa,bb,\lambda\}$

یعنی λ عضو خنثی عمل اتصال است.

$\lambda X=X\lambda=X$
$X0=\lambda$

$X_1=\{a,b\}$　　طول =۱

$X_2=X_1X_1=\{a,b\}\{a,b\}=\{aa,ab,ba,bb\}$　　طول =۲

$X_3=X_2X_1$　　طول=۳

.
.

$X_n=X_{n-1}X_1$　　طول = n

$\cup \Rightarrow \Sigma^*$

۱-۳-۲ اتصال دورشته

اگر $u,v\in\Sigma^*$ در اینصورت uv به صورت زیر تعریف میشود:

۱ - اگر length(u)=0 یعنی $u=\lambda$ در اینصورت $uv=\lambda v=v\lambda=v$

۲ - اگر length(u)=0 یعنی $u=\lambda$ در اینصورت $a\in\Sigma^*$عنصر الفبا و رشته $\omega\in\Sigma^*$ وجود دارند به‌قسمی‌که:

$length(\omega)=length(u)-1$　　　　$u=a\omega$

$\Rightarrow uv=(a\omega)v=a(\omega v)$

این روش اتصال دورشته را روش بازگشتی اتصال دورشته گویند.

❑ مثال ۱–۳ : $v=ba$, $u=ac$ $\Sigma=\{a,b,c\}$

$uv=(ac)(ba)=a(\,(\,c)\,ba))=a((c\lambda)(ba))=a(c((\lambda)ba)))=a(c(ba))$
$=a(cba)=acba$

$\Sigma^*=\bigcup_{i\geq 0} X_i \quad ; \quad X_i=X_1.X_1...X_1$

$X_0=\{\lambda\}$
$X_1=\Sigma$
$X_2=\Sigma\Sigma=\Sigma^2$
$X_3=X2X1=\Sigma^3$
.
.
$X_n=\Sigma^n$

$\Sigma^* =(\bigcup_{k\geq 1}\Sigma^k) \cup\{\lambda\}$

یعنی Σ^* اتحاد اتصال‌های مکرر Σ با خودش و رشته λ است. و هم چنین Σ^+ اتحاد اتصال‌های مکرر Σ با خودش است.

❑ مثال ۱–٤ :

$\Sigma=\{a,b\} \quad x=\{\},y=\{a\} \quad \varnothing\cup u=u\ ,\ \varnothing u=u\varnothing=u$
$x\cup y=\{a\} \quad xy=\{\}$
$\Sigma=\{a,b\} \quad x=\{\lambda\},y=\{a\}$
$x\cup y=\{\lambda,a\} \quad xy=\{a\}$

عملگرها
۱ –اتحاد
۲ – اتصال → مجموعه ها / رشته ها
۳ – *
٤ – +

❑ مثال ۱–۵ : با فرض $\Sigma=\{0,1,a\}$ مطلوبست

➢ مجموعه ای از رشته ها به‌طول ۳

$$X=(\Sigma\Sigma)\Sigma=\Sigma^2\,\Sigma=\Sigma^3=\{0,1,a\}\{0,1,a\}\{0,1,a\}$$

➢ مجموعه ای از رشته ها به‌طول ۲ یا ۳

$$Y=\Sigma^3\bigcup\Sigma^2=(\{0,1,a\}\{0,1,a\}\{0,1,a\})\bigcup(\{0,1,a\}\{0,1,a\})$$

➢ مجموعه ای از رشته ها که طول آنها مخالف ۲ و ۳ است .

$$Z=\{0,1,a\}^*-Y$$

- مجموعه ای از رشته ها بطول زوج

$$T = (\{0,1,a\}\{0,1,a\})^*$$

- مجموعه ای از رشته ها بطول فرد

$$E = \{0,1,a\}^* - (\{0,1,a\}\{0,1,a\})^* = (\sum^2)^* \sum$$

- مجموعه ای از رشته ها که حتماً شامل زیر رشتهٔ a1 باشند

$$Y = \{a\}\{1\} \quad , X = \{0,1,a\}^*, Z = \{0,1,a\}^*$$
$$w = X.Y.Z$$

- مجموعه ای از رشته ها که زیر رشتهٔ a1 در آنها ظاهر نشود

$$Q = \sum^* - W$$

- مجموعه ای از رشته ها که زیررشتهٔ a1 فقط و فقط دو بار ظاهر شود

$$Q\{a\}\{1\}\}Q\{a\}\{1\}Q\{a\}\{1\}$$

- مجموعه ای از رشته ها که با {1}{a} شروع یا به {a}{1} ختم شوند

$$(\{1\}\{a\}\{0,1,a\}^*) \bigcup (\{0,1,a\}\{a\}\{1\})$$

- مثال ۱–٦ : اگر $\Sigma=\{a\}$ مطلوبست تولید مجموعه رشته های به‌طول زوج

$y=\Sigma\Sigma=\{aa\}$

$\rightarrow x=\{\bigcup_{n\geq1} y^n) \cup\{\lambda\}\rightarrow x=y^*$

- مثال ۱–۷ : اگر $\Sigma=\{a,b,c\}$ مطلوبست مجموعه رشته های زیر

1. $x= \{\omega|\omega\lambda\in\Sigma^*, \text{length}(\omega)=3\}$
2. $y=\{\omega|\omega\lambda\in\Sigma^*, \text{length}(\omega)=3k, k\geq0\}$

$1 \rightarrow x=\{a,b,c\}\{a,b,c\}\{a,b,c\}$

$2 \rightarrow y=x^*$

- مثال ۱–۸ : اگر $\Sigma=\{a,b,c\}$ مطلوبست تولید رشته های زیر :

الف) مجموعه کلیه رشته‌هایی که فقط از a,b تشکیل شده باشند.

$X=\{a,b\} \qquad y=x^+$

ب) مجموعه کلیه رشته هایی که با a شروع و به b ختم شده باشند.

$X=\{a\}\Sigma^*\{b\}$

ج)مجموعه تمامی رشته هایی که در آنها ab حداقل یکبار ظاهر شده باشند

$\Sigma^*\{a\}\{b\}\Sigma^*=\Sigma^*\{ab\}\Sigma^*$

- مثال ۱–۹ : روی الفبای {a,b,c} رشتهٔ $\{a,b\}^*\{c\}^+$ دنباله ای از عناصر الفبا میباشد که با تعدادی (صفر یا بیشتر) a و b شروع شده و در انتهای آنها دنباله ای از c (حداقل یک c) قرار دارد.

۱-٤ مجموعه های با قاعده[1] یا منظم

تعریف: یک مجموعه را با قاعده گویند اگر آنرا بتوان از المانهای الفبا با استفاده ازاتصال[2] و عمل * تولید کرد. مجموعه های باقاعده بخش مهمی از زبان را تشکیل می دهند هم در نظریه زبانهای صوری و هم نظریه ماشینهای متناهی[3] کاربرد دارند. مجموعه با قاعده از ترکیب مجموعه تهی و مجموعه‌های تکین[3] همراه با عملیات مجاز روی مجموعه ها به دست می آیند.

تعریف :اگر Σ مجموعه الفبا باشد مجموعه با قاعده روی Σ به شکل بازگشتی زیر می‌باشد:

۱ . به ازای هر عنصر $\{a\},\{\lambda\},\varnothing,\ a\in\Sigma$ مجموعه های باقاعده هستند.← مجموعه های ابتدایی

۲ . اگر x,y مجموعه های باقاعده باشند در اینصورت $x^*,x^+,xy,x\cup y$ هم با قاعده هستند.← مجموعه های بازگشتی

۳ . x یک مجموعه باقاعده میباشد، اگر بتوان آنرا تنها با استفاده از المانهای ابتدایی یا مجموعه های ابتدایی با تعداد محدودی از اعمال مکرر عملگرهای اتصال ، اتحاد، *،+ تولید کرد.

- مثال ۱-۹ : مجموعه ای از رشته ها که با دنباله هایی از a شروع و بلافاصله با دنباله ای از b ختم می شوند با قاعده هستند؟

$\Sigma=\{a,b\}$
$\omega=a\ldots ab\ldots b$
$x=\{a\}^*\{b\}^*$

- مثال ۱-۱۰: اگر در سوال قبل تعداد aها و b ها برابر باشند مجموعه ها با قاعده هستند؟

$\omega=a\ldots ab\ldots b=\{a\}^n\{b\}^n$

خیر زیرا با هیچیک از عملگرهای اتحاد ، اتصال ، *،+ نمیتوان ساخت.پس نتیجه میگیریم که همهٔ زیرمجموعه های Σ^* را نمیتوان با استفاده از عملگرهای اتحاد و اتصال و* و + تولید کرد .

- مثال ۱-۱۱ : $\Sigma=\{a,b,c\}$ مجموعه با قاعده ω را تولید کنید به‌قسمی‌که ω شامل رشته هایی از Σ باشد که c تنها و تنها یکبار در آنها ظاهر شده باشد.

$\{a\},\{b\},\{c\},\{\lambda\}$ مجموعه های ابتدایی $\omega=\ldots c\ldots$ یعنی

$\omega=\{a,b\}^*\{c\}\{a,b\}^*$

- مثال ۱-۱۲ : با فرض $\Sigma=\{a,b,c\}$ مجموعه با قاعده ω' را تولید کنید به‌قسمی‌که در آن رشته ها تعداد c زوج باشد.

$\omega'=(\omega\omega)^*\{a,b\}^*$

اتصال با رشته هایی که تعداد c آنها صفر است.

---

[1] Regular set
[2] Concatenation
[3] Finite state machine
[3] singletone

- مثال ۱–۱۳ : با فرض $\Sigma=\{a,b,c\}$ مجموعه با قاعده ω'' بقسمیکه در آن رشته ها تعداد c فرد باشد.

$\omega''=(\omega\omega)^*\omega$

- مثال ۱–۱٤ : مجموعه های باقاعده ای که در رشته های آن زیر رشته های aa,bb حداقل یکبار ظاهر شوند.مانند aaababbb . (a,b به شکل کلی قابل قبول نیستند)

$x=((aabb)^+ \cup (abab)^+ \cup (abba)^+ \cup (baab)^+ \cup (baba)^+)^+$

۱–۵ عملگر تفاضل

برای تولید مجموعه های با قاعده معمولا از این عملگر استفاده می کنیم . مثلاً

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

x={ω|تعداد c هازوج باشد}

$y=\Sigma^*-x$ ، y={ω|تعداد c هافرد باشد}

عبارات باقاعده[۱] ← مفهوم جدیدی از مجموعه با قاعده است

تعریف :اگر Σ الفبا باشد عبارت با قاعده روی Σ به شکل بازگشتی زیر قابل تعریف است:

۱ – برای هر عنصر الفبا $\varnothing,\lambda,a,a\in\Sigma$ عبارت های باقاعده هستند← عبارت با قاعده ابتدایی

۲ – اگر E2,E1 دو عبارت با قاعده باشند در اینصورت عبارتهای E1.E2,E1UE2, E1*,E2+ نیز با قاعده هستند.

۳ – Eعبارت با قاعده است اگر بتوان آنرا فقط از عبارات ابتدایی با قاعده و توسط اعمال مکرر عملگرهای اتصال، اتحاد ، +، * بدست آورد.

- مثال ۱–۱۵ : $\Sigma = \{a,b,c\}$

a, b,c,λ و $\{a\},\{b\},\{c\},\{\lambda\}$ مجموعه های با قاعده هستند .

- مثال ۱–۱٦ : عبارت با قاعده‌ای که عناصر آن فقط از a درست شده باشد.

$$X' = a^+$$

- مثال ۱–۱۷ : مجموعه های با قاعده ای که رشته های آن تنها از a درست شده باشند

$$X = \{a\}^+$$

- مثال ۱–۱۸ : عبارت با قاعده ای که عناصر آن بطول ۳ هستند

y' مجموعه با قاعده ای که رشته های آن بطول ۳ هستند.

abc,aba,. . .

$y=\{\{a\}\{b\}\{c\}\{b\}\{a\}\{c\},...\}$ یا $y=\{a,b,c\}\{a,b,c\}\{a,b,c\}$

$y'=(a\cup b\cup c)(a\cup b\cup C)(a\cup b\cup c)$
$E = (a\cup b) \Rightarrow E= a,b$
$E1= (a\cup b)$, $E2=c$
$E=E1E2=(a\cup b)c \Rightarrow E=ac,bc$
$E=(a\cup b)(a\cup b)c$
$\Rightarrow E=aac,abc,bac,bbc$

- مثال ۱–۱۹ : $\Sigma=\{a,b,c\}$

---

[۱] Regular Expression

E را طوری بسازید که aa تنها وتنها یکبار ظاهر شود و a در جای دیگر ظاهر نشود.

$E=(b\cup c)^*aa(b\cup c)^*$

- مثال ۱–۲۰ : عبارت با قاعده ای بنویسید که یا با aa شروع شود یا به bb ختم گردد.

$\in 1=aa...$ 		$E1=aa(aUbUc)^*$ 		E2 در E1 مستتر است

$\in 2=...bb$ 		$\Rightarrow E=E1UE2$

$\in 3=aa...bb$ 		$E2=(aUbUc)bb$

E3 در E2 مستتر است

- اگر در مثال فوق یای انحصاری یا Xor باشد.

$E3 = aa(a\cup b\cup c)^*bb$

$E' =(E1UE2)-E3$

- مثال ۱–۲۱ : عبارت با قاعده ای بنویسید که فاقد Ca باشد (فقط با استفاده از اتحاد ،اتصال ،×،+)

$(a\cup C^*b^*)^*C^*$

- مثال ۱–۲۲ : عبارت با قاعده ای که در آن aحتما پیش از b و b حتما پیش از c ظاهر شود

$((a)^*(b)^*(c)^*) \cup ((c)^*(a)^*)$

اگر a,b با هم آمدند aحتما قبل از b و b حتما قبل از c باشد

- مثال ۱–۲۳ : مجموعهٔ باقاعده ای بنویسید که رشته های ان شامل حروف a,b,c باشد به‌طوری‌که a قبل از c و b هم قبل از a رخ دهد 		$\{c,b\}^*\cup\{b\}^*\{a\}^*\{c\}^*$

- مثال ۱–۲٤ : $\Sigma=\{a,b\}$ و مجموعه $\{ba\omega ab|\ \omega\in\{a,b\}^*\}$ روی Σ با قاعده است.

عبارت	مجموعه
A	{a}
B	{b}
Ab	{a}{b}
$a,b=a\cup b$	$\{a\}\cup\{b\}=\{a,b\}$
Ba	{b}{a}={ba}
$(a\cup b)^*$	{a,b}*
$ba(a\cup b)^*$	{ba}{a,b}*
$(ba)(a\cup b)^*(ab)$	{ba}{a,b}*{ab}

- مثال ۱–۲۵ :عبارتی را نشان دهید که دقیقا دو تا b در آن ظاهر شده باشد

$a^*(ba^*ba^*)$

عبارت‌های با قاعده ای که از یک مجموعه به دست می آیند، یکتا نیستند و دو عبارت که یک مجموعه را نشان میدهند همانند هستند. جدول زیر همانندی عبارت ها را نشان می دهد.

جدول همانندی‌های عبارت‌های باقاعده[1]

1- $\varnothing u=u\varnothing=\varnothing$	13-$(\lambda^*\varnothing^*)^*=\lambda$
2- $\lambda u=u\lambda=u$	14-$\lambda-\{\varnothing^*\}=\lambda-\{\lambda\}=\{\lambda\}$
3- $\varnothing^*=\varnothing$	15- $u^*=(u^*)^*$
4- $\lambda^*=\lambda$	16- $u(v\cup\omega)=uv\cup u\omega$
5- $u\cup v=v\cup u$	17- $(u\cup v)\omega=u\omega\cup v\omega$
6- $u\cup\varnothing=\varnothing\cup u=u$	
7- $u\cup u=u$	
8-$u^*\cup\varnothing^*=u^*$	
9-$(\lambda\cup u)^*=u^*$	18-$(uv)^*u=u(vU)^*$
10-$(\varnothing\cup u)^*=u^*$	19- $(u\cup v)^*=(u^*\cup v)^*=(u^*\cup v^*)^*$ $=u^*(u\cup v)^*=(u\cup vu^*)^*$ $=(u^*v^*)^*=u^*(vu^*)^*=(u^*vu^*)^*$
11-$u^*.u=u.u^*=u+$	20- $u.(v+\omega)=uv+u\omega$
12- $u.u^*+\lambda=u^*$	21- $(u+v)^*=(u^*+v^*)^*=(u^*.v^*)^*=u^*.(vu^*)^*$

مسائل فصل اول

۱-۱ نشان دهید $(u\cup v)^*=(u^*\cup v^*)$

۱-۲ نشان دهید $(u\cup v)^*=(u^*.v^*)^*$

۱-۳ نشان دهید $(uv)^*=u^*.v^*$ همشه برقرار نیست .

۱-٤ نشان دهید $u^*.v=v\cup u^*.uv$

۱-٥ نشان دهید $\phi^* = \lambda$

۱-٦ نشان دهید $A.(B \cup C)=A.B \cup A.C$

۱-۷ نشان دهید $u(vu)^*=(uv)^*.u$

۱-۸ با فرض $\Sigma=\{a,b\}, L1=\{a,ab,abb\}, L2=\{\lambda,b,a,bb\}$ آن گاه L1.L2 را بنویسید.

۱-۹ ثابت کنید عبارت $r=a^*(a+b)$ منظم است.

۱-۱۰ ثابت کنید $r=(0+1)^*(0+\lambda)$ منظم است .

۱-۱۱ فرض کنید $L1=\{10,1\}$ و $L2=\{011,11\}$ در اینصورت :

۱-۱۲ با فرض آنکه $\Sigma=\{0,1\}$ مطلوبست:

- عبارت منظمی بنویسید که از صفرها و یک ها تشکیل شده باشد .
- عبارت منظمی بنویسید که از صفرها و یک ها تشکیل شده و دارای حداقل دو صفر متوالی است .

---

[1] Regular Expression Identities

- عبارت منظمی بنویسید که از صفرها و یک ها تشکیل شده و دارای حداقل دو صفر است .

- عبارت منظمی بنویسید که شامل رشته ای از صفرها و یک ها هستند و با یک شروع میشوند و شامل دو صفر متوالی نیستند .

- عبارت منظمی بنویسید که شامل رشته ای از صفرها و یک ها هستند و شامل دو صفر متوالی نستند .

- عبارت منظمی بنویسید که از صفرها و یک ها تشکیل شده و به 011 ختم میشوند .

- عبارت منظمی بنویسید که نشاندهندهٔ مجموعه رشته هائی باشد که با صفر شروع میشوند و با یک خاتمه می یابند .

- عبارت منظمی بنویسید که نشاندهندهٔ مجموعه رشته های دودوئی غیر تهیی باشد که با بیت یکسانی شروع و خاتمه می یابند .

- مجموعه تمامی رشته هائی که یک ندارند .
- مجموعه تمامی رشته هائی که دقیقاً یک 1 دارند.

- مجموعه تمامی رشته هائی که دقیقاً دو 1 دارند.

- مجموعه تمامی رشته هائی که حداقل دو 1 دارند.

- عبارتهای منظمی شامل مجموعه رشته هائی که صفرهای متوالی ندارند .

۱-۱۳ با فرض آنکه $u,v \in \sum^*$ ثابت کنید :

$$(uv)^R = v^R u^R$$

منظور از نماد R وارون رشته است .

۱-۱٤ عبارت منظمی روی $\Sigma = \{0,1\}$ بنویسید که شامل زیر رشتهٔ 101 باشد .

۱-۱٥ ثابت کنید $(b^*(a \cup \lambda)b^*)^* = (a \cup b)^*$

۱-۱٦ با فرض آنکه $\Sigma = \{0,1,2\}$ باشد عبارت با قاعده ای بنویسید که شامل یک 0 و هر تعداد 1 و 2 باشد.

۱–۱۷ عبارت باقاعدهٔ L را بنویسید که همهٔ رشته های aو b را شامل شود و تعداد فردی کاراکتر b داشته باشد .

۱–۱۸ عبارت های منظم زیر را در نظر بگیرید:

R1=b*a(a+b)*
R2=(a+b)*a(a+b)*
R3=(a*b*)*ab*
R4=(a+b)*ab*

این عبارات نشاندهندهٔ چه رشته هائی هستند و کدامیک با هم همانند هستند ؟

۱–۱۹ با فرض آنکه α و β نشاندهندهٔ عبارات منظم باشند ؛ کدام یک از برابری های زیر ممکن است همیشه برقرار نباشد؟

1) $(\alpha+\beta)^* = \alpha^*(\beta\alpha)^*$
2) $(\alpha+\beta)^* = (\alpha^*\beta^*)^*$
3) $(\alpha+\beta)^* = \alpha^*(\beta\alpha^*)^*$
4) $(\alpha+\beta)^* = (\alpha^*+\beta^*)^*$

۱–۲۰ نشان دهید $(ba)^+.(a^*b^*\cup a^*) = (ba)^*.ba^+(b^*\cup\lambda)$

۱–۲۱ با فرض آنکه ω رشته ای دلخواه باشد ثابت کنید: $\forall i \geq 0 \quad (\omega^R)^i = (\omega^i)^R$

فصل دوم: گرامرهای مستقل از متن و پارسرها

مقدمه

گفتیم زبان مجموعه ای از رشته هایی است که ساختار خاصی را دارا هستند. مثلاً روی الفبای $\sum = \{a,b,c\}$ مجموعه های $\{a\}^n\{b\}^n$ و b* و {a,b,c}* وa+ همگی زبان هستند. برای تعریف زبان باید یک روش استاندارد مورد استفاده قرار بگیرد و علاوه بر روش تعریف زبان باید روشهایی را برای تولید عبارت‌های زبان از تعریف و ارزیابی تعلق یک نمونه یا رشته به زبان نیز تعریف کنیم . پیچیدگی یک زبان با پیچیدگی ساختار آن رابطه مستقیم دارد اگر مجموعه نا متناهی باشد حتما دارای ساختار است.

- **اجزای لازم برای تعریف یک زبان**

الفبای زبان
ساختار زبان
روش تولید رشته های زبان (جمله)
روش ارزیابی تعلق یک رشته به زبان
روش درک معنا

هدف دستیابی به تئوری جامع برای تعریف یک زبان

۲-۱ تئوری چامسکی

آبرام نوام چامسکی[۱] در هفتم دسامبر ۱۹۲۸ در پنسیلوانیا به دنیا آمد. او تحصیلات خویش را در دانشگاه پنسیلوانیا انجام داد و تحت تاثیر استاد خویش زلیگ هریس[2] قرار گرفت. فوق لیسانس خویش را در سال ۱۹۵۱ به پایان برد و به مدت چهار سال (۱۹۵۱–۵۵) در دانشگاه هاروارد به تدریس پرداخت. تا این‌که در سال ۱۹۵۵ موفق به دریافت درجه دکتری از دانشگاه پنسیلوانیا شد. از آن پس در دانشگاه MIT [3] برای سال‌های طولانی به‌عنوان استاد زبانهای جدید و زبان شناس به تدریس پرداخت. چامسکی نخستین گام خویش را در زبان شناسی رسماً با نوشتن رسالهٔ خویش به نام، دستور زبان زایشی عبری نوین[4] برداشت. که البته آنزمان نظر عدهٔ زیادی را به سوی خویش جلب نکرد و تنها تنی چند چون کوئین[5] و گودمن[٦] چامسکی را تشویق به ادامهٔ راه و تکوین فرضیهٔ زبان زایشی کردند. این اثر نخستین سنگ بنای نظریهٔ زبان شناسی نوین گشت.

در چند دههٔ پیشین هیچ زبان شناسی نبوده که از نظر دامنه و عمق نفوذی که دیدگاهایش بر جریان اندیشه و پژوهش در زبان و مسایل آن داشته با چامسکی برابری کند. او امروزه یکی از سرشناس‌ترین زبان شناسان به شمار میرود و شهرت او نه به سبب ابداع دستور زبان گشتاری، بلکه دیدگاههای او در بارهٔ ماهیت و هستی زبان و زبان آموزی سبب بلند آوازگی نام او شده است. بیشترین شهرت چامسکی به سبب دیدگاه‌هایی است که او راجع به

[۱] Abram Noam Chomsky
[۲] Zellig Harris
[۳] Massachusettes Institute of Technology
[٤] The Generative Grammar of Modern Hebrew
[٥] Quine
[٦] Goodman

امکان برخورداری از دانش و پژوهش زبان شناسی در زمینه های علمی دیگر چون روانشناسی، فلسفه و ... در ارتباط با زبان و اندیشه دارد.

چامسکی زبان را با چهار مولفه Σ,V,R,S تعریف می کند که این چهار مولفه در مجموع گرامر زبان نام دارند.

$$G_L = (\Sigma, V, R, S)$$

Σ : الفبای زبان

V : مجموعه متناهی از عناصر به نام متغییر[1]

R: بیانگر ساختار زبان و هدایت کننده عملیات و تولید جمله های زبان[2]

S[3] : بعنوان مبدا تولید جمله های زبان $S \in V$ یعنی عنصری از V

R مجموعه ای است متناهی از قوانین جایگزینی به صورت زیر :

$u \rightarrow v$

رشته $u \in (\Sigma \cup V)^+$

رشته $v \in (\Sigma \cup V)^*$

این مجموعه از قوانین جایگزینی ساختار زبان را به شکل سلسله مراتبی تعریف می کنند.(از بالا به پائین) در صورت لزوم u را میتوان به وسیله v جایگزین ساخت.

❑ مثال ۲-۱ :

$G=(\Sigma,V,R,S)$; $\Sigma=\{a,b,c\}$; $V=\{S,A,B\}$

R={ S →AB , 1
A→aa , 2
B→bb , 3
A→c , 4
B→b , 5 }

```
  S
  ↓
A    B
↓    ↓
aa   B
↓    ↓
aa   b
```

جریان جایگزینی های مکرر

در این سطح دیگر جایگزینی ممکن نیست در اینجا کار متوقف شده و به دنباله ای از الفبا رسیدیم که جمله آن متعلق به زبان است یعنی aab متعلق به زبان میباشد.

[1] Variable
[2] Production
[3] Start Symbol

$$\begin{array}{ccc} & S & \\ & \downarrow & \\ A & & B \\ & & \downarrow \\ c & & bb \end{array}$$

cbb جمله ای از زبان

S را بعنوان تمامی رشته ها یا جمله های زبان تعریف می کنیم. جمله های زبان در یک زنجیره از جایگزینی‌های مکرر که بوسیله قوانین جایگزینی هدایت میشوند از S تولید می گردند و قوانین جایگزینی تضمین می کنند که جمله های تولید شده از ساختار خاص زبان تبعیت می کنند. گرامر یک زبان باید بتواند تولید همگی جمله های زبان را تضمین کند.

❑ مثال ۲–۲ : گرامر زبان زیر را طراحی کنید

$\sum=\{a,b\}$
L=a*b*
$GL=(\sum,V,R,S)$ $\sum=\{a,b\},V=\{S,A,B\}$

A,B را اصطلا حا متغیرهای بازگشتی[1] می گویند.

R:{$S\rightarrow AB$,
$A\rightarrow aA$,
$A\rightarrow \lambda$,
$B\rightarrow bB$,
$B\rightarrow \lambda$}

$$\begin{array}{ccccc} & & S & & \\ & & \downarrow & & \\ A & & & & B \\ \downarrow & & & & \downarrow \\ a & & A & & \lambda \\ \downarrow & & \downarrow & & \downarrow \\ a & a & A & & \lambda \\ \downarrow & \downarrow & \downarrow & & \downarrow \\ a & a & a & A & \lambda \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ a & a & a & \lambda & \lambda \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \end{array}$$

❑ مثال۲–۳ :

R:

<جمله> → <فاعل> <مفعول> را <فعل> .

<مفعول> → <اسم>

<فعل> → <u>خورد</u>

<فعل> → <u>خرید</u>

<فاعل> → <اسم>

<اسم> → <u>کتاب</u>

<اسم> → <u>حسن</u>

کلمه هائی که زیر آنها خط کشیده شده عناصر الفبا فرض شده اند.

[1] Recursive Variable

$\sum$ = {حسن، کتاب، خرید، خورد، را ، .}

V={< اسم > ، < فاعل > ، < مفعول > ، < فعل > ، < جمله >}

S={< جمله >}

< جمله > → <فاعل> <مفعول> را <فعل> .

<فاعل> <مفعول> را خرید.

<فاعل> <اسم> را خرید.

<فاعل> کتاب را خرید.

<اسم> کتاب را خرید.

حسن کتاب را خرید.

- انواع گرامرها صرفاً ساختار ظاهری زبان را به ما میدهند و معانی زبان برای ما نامشخص است ولی در زبانهای ساده معانی زبان در ساختار آن گنجانده میشود. گرامر یک زبان قادر به توصیف ساختار ظاهری زبان است ولیکن نمیتواند ساختار معنائی یا درونی زبان را بیان و فرمولبندی کند.

❑ مثال ۲-٤ : یک گرامر ساده برای زبان لاتین

$\sum$={eat ,with , the , book ,ali}
V={S,M,N,Q,p,v}

R:{S→N V
N→D M
v→ p Q
Q→D M
p→eat
p→with
D→the
M→book
M→ali}

S
N *V*
D *M* *V*
the *M* *p* *Q*
the *book* *p* *Q*
the *book* *p* *Q*
the *book* *with* *Q*
the *book* *with* *D* *M*
the *book* *with* *the* *M*
the *book* *with* *the* *book*

جمله از نظر ساختار صحیح ولی از نظر معنا درست نیست. گرامر یک زبان بیانگر ساختار یک زبان است. اگر معنا ی مورد نظر در زبان را بتوان در گرامر به گونه ای ذخیره کرد در اینصورت تئوری ارائه شده جامع ترین تئوری برای زبانها می باشد. اما در زبانهای طبیعی نمیتوان معنا را در گرامر گنجاند.برای تشخیص یک جمله زبان کافیست سعی کنیم جمله را از S به‌وسیله R تولید کنیم.

- مثال ۲-۵ : گرامر $\begin{cases} S \rightarrow aSb \\ S \rightarrow ab \end{cases}$ زبان a^nb^n را تعریف میکند.

- مثال ۲-٦ گرامر G زبانی را تولید می کند که جمله های آن اگر با a شروع شود با b و اگر با b شروع شود با a خاتمه می یابد و تعداد a ها و b ها با هم برابر می باشد. ababab baabab

$$G: \begin{cases} S \rightarrow aSb \\ S \rightarrow bSa \\ S \rightarrow ab \\ S \rightarrow ba \end{cases}$$

- مثال ۲-۷ :

$$\Sigma=\{a,b,c\}$$
$$L=(aUb)^*cc(aUb)^*$$
$$\begin{cases} S \rightarrow ABA \\ B \rightarrow cc \\ A \rightarrow a|ba \\ A \rightarrow \lambda \end{cases}$$

$$L=\{\omega|Length(\omega)=3\}$$
$$\Sigma=\{a,b,c\}$$
$$\begin{cases} S \rightarrow AAA \\ \left.\begin{array}{l} A \rightarrow a \\ A \rightarrow b \\ A \rightarrow c \end{array}\right\} \equiv A \rightarrow a|b|c \end{cases}$$

- مثال۲-۹ :

$$L'=\{\omega|Length(\omega)=3\alpha\ ;\ \alpha\in N\}\quad \begin{cases} S \rightarrow AAAS \\ S \rightarrow \lambda \\ A \rightarrow a \\ A \rightarrow b \\ A \rightarrow c \end{cases}$$

۲-۲ دسته بندی زبانها یا سلسله مراتب چامسکی[1]

زبانهای موجود را به ٤ دسته می توان تقسیم کرد (این گروه بندی بر اساس درجه پیچیدگی زبان‌ها می‌باشد):

۱ – زبان‌های با قاعده [۲] (Type_3) T3

۲ – زبان‌های مستقل از متن [3] (Type_2) T2

۳ – زبان‌های وابسته به متن [4] (Type_1)T1

٤ – زبان‌های بدون محدودیت[۱] (Type_0) T0

---

[۱] Chomsky Hierarchy
[۲] Regular Language
[۳] Context Free Language
[٤] Case Sensitive Language

تفاوت این زبانها در ساختارشان است. ساختار زبانهای با قاعده بمراتب ساده تر است از سایر انواع زبانها .با توجه به این نکته که ساختار زبان توسط قواعد جایگزینی بیان میشود بنابراین باید تفاوت این ٤ دسته را در شکل قوانین جستجوکرد.همهٔ انواع زبان‌های باقاعده، مستقل از متن و وابسته به متن زبان بدون محدوودیت محسوب میشوند ولی عکس آن برقرار نمیباشد.

$$(T1 \cup T2 \cup T3) \subset T0$$

۲-۲-۱ زبان‌های با قاعده

قوانین زبان‌های با قاعده به شکل زیر است:

1 . $A \in V,\ a \in \Sigma \qquad A \to a$
2 . $A, A' \in V,\ a \in \Sigma \qquad A \to aA'$
3 . $S \to \lambda$ در صورتیکه λ متعلق به زبان باشد

❑ مثال ۲-۱۰:

$\left\{ \begin{array}{l} S \to aSb \\ S \to \lambda \end{array} \right.$ با قاعده نیست

❑ مثال ۲-۱۱ :

$\left\{ \begin{array}{l} S \to aS \\ S \to bS \\ S \to \lambda \end{array} \right.$ با قاعده است

❑ مثال۲-۱۲ : $\left\{ \begin{array}{l} S \to Sb \\ S \to \lambda \end{array} \right.$ با قاعده نیست

برای هر زبان با قاعده حداقل یک گرامر با قاعده موجود است . عبارتهای باقاعده معادل با مجموعه های با قاعده و زبانهای با قاعده ساختار ساده ای دارند و این ساختار امکان استفاده از ماشین را برای پردازش زبانهای با قاعده فراهم می سازد.

۲-۲-۲ زبان‌های مستقل از متن

قوانین زبانهای مستقل از متن به شکل زیر هستند:

1 - $A \in V,\ v \in (\Sigma \cup V)+ \qquad A \to v$
2 - $S \to \lambda$ اگر λ عضوی از زبان باشد

زبانهای با قاعده زیر مجموعه ای از زبانهای مستقل از متن هستندولی هیچگاه برابر نخواهند بود. مثلاً $L = a^n b^n$ $n \geq 0$ با قاعده نیست ولی مستقل از متن است.

[1] Unrestricted Language

۲-۲-۳ زبان‌های وابسته به متن

قوانین گرامر زبانهای وابسته به متن به فرم زیر است :

$$u \to v$$

$$u,v \in (\Sigma \cup V)^+$$

شرط تکمیلی $length(u) \le length(v)$

چون طول u همیشه بزرگتر یا مساوی یک است شرط تکمیلی نشان می دهد که v هیچگاه λ نخواهد بود.

زبانهای مستقل از متن که در آنها λ وجود دارد زیر مجموعه زبانهای وابسته به متن نیستند.

۲-۲-٤ زبانهای بدون محدودیت

گرامر و قوانین زبانهای بدون محدودیت به فرم زیر است :

$$u \to v \qquad \begin{cases} u \in (\Sigma \cup V)^+ \\ v \in (\Sigma \cup V)^+ \end{cases}$$

- **بحث کلی**

زبانهای بدون محدودیت $\subset$ زبانهای مستقل از متن $\subset$ زبانهای با قاعده

زبانهای بدون محدودیت $\subset$ زبانهای وابسته به متن

۲-۳ اشتقاق[۱] یا مکانیزم تولید جمله های زبان

- اشتقاق دنباله ای از قوانین جایگزینی است که قادر است از نماد آغازگر جمله تولید نماید.

اشتقاق (1,3,5,6)

$$G: \begin{cases} 1) & S \to ABS \\ 2) & A \to aA \\ 3) & A \to a \\ 4) & B \to bB \\ 5) & B \to b \\ 6) & S \to ab \end{cases}$$

$$S \overset{1}{\downarrow} ABS \overset{3}{\downarrow} aBS \overset{5}{\downarrow} abS \overset{6}{\downarrow} abab$$

هر مرحله از اشتقاق را با نماد $x \longrightarrow y$ نمایش می دهیم.

زنجیره اشتقاق $S \xrightarrow{1} ABS \xrightarrow{3} aBS \xrightarrow{5} abS \xrightarrow{6} abab$

زنجیره اشتقاق: زنجیره ای است که با S شروع و به یک جمله ختم میشود. در یک زنجیره اشتقاق رشته هایی تولید میشوند که به دسته های زیر قابل تقسیم هستند:

۱) جمله[۲] ← رشته ای است که فقط از عناصر الفبا تشکیل شده است

۲) شبه جمله[3] ← رشته هایی هستندکه ازعناصرالفباویامتغییرهاتشکیل شده اند

[۱] Derivation
[۲] Sentence
[۳] Sentential form

- هر زنجیره اشتقاق فقط یک جمله دارد که در آخر آن واقع است. میانی ها شبه جمله ها هستند.
- جمله میتواند λ هم باشد اما شبه جمله نه ، چون طول آن حداقل باید ۱ باشد.

تعریف: جمله ω متعلق به زبان L است اگر بتوان آنرا از نماد آغازگر گرامر زبان L در یک زنجیره اشتقاق تولید کرد.

$\omega \in L$ If $S \Rightarrow \ldots \Rightarrow \omega$

$$S \overset{*}{\Rightarrow} \omega$$

- علت استفاده از درخت اشتقاق
  ۱- تولید جمله های زبان در Case tool هایی که source زبان را تولید میکنند
  ۲- تشخیص تعلق پذیری جمله ها به یک زبان
- نمایش اشتقاق
  ۱- زنجیره اشتقاق (مستقل از نوع زبان)
  ۲- درخت اشتقاق (عموما در مورد زبانهای مستقل از متن)

❑ مثال ۲-۱۳ : آیا ω=aabbabb در مثال قبلی عضوی از L است؟

زنجیره اشتقاق را نمیتوان برای این جمله تولید کرد زیرا از بسط S حتما در پایان کار جمله ab بدست می آید بنابراین $\omega \notin L$

❑ مثال ۲-۱۴ : تعلق پذیری جملهٔ $(a^+b^+)^*ab$ را بررسی کنید.

$S \to ABS \to ABABS \to (AB)^+S \to (AB)^+ab \to (a^+b^+)^+ab$

و $S \to ab$

$(a^+b^+)^+ab \cup ab = (a^+b^+)^*ab$

- زبان گرامر مجموعه جمله هایی است که از نماد آغازین گرامر تحت عمل اشتقاق تولید می شوند.

$$L(G) = \{\omega \mid \omega \in \textstyle\sum^*, S \overset{*}{\Rightarrow} \omega\}$$

❑ مثال ۲-۱۵: زبان گرامر $\left\{\begin{array}{l} S \to aS \\ S \to \lambda \end{array}\right.$ چیست ؟ a^*

❑ مثال ۲-۱۶ : زبان گرامر $\left\{\begin{array}{l} S \to AS \\ A \to aa \\ A \to bb \\ A \to ab \\ A \to ba \\ S \to \lambda \end{array}\right.$ چیست؟

تمامی دنباله های زوجی از ab

$((a\cup b)(a\cup b))^*$

$S\to AS\to AAS\to aabaAS\to aababbAS\to aababbaa$

مثال ۲–۱۷ : زبان گرامر $G\begin{cases} S\to ASBC\\ S\to abc\\ cB\to Bc\\ bB\to bb\\ A\to a\\ cC\to cc\end{cases}$ چیست؟

حل:

1) $S\to ASBC\to aabcBC\to aabBCC\to aabbcc$

2) $S\to ASBC\to aSBC\to aASBCBC\to aaabcBCBC\to aaabBcCBC$

$\to aaabbccBC\to aaabbccBc\to aaabbcBcc\to aaabbBccc\to aaabbbccc$

در حالت کلی

$$
\begin{aligned}
&S\to ASBC\\
&\to AASBCBC\\
&\to AAASBCBCBC\\
&\vdots\\
&\to (A)^n S(BC)^n\\
&\to (A)^n abc(BC)^n\\
&\to a^{n+1}bcBCBC...BC\\
&\to a^{n+1}bBcCBCB...BC\\
&\to a^{n+1}bbccBCB...BC\to ...\\
&\to a^{n+1}bb...bccc...c\to a^{n+1}b^{n+1}c^{n+1}
\end{aligned}
$$

$$a^{n+1}bb..bccc..c\to a^{n+1}b^{n+1}c^{n+1}\Rightarrow L(G)=\{a^{+1}b^{n+1}c^{n+1}:n>0\}$$

مثال: با فرض داشتن گرامر زیر اشتقاق p =baaaabab را بنویسید.

$$\begin{cases} S\to RT\\ T\to aTB\\ TB\to BZ\\ aB\to Ba\\ RB\to bZ\\ Za\to aZ\\ Zb\to bZ\\ ZB\to ab\end{cases}$$

حل:

$$\begin{aligned}S &\Rightarrow RT \Rightarrow RaTB \Rightarrow RaaTBB\\ &\Rightarrow RaaaTBBB \Rightarrow RaaaBZBB\\ &\Rightarrow RaaBaZBB \Rightarrow RaBaaZBB\\ &\Rightarrow RBaaaZBB \Rightarrow bZaaaZBB\\ &\Rightarrow bZaaaabB \Rightarrow baZaaabB\\ &\Rightarrow baaZaabbB \Rightarrow baaaZabB\\ &\Rightarrow baaaaZbB \Rightarrow baaaabZB\\ &\Rightarrow baaaabab\end{aligned}$$

۹۵. نشان دهید برای جملهٔ $\omega = aabbab$ لااقل دو اشتقاق چپ مختلف وجود دارد.

$$G:\begin{cases}S \to bA|\alpha B\\ A \to aS|bAA|a\\ B \to bS|\alpha BB|b\end{cases}$$

حل:

اشتقاق چپ اول $S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabbB \Rightarrow aabbS \Rightarrow aabbaB \Rightarrow aabbab$

اشتقاق چپ دوم $S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabSB \Rightarrow aabbAB \Rightarrow aabbaB \Rightarrow aabbab$

۲-٤ عملیات روی ز بان‌ها

۲-٤-۱ عملگر تفاضل

$$L_1 - L_2 = \{\omega : \omega \in L_1 \wedge \omega \notin L_2\}$$

۲-٤-۲ عملگر مکمل یا متمم

$$\bar{L} = \{\omega : \omega \in \Sigma^* \wedge \omega \notin L\} = \Sigma^* - L$$

۲-٤-۳ عملگر اجتماع

$$L_1 \cup L_2 = \{\omega : \omega \in L_1 \vee \omega \in L_2\}$$

۲-٤-٤ عملگر اشتراک

$$L_1 \cap L_2 = \{\omega : \omega \in L_1 \wedge \omega \in L_2\}$$

۲-٤- ٥ عملگر تقسیم

$$\frac{L_1}{L_2} = \{x : \exists y \in L_2 \ni xy \in L_1\}$$

- مثال ۲–۱۸ با فرض L1=0*10* و L2=10*1، ؛ $L1/L2 = \varnothing$ زیرا هر $y \in L2$ دارای دو تا 1 است و هر $xy \in L1$ دارای یک 1 است بنابراین $\exists' y \in L2$

حال با فرض L3=0*1 خواهیم داشت: L1/L3=0* زیرا :

$$\exists y = 1 \in 0^*1 \ni x = 0^* \Rightarrow xy = 0^*1 \in L1$$

و $L2/L3=10^*$ زیرا:

$$\exists y = 1 \in 0^*1 \ni x = 10^* \Rightarrow xy = 10^*1 \in L1$$

بعداً ثابت خواهد شد که کلاس مجموعه های باقاعده تحت عمل تقسیم بسته است.

➢ **نکات**

- *کلاس Type-0 تحت همگی اعمال بجز مکمل بسته است .*
- *کلاس Type-2 تحت همگی اعمال بجز مکمل و اشتراک بسته است .*
- *کلاس Type-3 تحت همگی اعمال بسته است .*

❑ مثال ۲–۱۹ فرض کنید :

$$L1=\{0^m1^n2^p \mid m=n\}, L2=\{0^m1^n2^p \mid n=p\}, L3=\{0^m1^n2^p \mid m=n \quad \text{or} \quad n=p\}$$

L1 را میتوان از اتصال زبانهای مستقل از متن 2^* و $\{0^n1^n \mid n \geq 0\}$ به دست آورد پس L1 مستقل از متن است . زبان L2 مستقل از متن است و $L3 = L1 \cup L2$ پس L3 هم مستقل از متن است .

$L1 \cap L2 = \{0^m1^n2^p \mid m = n = p\}$ که مستقل از متن نیست پس این مثال نقض نشان میدهد که کلاس زبانهای مستقل از متن تحت اشتراک بسته نیست .

از جنبهٔ عملی در گرامر $G=(\Sigma,V,R,S)$ میتوان دو مساله را بررسی کرد :

۱– مسالهٔ عضویت[۱] به این معنی که با فر داشتن رشته ای روی Σ آیا این رشته متعق به LG) است یا خیر؟

۲– مسالهٔ تجزیه [۲] به این معنی که اگر رشته ای به زبان متعلق باشد چگونه میتوان آنرا از S بدست آورد؟

مسالهٔ عضویت در گرامرهای Type_0 غیرقابل تصمیم گیری [۳] است اما در گرامرهای وابسته به متن قابل تصمیم گیری است .در زبانهای مستقل از متن ؛ تعلق پذیری قابل تصمیم گیری است و زمان آن تابع چند جمله ای [۴] میباشد و بالاخره برای گرامرهای منظم این تابع خطی است.

❑ مثال ۲–۲۰ نشاندهید زبان Type_2 تحت مکمل بسته نیست .

برای اثبات از برهان خلف استفاده میکنیم اگر

$$L \in T2 \Rightarrow \overline{L} \in T2$$

$$if \qquad L1, L2 \in T2 \Rightarrow L\overline{1}, L\overline{2} \in T2$$

حال فرض میکنیم $L3 = L\overline{1} \cup L\overline{2}$ در اینصورت $L3 \in T2$ و داریم :

$L\overline{3} = \overline{(L\overline{1} \cup L\overline{2})} = L1 \cap L2 \notin T2$ زیرا در مثال ۲—۱۸ نشان دادیم زبان Type_2 تحت عمل اشتراک بسته نیست .

---

[۱] Membership Problem
[۲] Parsing Problem
[۳] Undecidable
[۴] Polynomial

۲-۵ درخت اشتقاق[۱]

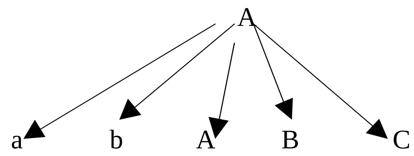
راه دوم نمایش اشتقاق صرف نظر از اینکه کدام قوانین مورد استفاده قرار می گیرند استفاده از درخت اشتقاق است. درخت اشتقاق یک درخت مرتب است که در آن گره ها با سمت چپ قوانین علامت گذاری می‌شوند و بچه های یک گره سمت راست آن قانون هستند و توسط ماشین براحتی قابل نمایش و استفاده می باشند.

۱ – ریشه درخت اشتقاق نماد آغاز گر است :S

۲ – هر گره داخلی درخت معرف یک متغییر است:V

۳ – گره های خارجی یا برگهای درخت معرف عناصر الفبا می باشند.

❑ مثال ۲-۲۱ : A→abABc



❑ مثال ۲-۲۲ :

$$\begin{cases} S \to ASB \\ S \to AB \\ A \to a \\ B \to b \end{cases}$$

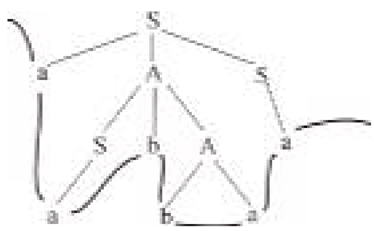
پیمایش برگهای درخت بدون توجه به سطح آنها از چپ به راست .یعنی پیمایش برگهای درخت از چپ براست و مستقل از سطح آنها جمله تولید می شود.

مثال: اشتقاق رشتهٔ w = aabbaa را با داشتن گرامر زیر بنویسید و درخت آن را رسم کنید.

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

$$R:\begin{vmatrix} S \to aAS \mid a \\ A \to SbA \mid SS \mid ba \end{vmatrix}$$

حل:



$$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$$

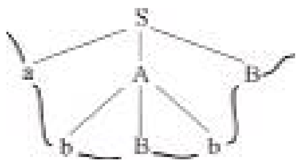
---

[۱] Derivation Tree

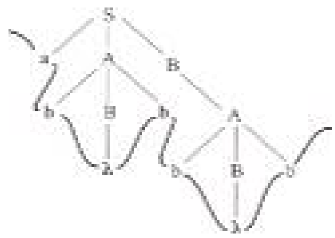
مثال ۲–۲۳: گرامر G با قوانین زیر مفروض است: درخت اشتقاق را برای شبه جملهٔ abBbB رسم کنید و سپس درخت اشتقاق کامل را برای جملهٔ aabbbb رسم کنید.

$$\begin{cases} S \rightarrow aAB \\ A \rightarrow bBb \\ B \rightarrow A \mid \lambda \end{cases}$$

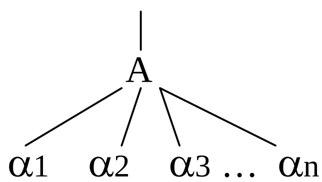
حل:



و درخت اشتقاق کامل برای جملهٔ aabbbb در زیر نشان داده شده است:



برای تولید درخت اشتقاق یک متغییر بسط داده نشده را انتخاب می کنیم (مثلاً A) و سپس یکی از قوانین $A\rightarrow \alpha1,\alpha2,...,\alpha n$ را انتخاب کرده، زیر درخت زیر تولید میشود:

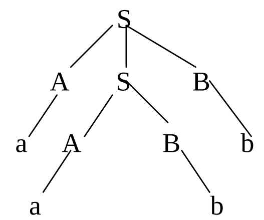


اگر همیشه سمت چپ ترین متغییر بسط داده نشده در درخت اشتقاق (مستقل از سطح) برای بسط انتخاب شود در اینصورت درخت اشتقاق درخت اشتقاق چپ[1] نام دارد.
حال اگر با اشتقاق راست[2] برویم ترتیب تولید عناصر جمله ها متفاوت خواهد شد ولی درخت اشتقاق آن کاملاً شبیه درخت اشتقاق چپ خواهد شد.

راست $S\rightarrow ASB\rightarrow ASb\rightarrow AABb\rightarrow AAbb\rightarrow Aabb\rightarrow aabb$

چپ $S\rightarrow ASB\rightarrow aSB\rightarrow aABB\rightarrow aaBB\rightarrow aabB\rightarrow aabb$

[1] Left Derivation
[2] Right Derivation



در ادامه بحث اشتقاق را با توجه به زبانهای مستقل از متن و با قاعده مطرح می کنیم.
سئوالی که در اینجا مطرح میشود اینست که آیا اشتقاق الگوریتمیک است ؟ در محدوده زبانهای مستقل از متن پاسخ مثبت است ولی در محدوده زبانهای بدون محدودیت سطح بالا مثل فارسی و لاتین پاسخ منفی است.
هدف تولید الگوریتمی برای انجام عمل اشتقاق در محدوده زبانهای مستقل از متن است.

در زبانهای مستقل از متن متغییرهای موجود در شبه جمله مستقل از هم بسط داده میشوند یا جایگزین می گردند و جایگزین ساختن یک متغییر در شبه جمله تاثیری بر سایر عناصر آن شبه جمله ندارد . یعنی متغییرهای یک شبه جمله باید حتما جایگزین شوند و ترتیب این جایگزینی اهمیتی در نتیجه نهایی نخواهد گذاشت.

❑ مثال ۲-۲٤ :

$$\begin{cases} S\rightarrow ABS \\ S\rightarrow m \\ A\rightarrow aA \\ B\rightarrow bB \\ A\rightarrow a \\ B\rightarrow b \end{cases}$$

$S\rightarrow ABS$ و هیچ تفاوتی ندارد که نخست کدام را جایگزین سازیم چون مستقل از متن است.

مثال: گرامر زیر مفروض است:

$$G: S \rightarrow (\ \ ) \mid (S) \mid SS$$

آیا جملهٔ w= ((( ))( )) از گرامر فوق ناشی شده است ؟

حل: برای پاسخ به این سوال باید رشتهٔ مزبور را از درخت اشتقاق بسازیم .

$$\begin{array}{c} S \\ \downarrow \\ (S) \\ \downarrow \\ (SS) \\ \downarrow \\ ((S)S) \\ \downarrow \\ (((\ ))S) \\ \downarrow \\ (((\ ))(\ )) \end{array}$$

۲-۶ ماشینی ساختن عمل اشتقاق

برای اینکار باید ماشین را در جهت انتخاب یک متغییر برای ادامه اشتقاق از بین متغییرهای دیگر شبه جمله هدایت کرد .

- روش انتخاب متغییر از شبه جمله چیست؟

با توجه به این نکته که انتخاب متغییرهای یک شبه جمله هیچ تاثیری در نتیجه کار ندارد میتوان همواره سمت چپ ترین متغییر در شبه جمله را برای ادامه اشتقاق یا بسط شبه جمله انتخاب کرد.

۲-۶-۱ روش‌های متداول انتخاب متغییر از شبه جمله

در **اشتقاق چپ**[1] همواره سمت چپ ترین متغییر در شبه جمله بسط داده می شود.

در **اشتقاق راست**[2] همواره سمت راست ترین متغییر در شبه جمله بسط داده می شود.

طرز نمایش

L.M.D=(شماره قانون‌های به‌کاررفته)

R.M.D=(شماره قانون‌های به‌کاررفته)

ممکن است برای تولید یک جمله چند اشتقاق چپ یا راست بسته به گرامر وجود داشته باشد. اما متداول‌ترین روش اشتقاق، اشتقاق چپ است. تمامی جمله های یک زبان توسط اشتقاق چپ قابل تولید هستند.

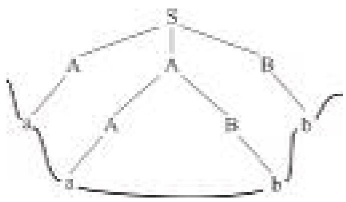
- مثال ۲-۲۵: گرامر زیر مفروض است. اشتقاق راست و چپ را برای جملهٔ aabb نشان دهید و سپس درخت اشتقاق را برای این جمله رسم کنید.

$$G:\begin{cases} S \to ASB \mid AB \\ A \to a \\ B \to b \end{cases}$$

حل:

راست $S \to ASB \to ASb \to AABb \to AAbb \to Aabb \to aabb$

چپ $S \to ASB \to aSB \to aABB \to aaBB \to aabB \to aabb$



[1] Left Most Derivation=L.M.D
[2] Right Most Derivation=R.M.D

با تعویض این‌دو و به‌همین ترتیب با تعیین اولین اشتقاق غیر چپ و عوض کردن آن با اولین اشتقاق چپ میتوان به اشتقاقی رسید که همگی چپ باشند← (1,3,1,4,2,3,4,3,4)

مثال ۲–۲٦ : گرامری با قوانین زیرمفروض است:

$$\begin{cases} S \to aAB \\ A \to bBb \\ B \to A|\lambda \end{cases}$$

یک اشتقاق چپ و یک اشتقاق راست برای جملهٔ abbbb بنویسید.

حل:

$$[L.M.D] = S \to aAB \to abBbB \to abAbB \to abbBbbB \to abbbbB \to abbbb$$
$$[R.M.D] = S \to aAB \to aA \to abBb \to abABb \to abABb \to abbBbbB \to abbbbB \to abbbb$$

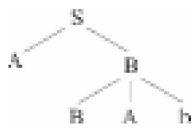
و در حالت کلی داریم:

$$\begin{cases} if \;\; S \to \omega \quad then \quad S \xrightarrow[L]{*} \omega \\ if \;\; S \to \omega \quad then \quad S \xrightarrow[R]{*} \omega \end{cases}$$

مثال: شکل زیر در بخشی از درخت اشتقاق یک گرامر به چشم میخورد . در این گرامر کدام قانون الزاماً وجود دارد؟

$a) S \to A \qquad b) S \to B \qquad c) S \to AB, S \to bAb \qquad d) a, b$

حل: آشکارا گزینه c درست می‌باشد.



۲–٦ ابهام[۱] در گرامر

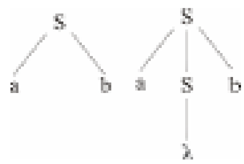
گرامر G مبهم است اگر برای تولید جمله $x \in L(G)$ لااقل دو درخت اشتقاق چپ متفاوت داشته باشیم. یعنی در گرامر غیر مبهم برای هر جمله فقط و فقط یک درخت اشتقاق وجود خواهد داشت.

مثال: ابهام در گرامر زیر را نشان دهید.

$$G: \begin{cases} S \to aSb \\ S \to \lambda \\ S \to ab \end{cases}$$

حل: برای جمله ab دو درخت اشتقاق چپ مختلف می‌توان نشان داد.

[۱] Ambiguity



درخت‌ها یکسان ولی جمله‌ها متفاوتند. یعنی جمله های تولید شده توسط دو درخت یکی هستند ولی معنای آنها متفاوت است! غیر مبهم بودن ویژگی مهمی است زییرا تفسیر یکتائی از هر جمله متعلق به زبان را ارائه میدهد.

- مثال ۲–۲۷ : ابهام در گرامر زیر را نشان دهید.

1) $\langle If\ -St\rangle \rightarrow If\ \langle Bexp\rangle\, then\ \langle St\rangle\, Else\ \langle St\rangle$

2) $\langle If\ -St\rangle \rightarrow If\ \langle Bexp\rangle\, then\ \langle St\rangle$

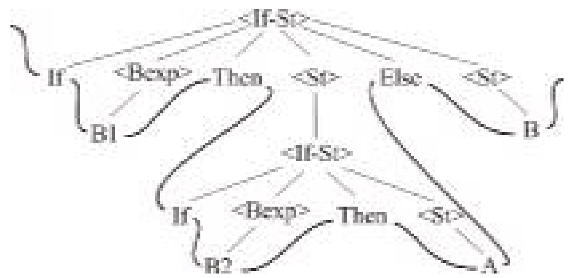
$\langle St\rangle \rightarrow \langle If - St\rangle$

$\langle St\rangle \rightarrow A|B|C$

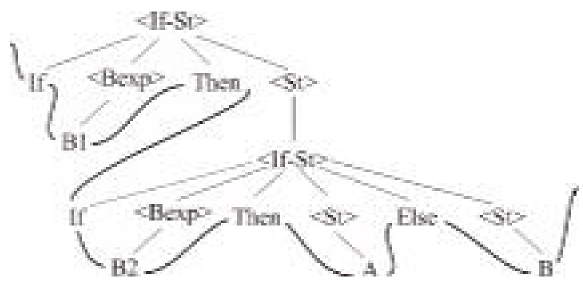
$\langle Bexp\rangle \rightarrow B1|B2$

حل: برای جملهٔ If B1 Then If B2 then A else B می‌توان دو درخت اشتقاق مختلف نمایش داد و این جمله از دید کمپایلر مبهم است زیرا مشخص نمی‌باشد Else دوم به کدام if تعلق دارد.

درخت اشتقاق اول:



درخت اشتقاق دوم:



در زبانهای با قاعده ناچار به اشتقاق از چپ هستیم گنگی یا ابهام یک خصیصه معمول در زبانهای طبیعی است که به روش های گوناگون با آن برخورد میشود. در زبانهای برنامه سازان چون باید برای هر جمله تنها یک متغییر

وجود داشته باشد ابهام را تا حد امکان باید رفع کنیم و اغلب اینکار با دوباره نویسی گرامر بـه یـک شـکل معـادل و غیر گنگ انجام میشود.

برخی از زبانها داتاً مبهم[1] هستند و مثالی از این دست در زیر به چشم میخورد:

- مثال ۲–۲۸　$\{0^m1^m2^n3^n \mid m,n>0\} \bigcup \{0^m1^n2^n3^m \mid m,n>0\}$

مثال: ثابت کنید گرامر زیر مبهم است .

$$G: E \to E+E \mid E*E \mid (E) \mid a \mid b$$

حل: برای جملهٔ $w=a*b+b$ میتوان دو درخت اشتقاق متفاوت نشان داد.

۲–۷　شبیه سازی عمل اشتقاق بر اساس اشتقاق چپ توسط ماشین

این الگوریتم باید در انتخاب قانون مورد استفاده تصمیم گیری کند . این الگوریتم گرافی را به نام گراف پارس تولید میکند .هر گره از این درخت یک شبه جمله است ، گره مبدا S (نماد آغازگر) است. هر لبه از گراف معرف قانونی است که شبه جمله را به شبه جمله دیگر تبدیل کرده است. گره های پایانی گراف جمله ها هستند . الگوریتم برای تولید جمله ω اقدام به تولید گراف می کند. این عمل آنقدر ادامه پیدا میکند که :

۱) ω بعنوان یک گره از گراف ظاهر شود . لذا ω متعلق به زبان گرامری است.$\omega \in L(G)$

۲) رشد گراف کاملا متوقف شود یعنی کلیه جمله های زبان گرامر تولید شوند . ولیکن ω ظاهر نشود در اینصورت $\omega \notin L(G)$

با توجه به هزینه تولید گراف و نامتناهی بودن زبانها باید مکانیزمی برای کنترل رشد گراف تعریف کرد . مکانیزم مورد استفاده بر اساس مفهوم پیشوندی[2] تعریف میشود.

---

[1] Inherently Ambiguous
[2] Prefix

۱) پیشوند جمله

$$\omega = \overbrace{\alpha_1\alpha_2\alpha_3}^{\beta_1}\ \overbrace{\alpha_4\ldots\alpha_n}^{\beta_2} \rightarrow \omega = \beta_1\beta_2$$

β_1 **پیشوند** $:\omega$

$$\omega = \underset{\beta_1}{}\ \underbrace{\alpha_1\alpha_2\alpha_3\ldots\alpha_n}_{\beta_2} \rightarrow \omega = \lambda\beta_2$$

$\lambda = \beta_1$ **پیشوند**ω **است**

$$n+1 \text{ پیشوند } \begin{cases} \beta_1=\lambda \\ \beta_1=\alpha_1 \\ \beta_1=\alpha_1\alpha_2 \\ . \\ . \\ \beta_1=\alpha_1\alpha_2\ldots\alpha_n \end{cases}$$

پس جمله می‌تواند n+1 پیشوند داشته باشد که n تعداد عناصر ان جمله است.

۲) پیشوند شبه جمله ای که جمله نیست :

$$\omega=\alpha_1 A\alpha_2 \Rightarrow \alpha_1 \in \Sigma^*$$

α_1 پیشوند ω است و A سمت چپ ترین متغیر در جمله

هر شبه جمله ای که جمله نباشد فقط و فقط یک پیشوند دارد.

- مکانیزم مورد استفاده بر اساس مفهوم پیشوند تعریف می شود .

" گره u از گراف پارس در صورتی توسعه می یابد که پیشوند uپیشوندی از ω باشد" برای طراحی الگوریتم از صف استفاده میشود.

۲-۸ تحلیل نحوی [۱]

در مرحلهٔ تحلیل نحوی، برنامه ورودی از نظر دستوری بررسی میشود. تحلیلگر نحوی یا پارسر برنامهٔ ورودی را که به شکل دنباله ای از توکن‌هاست، از اسکنر میگیرد و تعیین میکند که آیا این جمله میتواند بوسیلهٔ گرامر زبان مورد نظر تولید شود یا خیر؟ به‌طور کلی سه نوع روش تحلیل نحوی وجود دارد :

۱- روش عمومی [۲] که چندان کاراَ نیست ولی با هر نوع گرامری کار می‌کند.

۲- روش‌های بالا به پائین [۳]

۳- روش‌های پائین به بالا[۴]

روش‌های بالا به پائین درخت تجزیه [۵] را از بالا به پائین میسازند در حالی‌که روشهای پائین به بالا به‌وارونه عمل میکنند یعنی درخت تجزیه را از پائین به بالا تولید میکنند. هر دو روش ورودی را از چپ به راست و در هر قدم تنها یک توکن را بررسی میکنند.

[۱] Syntax analysis
[۲] Universal
[۳] Top-Down
[۴] Bottom_Up
[۵] Parse Tree

۲-۸-۱ تجزیهٔ بالا به پائین

در حالت کلی یک پارسر بالا به پائین باید بتواند در صورت لزوم عمل پی جوئی [6] را انجام دهد . یعنی پارسر در بررسی یک جمله نیاز پیدا میکند که یک مرحله به عقب برگردد و قاعده ای دیگر از گرامر را بررسی کند. تا رشتهٔ ورودی، برای گرامر قابل قبول باشد. عمل تحلیل نحوی را در حالت کلی میتوان با سعی و خطا انجام داد ولیکن بهتر است پارسرها بگونه ای طراحی و پیاده سازی شوند که نیازی به پی جوئی نداشته باشند. به اینگونه پاسرها که عمل پی جوئی را انجام نمیدهند، پارسر پیشگو[7] گفته میشود. برخی پارسرها به شکل حریصانه[8] عمل میکنند. یعنی با دریافت توکن درخت تجزیه را تا حد امکان گسترش میدهند و تنها هنگامیکه دیگر امکان گسترش درخت پارس وجود ندارد اقدام به دریافت توکن بعدی میکنند.

۲-۸-۱-۱ پارسر تولید کننده گراف پارس از بالا به پائین در پهنا

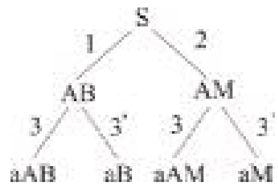
الگوریتم Pars1 گراف پارس را از بالا به پائین یعنی از S به سمت جمله مورد نظر شکل میدهد. پس الگوریتم تولید گراف پارس از بالا به پائین نامیده میشود. از طرف دیگر گراف پارس سطر به سطر شکل میگیرد بنابراین الگوریتم Pars1 تولید کننده گراف پارس در سطح از بالا به پائین است. در این الگوریتم قدیمی ترین گره در گراف را انتخاب می کنیم و شرط پیشوند را چک میکنیم و بسط می دهیم.

```
Procedure pars1(G=(∑,V,R,S),ω)
{ G  is  a  Context Free Grammer  }
 Queue  Q
  Creat(Q)
  Insert(S)        Places string at    of Queue
  Repeat
     DEQ(u)     Returns item at front of Queue & deletes it
     If u=α1Aα2 , α1∈∑*  Then
        If α1 Is a prefix of ω Then
           For every rule r:A→x Do
              Apply  r to u   produce α1xα2
                       Insert(α1xα2)
           End_For
        End_If
     End_If
  Until (u=ω Or Is_Empty(Q))
  If u=ω Then
    Return('Success')
  Else
    Return('Failuse')
  End_If
End{Pars1}
```

هنگام دستیابی به صف برای گرفتن یک شبه جمله اگر صف خالی باشد در اینصورت جمله متعلق به زبان نخواهد بود.

---

[6] Back -tracking
[7] Predictive
[8] Greedy

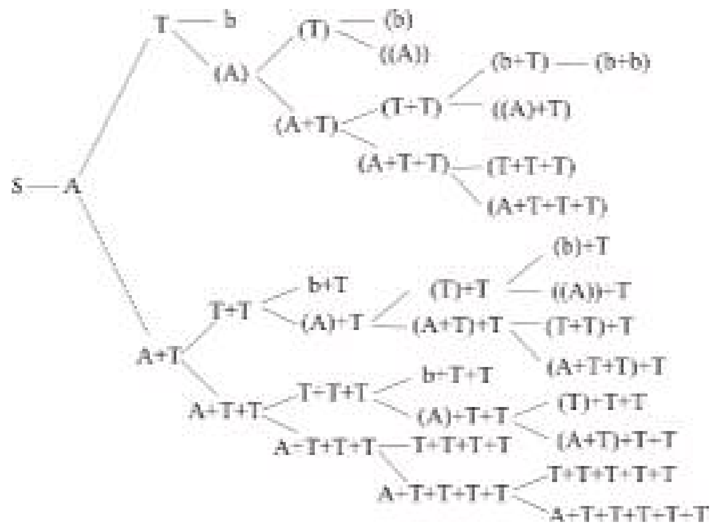


$$
\begin{array}{ll}
1: & S \to AB \\
2: & S \to AM \\
3,3': & A \to aA|a \\
4,4': & B \to bB|b \\
5: & M \to mM|m
\end{array}
$$

❏ مثال ۲–۲۸: فرض میکنیم زبان AE[1] شامل عبارت های ریاضی با یک متغییر b و اپراتور + و پرانتزباز و بسته باشد. رشته های تولید شده توسط AE عبارتند از b , (b), (b+b),(b)+b

AE: V={S,A,T}
∑={b, + , (,)}
R:
1) S→A
2) A→T
3) A→A+T
4) T→b
5) T→(A)

درخت جستجوی عبارت پارس (b+b) با استفاده از الگوریتم Pars1 در زیر نشان داده شده است.



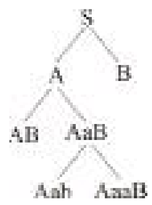
اشتقاق	ورودی خوانده شده توسط پارسر
$S \to A$	λ
$\to T$	λ
$\to (A)$	(
$\to (A+T)$	(
$\to (T+T)$	(
$\to (b+T)$	$(b+$
$\to (b+b)$	$(b+b)$

[1] Additive Expression

❑ مثال ۲–۲۹: گرامر زیر را در نظر بگیرید:

$$\begin{cases} S \to AB \\ A \to Aa \\ A \to a \\ B \to b \end{cases}$$

اگر فرض کنیم w=b باشد الگوریتم pars1 به انتها نمی‌رسد!



متوقف نمی شود!

این مثال نشان میدهند که الگوریتم Pars1 همیشه به انتها نخواهد رسید.بنابراین میتوان برای ایـن الگـوریتم حـالات زیر را بطور خلاصه در نظر گرفت:

pars1 → متوقف میشود → $\omega \in L(G)$ ، $\omega \notin L(G)$

pars1 → متوقف نمیشود → $\omega \notin L(G)$

مثال: الگوریتم پارس 1 را برای رشتهٔ $b+(b)$ با استفاده از گرامر قبل پیمایش کنید.

حل:

اشتقاق	ورودی خوانده شده توسط پارسر
$S \to A$	λ
$\to A+T$	λ
$\to T+T$	λ
$\to b+T$	$b+$
$\to b+(A)$	$b+($
$\to b+(T)$	$b+($
$\to b+(b)$	$b+(b)$

روش دیگر تولید گراف پارس از بالا به پائین تولید گراف در عمق و شاخه است. در این روش هم از شرط پیشوند برای کنترل رشد گراف استفاده میکنیم . برای طراحی الگوریتم از سازه پشته[1] استفاده می کنیم .الگوریتم تولید شده را Pars2 می نامیم که تولید کننده گراف در عمق و از بالا به پائین[2] است.

با استفاده از پشته می توان گراف مورد نظر را پیاده سازی کرد. در این حالت گراف به شکل عمق تشکیل و تکمیل می شود. در پشته در هر لحظه از push و یا pop یک زوج از عناصر یعنی شبه جمله و مجموعه قوانین که در شبه جمله قابل اعمال هستند و هنوز استفاده نشده اند ذخیره میشود.

اگر مجموعه قوانین زیاد باشند یکی از آنها را اعمال کرده بقیه را همراه با شبه جمله مربوطه در پشته ذخیره میکنیم . در تولید گراف پارس در عمق معمولا جدید ترین گره گراف بسط داده میشود اول جدیدترین گره انتخاب میشود ، شرط پیشوند بررسی و چک شده ، سپس بسط داده می شود.

Procedure pars2(G=(∑,V,R,S),ω)

Create(st) or push(st,[s,0]) *{st: Stack top*

{و 0 شماره قانونی است که به شبه جمله اعمال میشود تا گرهٔ بعدی در مسیر ایجاد گردد.

Push(st,[S,R]) *Push(st,[S,{r|r:S→k}])*

Repeat *{منجر به ساختن شاخه جدید میشود }*

Pop(st,[u,rules])
Flag=false
Repeat
If $u \notin \Sigma^*$ then let $u=\alpha_1 A \alpha_2$, $\alpha_1 \in \Sigma^*$
If α_1 Is a prefix of ω then
If there is not a r:A→x in Rules then
Flag=true else
Remove r from rules
If rules<>{} then
Push(st,[$\alpha_1 A \alpha_2$,rules])
End-if;
Apply r:A→x to u and generate
$u=\alpha_1 x \alpha_2$
rules=R

{مجموعه قوانین rule با کلیه قوانین مجددا set میشوند.}

Let $u=\alpha'_1 A' \alpha'_2$, rules={r|r:A'→x}
else flag=true
end-if;

{یعنی قانونی وجود نداشته باشد که سمت چپ ترین متغییر را بسط دهد }

else flag=true
end-if;
until ($u \in \Sigma^*$ Or flag)
until ($u=\omega$ or Is_empty(st))

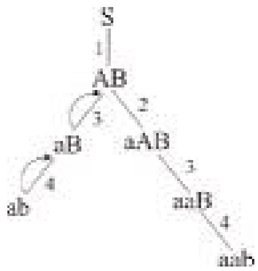
[1] Stack strcture
[2] Top to bottom

❑ مثال ۲-۳۰: گرامر زیر را در نظر بگیرید

$1: \quad S \to AB$
$2: \quad A \to aA$
$3: \quad A \to a$
$4: \quad B \to b$

با فرض آنکه (ω=aab) باشد پیمایش الگوریتم پارس ۲ چنین است:

	aab	*123*
	aaB	*124*
	aAB	*134*
Pop2	*ab*	*123*
Pop3	*aB*	*124*
	AB	*124*
	S	*234*
Pop1	*S*	*1234*



ستون اول بیانگر شبه جمله ها ست که نهایتاً به جمله ω=aab ختم میشود و ستون دوم نشاندهندهٔ شماره قوانین بکار نرفته (یا شمارهٔ قوانین به‌کار رفته) است .

- گراف پارس در پهنا همه جمله ها را تولید می کند ولی گراف در عمق تنها جمله های مورد نیاز را تولید میکند.
- در پارس ۲ هم بمانند پارس ۱ وجود حلقهٔ نامتناهی امکان دارد ؛ یعنی اگر در صورت تعلق و یا عدم تعلق این امکان وجود دارد که البته در الگوریتم پارس ۱ این امکان تنها در صورت عدم تعلق جمله به چشم میخورد .

❑ مثال ۲-۳۱: گرامر زیر را در نظر بگیرید

$$\begin{array}{|l} S \to AB \\ A \to Aa \\ A \to a \\ B \to b \end{array}$$

با فرض اینکه w=ab باشد پیمایش الگوریتم پارس ۲ برای این جمله منجر به حلقهٔ نامتناهی خواهد شد. بنابراین میتوان حالات زیر را برای پارس ۲ به شکل زیر خلاصه کرد:

pars2 → متوقف میشود → $\omega \in L(G)$ ، $\omega \notin L(G)$

pars2 → متوقف نمیشود → جوابی برای مساله وجود ندارد.

برای برطرف کردن مشکل قرار گرفتن در حلقهٔ نامتناهی دو راه حل به نظر میرسد یکی اینکه الگوریتم پارسر را عوض کنیم و دیگری اینکه در قوانین گرامر اصلاح و تجدید نظر کنیم .

مثال: $((b))$را با الگوریتم پارس ۲ با گرامر زیر پیمایش کنید.

حل:

اشتقاق	پشته
$S \to A$	$[S,1]$
$\to T$	$[A,2]$
$\to (A)$	$[T,5]$
$\to (T)$	$[(A),2]$
$\to ((A))$	$[(T),5]$
$\to ((T))$	$[((A),2)]$
$\to ((b))$	$[((T),4)]$

۲-۸-۲ تجزیه از پائین به بالا

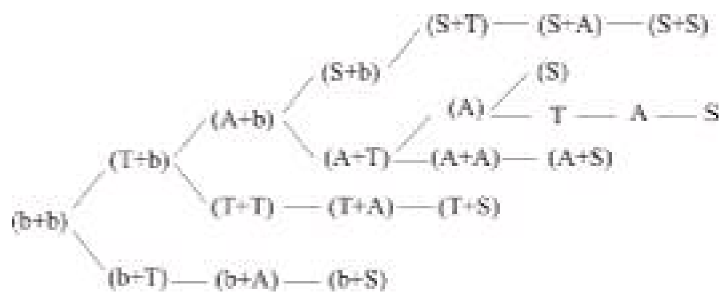
پارسرهای ۱و۲ تولید کنندهٔ گراف از نماد آغاز گر به جمله $(S \overset{*}{\Rightarrow} \omega)$ هستند و در پارسرهائی که در زیر درباره آنها در زیر بحث خواهد شد دیدگاه کمی تغییر پیدا میکند ؛ یعنی فرض براینست که از جمله بخواهیم به‌طور معکوس به نماد آغازگر برسیم.

```
procedure pars3(G=(∑,V,R,S),ω)
     create(q)
        Insert(q,ω)
        Repeat
             Deq(Q,u)
             If  u<>S then
                   For Every reduceable substring α of u do
                        Let u=xαy
                      For every rule r:A→α do
                             Apply  r  to  u  and generate xAy
                             Insert(Q,xAy)
                        End-for
                   End-for
             End-if
      Until (u=s Or Is_empty(Q)
      If u=s then return('success')
      Else return('failure')
```

در پارسرهای تولید کننده گراف پارس از پائین به بالا، قدیمی ترین گره در گراف انتخاب میشود وبه هر زیر رشته قابل کاهش و هر قانونی که میتواند این زیر رشته را کاهش دهد یک گره جدید به وجود می آورد . این نوع پارسرها فقط برای اشتقاق از راست طراحی شده اند و از پائین به بالا[1] هستند .

- مثال ۲-۳۲ پیمایش جملهٔ w=(b+b) را با پارسر نوع ۳ با بکارگیری قوانین مثال ۲-۲۵ نشان میدهد :

$$\begin{aligned} S &\Rightarrow A \\ &\Rightarrow T \\ &\Rightarrow (A) \\ &\Rightarrow (A+T) \\ &\Rightarrow (A+b) \\ &\Rightarrow (T+b) \\ &\Rightarrow (b+b) \end{aligned}$$



- تذکر : هنگامیکه در گرامر جمله S→λ وجود داشته باشد این روال[1] در حلقه نا متناهی قرار میگیرد و اگر متغییربه متغیر دیگری نسبت داده شود وبعد بخودش باز گردد ؛امکان حلقهٔ نامتناهی وجود خواهد داشت

مثال۲-۳۳ گرامر زیر را در نظر بگیرید: برای رشته $\omega = b$ با الگوریتم پارس 3 گرامر را پیمایش کنید.

$$G:\left| \begin{array}{l} S \to AB \\ S \to \lambda \\ A \to aA \\ A \to a \\ B \to b \end{array} \right.$$

حل: b دو زیر رشته قابل کاهش دارد و از آنجائیکه $b \notin L(G)$ بنابراین الگوریتم پارس 3 در حلقهٔ نامتناهی قرار میگیرد.

- نکته :در pars3 هنگامیکه جمله ورودی متعلق به زبان نباشد و S→λ در R موجود باشد قطعاً در حلقه نامتناهی قرار میگیرد.بنابراین راه حل این است که به گونه ای بتوان قانون S→λ را تنها و تنها به تولید جمله λ منحصر کرد. در این صورت از وجود حلقه نامتناهی می توان پرهیز کرد.

[1] Bottom -up

❑ مثال۲–۳٤ در این جا قانون $S\rightarrow\lambda$ ؛ تنها و تنها به تولید جمله λ منحصر شده است .

$$\begin{cases} S\rightarrow aSb \\ S\rightarrow\lambda \\ S\rightarrow ab \end{cases}$$

یک روش دیگر تجزیه از بالا به پائین به روش انتقال – کاهش[۲]موسوم است . در این‌روش عکس تجزیهٔ بالا به پائین انجام میشود. به این ترتیب که از رشتهٔ ورودی شروع میکند و ساخت درخت تجزیه از برگها آغاز شده و به سوی ریشه پیش میرود. در الگوریتم pars4 گراف پارس در عمق تشکیل میشود و برای ساخت آن از پشته استفاده میکنیم . در هر مرحله از ذخیره و یا بازیابی سه تایی [u,v,rules] استفاده میشود . uبخشی از جمله ورودی کاهش یافته است .$u\in\Sigma^*$ بقیه ورودی میباشد که باید کاهش یابد و rules مجموعه قوانینی است که برای کاهش حتمالی پسوندهای u میتواند مورد استفاده قرار بگیرد . α_2 برای شبه جمله $u=\alpha_1\alpha_2$ پسوند (suffix) میباشد. هر گره از گراف به شکل u,v,rules است.اگر پسوندی از uتوسط حداقل یک قانون از rules قابل کاهش باشد این قانون را باید از rules حذف کرد و سه تایی u,v,rules را جدید را به پشته افزود . u تحت تاثیر این قانون انتخابی کاهش می یابد و تبدیل به u' میشود و گره جدید به شکل u′,v,rules تولید می گردد.

♦ اگر پسوندی از u بر اساس rules قابل کاهش نباشد:

۱ – اگر $x<>\lambda$ سمت چپ ترین عنصر v از آن جدا و به سمت راست ترین عضو u متصل میشود و rules=r خواهد شد . یعنی اگر $v=av'$ شبه جمله میشود ua,v′,r این عمل را shift گویند . بهمین دلیل این پارسر را گاهی اوقات Shift-Reduce-parser میگویند.

۲ – اگر $v=\lambda$ در اینصورت اگر u=s باشد جمله ورودی به زبان متعلق است و گرنه رشد گراف در این نقطه متوقف میشود و از پشته سه تایی جدید بازیابی میگردد. چنانچه پشته خالی باشد جمله متغلق به زبان نمیباشد.

Procedure pars4(G=(∑,V,R,S), ω)

```
    Create(ω)
    Push(st,[λ,ω,R])
    Repeat
        Pop(st,[u,v,rules])
        Flag=false
        Repeat
            If a suffix of u can be reduced using rules  then
                Select r from rules r:A→x
                Let u=αx
                Push(st,[u,v=rules-{r}])
                Reduce u using r and generate u=αA
                Rules=R
            Else
                If v=λ then flag=true
                Else
                    Let v=av′
                    u=ua
                    v=v′
                    rules=R
                end-if
```

[۲] Shift - Reduce

```
                    end-if
                Until(flag)
        Until((u=s & flag) or is_empty(st))
        If u=s & flag then return('success')
        Else return('failuse')
        End-if
End{pars4}
```

- این الگوریتم چنانچه جمله به زبان متعلق باشد یا نباشد ($S\to\lambda\in R$)در حلقه نامتناهی قرار میگیرد.در تمامی این الگوریتم ها فرض میشود گرامر مستقل از متن است یعنی سمبل آغازین در طرف راست هیچ قاعده ای قرار نمی گیرد.

- پسوند : برخلاف پیشوند شبه جمله است و در هر شبه جمله ای وجود دارد.

$u\to\alpha_0\alpha_1\alpha_2\ldots\alpha_n$

پسوندها

λ
α_n
$\alpha_{n-1}\alpha_n$
.
.
U

- مثال ۲–۳۵ : گرامر زیر را در نظر بگیرید :

1- $S\to AB$
2- $A\to aA$
3- $A\to a$
4- $B\to b$

	S	λ	
	AB	λ	[2,3,4]
	Ab	λ	[1,2,3]
	AA	b	[1,3,4]
	Aa	b	[1,2,4]
Pop4	Aab	λ	[1,2]
Pop1	AAB	λ	[2,3,4]
Pop2	AAb	λ	[1,2,3]
Pop3	Aa	b	[1,2,4]
	A	ab	[1,2,4]

مثال ۲–۳۶ فرض کنیم با الگوریتم پارس ٤ و با بکارگیری قوانین مثال ۲–۲۵ بخواهیم اشتقاق (b+b) را بسازیم

[S,0]		[S,1]				
	[A,2]					
	[T,4]		[T,5]			
	B		[(A),2]			[(A),3]
			[(T),4]		[(T),5]	[(A+T),2]
						[(T+T),4]
			(b)	((A))		[(b+T),4]
						(b+b)

پشتـــــــه	اشتقـــاق
[S,1]	$S \Rightarrow A$
[A,2]	$\Rightarrow T$
[T,5]	$\Rightarrow (A)$
[(A),3]	$\Rightarrow (A+T)$
[(A+T),2]	$\Rightarrow (T+T)$
[(T+T),4]	$\Rightarrow (b+T)$
[(b+T),4]	$\Rightarrow (b+b)$

مثال ۲–۳۷ : اشتقاق b+(b) را با الگوریتم اخیر بسازید.

حل: مراحل کار در زیر نشان داده شده است.

u	v	عمل
λ	$(b)+b$	*Pop*
$($	$b)+b$	*Shift*
$(b$	$)+b$	*Shift*
$(T$	$)+b$	*Reduction*
$(A$	$)+b$	*Reduction*
(A)	$+b$	*Shift*
T	$+b$	*Reduction*
A	$+b$	*Reduction*
$A+$	b	*Shift*
$A+b$	λ	*Shift*
$A+T$	λ	*Reduction*
T	λ	*Reduction*
A	λ	*Reduction*
S	λ	*Reduction*

❑ مثال ۲–۳۸ گرامر زیر را در نظر بگیرید

AE:
$V=\{S, A, T\}$
$\Sigma= \{b, +, (,)\}$
R :{
1- $S \rightarrow A$,
2- $A \rightarrow T$,
3- $A \rightarrow A+T$,
4- $T \rightarrow b$,
5- $T \rightarrow (A)$}

الگوریتم pars4 را برای اشتقاق رشته (b+b) را پیمایش کنید.

پشته	u	v	عمل
[λ,∅,(b+b)]	λ	(b+b)	Pop
	(	b+b)	Shift
	(b	+b)	Shift
[(b,4,+b)]	(T	+b)	Reduction
[(T,2,+b)] [(b,4,+b)]	(A	+b)	Reduction
[(T,2,+b)] [(b,4,+b)]	(A+	b)	Shift
[(T,2,+b)] [(b,4,+b)]	(a+b	)	Shift
[(A+B,4,)] [(T,2,+b)] [(b,4,+b)]	(A+T	)	Reduction
[(A+T,2,)] [(A+b,4,)] [(T,2,+b)] [(b,4,+b)]	(A+A	)	Reduction
[(A+T,2,)] [(A+b,4,)] [(T,2,+b)] [(b,4,+b)]	(A+A)	λ	Shift
[(A+b,4,)] [(T,2,+b)] [(b,4,+b)]	(A+T	)	Pop
[(A+T,3,)] [(A+b,4,)] [(T,2,+b)] [(b,4,+b)]	(A	)	Reduction
	(A)	λ	Shift
[(A),5,λ] [(A+T,3,)] [(A+b,4,)] [(T,2,+b)] [(b,4,+b)]	T	λ	Reduction
[T,2,λ] [(a),S,λ] . . [(b,4,+b)]	A	λ	Reduction
[A,1,λ] [T,2,λ],…	S	λ	Reduction

۲-۹ زبان‌های غیر مستقل از متن

زبانهائی هستند که نمیتوان آنها را توسط یک گرامر مستقل از متن توصیف کرد. همچنین محـدودیتهائی در زبانهـای برنامه نویسی وجود دارد که نمیتوان آنها را توسط گرامرهای مستقل از متن اعمال کرد.

❑ مثال ۲-۳۹ : زبان زیر را در نظر بگیرید :

$$L = \{\omega c \omega \mid \omega \quad is \quad in \quad (a \mid b)^*\}$$

زبان L مستقل از متن نیست .این زبان رشته هائی به صورت aabcaab تولید میکند . ایـن رشـته هـا را مـی‌تـوان مشابهٔ این محدودیت در زبانهای برنامه سازی فرض کرد که تعریف متغیرها باید قبل از استفاده از آنها باشد، به ایـن ترتیب که w اول در رشتهٔ wcw بیانگر تعریف متغیر است و w دوم نشاندهندهٔ استفاده از متغیر .

```
declaration        abc
   -begin           w
      .
c     .
      .
   -end

abc=...
 w
```

❑ مثال ۲-۴۰ : زبان $L = \{a^n b^m c^n d^m \mid m \geq 1 \wedge n \geq 1\}$ مستقل از متن نیست . این زبان رشته هـائی بـه شکل $a^+b^+c^+d^+$ ایجاد میکند که در آنها تعداد تکرار a با c برابر است و تعداد تکرار b بـا d یکـی اسـت . این زبان مشابهٔ این محدودیت در زبانها برنامه سازی است که تعداد پارامترهـا در تعریـف یـک روال بایـستی بـا تعداد آرگومانها در فراخوانی آن روال برابر باشد .

```
declaration   proc1(a,a,a)
declaration   proc2(b,b)
.
.
.
.
call   proc1(c,c,c)
call   proc2(d,d)
```

❑ مثال ۲-۴۱ : زبان $L = \{\omega c \omega^R \mid \omega \quad is \quad in \quad (a \mid b)^*\}$ مستقل از متن است و با گرامر زیـر قابل توصیف میباشد:

$$S \rightarrow aSa \mid bSb \mid c$$

- مثال ۲-۴۲ : زبان $L=\{a^nb^nc^md^m \mid m\ge 1 \wedge n\ge 1\}$ مستقل از متن است و با گرامر زیر توصیف میشود:

$$\begin{cases} S\to AB \\ A\to aAb \mid ab \\ B\to cBd \mid cd \end{cases}$$

- مثال ۲-۴۳ : زبان $L=\{a^nb^n \mid n\ge 1\}$ نیز مستقل از متن است و گرامر نشاندهندهٔ آن عبارتست از :

$$S\to aSb \mid ab$$

۲-۱۰ الگوریتمی برای تعیین منظم بودن زبان

در این بخش میخواهیم از نتیجهٔ مهمی استفاده کنیم که لم تزریق یا پمپاژ [1] نام دارد . لم تزریق یکی از قویترین ابزار در تعیین ویژگی منظم بودن زبان است . و همچنین در تکامل الگوریتمهائی وابسته به اتومتای متناهی[2] کاربرد دارد . مانند اینکه آیا زبان توسط اتومتا پذیرفته میشود یا خیر؟ زیرا اگر زبانی منظم باشد توسط اتومتای متناهی پذیرفته میشود. قضیهٔ لم تزریق به قرار زیر است:

فرض کنید L مجموعهٔ منظم باشد . ثابت n ای وجود دارد که اگر z در هر کلمه ای از L باشد($z\in L$) و |z|>=n ما میتوانیم z را به شکل z=uvw بنویسیم بطوریکه |uv|<=n و |v|>=1 باشد. بنابراین $\forall i\ge 0 \quad uv^iw\in L$ (به این معنی که اگر رشتهٔ میانی هر چند بار تکرار شود سرانجام به حالت نهائی خواهیم رسید) بعلاوه n نباید از تعداد حالات کوچکترین اتومتای متناهی پذیرندهٔ L بیشتر باشد[3].

کاربرد لم تزریق در اثبات این است که مجموعه های مشخصی باقاعده هستند یا خیر؟ و برای بکار بردن آن باید به نکات زیر توجه داشت :

۱- گزینش و تعیین زبان L که میخواهیم ثابت کنیم منظم است یا خیر ؟
۲- گزینش n که باید در تکرار بدست آید .
۳- گزینش رشتهٔ $z\in L$ که صریحاً وابسته به مقدار انتخابی n در گام ۲ دارد .
۴- شکستن z به u و v و w بطوریکه : |uv|<=n و |v|>=1

لم تزریق را میتوان با گزاره های ریاضی به شکل زیر بیان کرد :

$$(\forall L)(\exists n)(\forall z)[z\in L\& \mid z\mid\ge n \Rightarrow (\exists u,v,w)(z=uvw,\mid uv\mid\le n,\mid v\mid\ge 1$$
$$\&(\forall i)(uv^iw\in L)]$$

- لم تزریق یک شرط لازم برای منظم بودن زبان است و نه شرط کافی پس ما فقط میتوانیم از این لم برای منظم نبودن زبان استفاده کنیم .

---

[1] Pumping Lemma

[2] Finite Automata در بخش دوم بررسی خواهد شد.

❑ مثال ۲-٤٤ : $L = \{0^{i^2} \mid i \geq 1\}$ این زبان شامل همهٔ رشته هائی از 0 بطول مربع کامل است . در اینجا نشان میدهیم که L منظم نیست :

حل: برای اینکار از برهان خلف استفاده میکنیم ؛ فرض کنید L منظم باشد بر طبق قضیهٔ تزریق اگر $z = 0^{n^2}$ باید :

$$0^{n^2} = uvw \ni 1 \leq |v| \leq n, \forall i \quad uv^i w \in L$$

اگر فرض کنیم I=2 باشد :

$$n^2 \leq |uv^2 w| \leq n^2 + n < (n+1)^2$$

یعنی uv^2w مربع کامل نیست ؛ بنابراین :

$$uv^2 w \notin L$$

❑ مثال ۲-٤٥ : نشاندهید زبان $L = \{a^i b^i \mid i \in \mathrm{IN}\}$ منظم نیست.

حل: اگر L منظم باشد $\exists n$ بقسمیکه شرایط لم تزریق برقرار باشد . فرض میکنیم $z = a^n b^n \in L$ داریم :

|z|=2n>n اگر z=uvw و $v \neq \lambda$ و I=2 در اینصورت $z = uv^2w$

$$N_a(uv^2w) = N_a(uvw) + N_a(v) = N_a(z) + N_a(v) = n + N_a(v) > n$$

که به تناقص رسیدیم یعنی شرط برقرار نمیباشد .[منظور از نماد $N_a(X)$ تعداد کاراکتر a دررشتهٔ X میباشد.]

مسائل فصل دوم

۲–۱ عبارت با قاعدهٔ زبان زیر را بنویسید.

$L1=\{\omega \mid \omega\in\{a,b\}^*,$ ω به b ختم میشود$\}$

۲–۲ عبارت با قاعدهٔ زبان زیر را بنویسید .

$L1=\{\omega \mid \omega\in\{a,b\}^*,$ ω شامل دو زیر رشتهٔ bb است$\}$

۲–۳ گرامر زبان زیر را بنویسید

$$L=\{o^m 1^n \mid m>n\geq 0\}$$

۲–٤ گرامر و عبارت منظمی بنوی‌سی‌د که زبان $L=\{0^m 1^n \mid$ m+n فرد است$\}$ را توصی‌ف کند.

۲–٥ گرامر زبان زیر را بنویسید .

$$L=\{0^k 1^m 0^n \mid n=k+m\}$$

۲–٦ $L1=\{\omega \mid \omega\in\{a,b\}^*,$ w فقط و فقط یک a دارد . $\}$

۲–۷ $L1=\{\omega \mid \omega\in\{a,b\}^*,$ w حداقل یک a دارد. $\}$

۲–۸ $L1=\{\omega \mid \omega\in\{a,b\}^*,$ w حداکثر سه a دارد. $\}$

۲–۹ گرامر زبان L=(10)* را بنویسید .

۲–۱۰ گرامر زبان L=ab*+c* را بنویسید.

۲–۱۱ برای زبان مستقل از متن زیر گرامر بنویسید.

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

۲–۱۲　　گرامر زبان زیر را بنویسید.

$$L = \{\omega \in (0+1)^* \mid N_0(\omega) = N_1(\omega)\}$$

۲–۱۳　　گرامری بنویسید که زبان زیررا توصیف کند .

۲–۱٤　　زبانی که گرامر زیر را تولید میکند بنویسید.

$$G:\begin{cases} S \to abB \\ A \to aaBb \\ B \to bbAa \\ A \to \lambda \end{cases}$$

و سپس درخت اشتقاق را برای دو جملهٔ abbba و abbbaabbaba رسم کنید .

۲–۱۵　　زبان گرامر زیر را بدست آورید.و درخت اشتقاق را برای جمله های abc و aabbcc رسم کنید .

$$\begin{aligned} &\Sigma = \{a,b,c\} \\ &V = \{A,B,C\} \\ &S = A \\ &R:\left|\begin{array}{l} A \to aABC \mid abc \\ CB \to BC \\ bB \to bb \\ bC \to bc \\ cC \to cc \end{array}\right. \end{aligned}$$

۲–۱٦　　زبان گرامر زیر را بنویسید.

$$\begin{aligned} &\Sigma = \{a,b\} \\ &V = \{S\} \\ &G: S \to aSb \mid ab \end{aligned}$$

۲–۱۷　زبان گرامر زیر را بنویسید.

$$\begin{aligned} &\Sigma = \{a,b\} \\ &V = \{S\} \\ &G: S \to aSb \mid \lambda \end{aligned}$$

۱۸–۲ گرامر زیر مفروض است :

$$G : S \to (\ \ ) \mid (S) \mid SS$$

آیا جملهٔ w= ((( )) ( ) ((از گرامر فوق ناشی شده است ؟

۱۹–۲ ثابت کنید عبارت های باقاعدهٔ r1=(aa)*(bb)*b و r2=(aa)*b(bb)* همانند هستند .

۲۰–۲ الگوریتمی برای برابری دو زبان L1 و L2 ارائه دهید.

۲۱–۲ گرامر زیان زیر را بنویسید :

$$L = \{a^m b^n c^m d^n \mid m, n \geq 1\}$$

۲۲–۲ گرامر زیانهای زیر را بنویسید:

$$L1 = \{a^n b^n c^k \mid n, k \geq 1\}$$
$$L2 = \{a^k b^n c^n \mid n, k \geq 1\}$$

۲۳–۲ با فرض داشتن گرامر زیر اشتقاق p=baaaabab را بنویسید.

$$\begin{cases} S \to RT \\ T \to aTB \\ TB \to BZ \\ aB \to Ba \\ RB \to bZ \\ Za \to aZ \\ Zb \to bZ \\ ZB \to ab \end{cases}$$

$$\begin{aligned} S &\Rightarrow RT \Rightarrow RaTB \Rightarrow RaaTBB \\ &\Rightarrow RaaaTBBB \Rightarrow RaaaBZBB \\ &\Rightarrow RaaBaZBB \Rightarrow RaBaaZBB \\ &\Rightarrow RBaaaZBB \Rightarrow bZaaaZBB \\ &\Rightarrow bZaaaabB \Rightarrow baZaaabB \\ &\Rightarrow baaZaabbB \Rightarrow baaaZabB \\ &\Rightarrow baaaaZbB \Rightarrow baaaabZB \\ &\Rightarrow baaaabab \end{aligned}$$

۲–۲٤ گرامری بنویسید که زبان زیر را تولید کند :

L=(a*b)*(b*a|ba*)d

۲–۲٥ با فرض داشتن گرامر زیر زبان آنرا بنویسید.

$$G=(\sum,V,R,S)$$
$$\sum=\{a,b\}$$
$$V=\{S,A,B\}$$
$$R:\begin{cases}S\to aB\mid bA\\ A\to a\mid aS\mid bAA\\ B\to aBB\mid bS\mid b\end{cases}$$

۲–۲٦ اشتقاق رشتهٔ w=aabbaa را با داشتن گرامر زیر بنویسید و درخت آن را رسم کنید .

$$G=(\{a,b\},\{S,A\},R,S\}$$
$$R:\left|\begin{array}{l}S\to aAS\mid a\\ A\to SbA\mid SS\mid ba\end{array}\right.$$

۲–۲۷ زبان گرامر زیر را بنویسید .

$$S\to aSa\mid bSb\mid \lambda$$

۲–۲۸ زبان گرامر زیر را بنویسید .

$$G:S\to aSb\mid \lambda$$

۲–۲۹ زبان گرامر زیر را بنویسید.

$$G:S\to aSb\mid \lambda$$

۲–۳۰ گرامر زبان زیر را بنویسید .

$$L=\{\lambda,aa,abba,abbbba,...\}$$

۲–۳۱ گرامر زبان زیر را بنویسید.

$$L=\{a^ib^i\mid i\ge 1\}$$

۲–۳۲ زبان گرامر زیر را بنویسید .

$$G = (\{a,b,c\},\{S,X,Y\},R,S\}$$

$$G:\begin{cases} S \to abc \mid aXbc \\ Xb \to bX \\ Xc \to Ybcc \\ bY \to Yb \\ aY \to aaX \mid aa \end{cases}$$

۲–۳۳ گرامری بنویسید که زبان زیر را تولید کند .

$$L = \{a^i b^j \mid i, j = 0,1,2,... \;\&\; j \ge i\}$$

۲–۳۴ گرامری بنویسید که زبان زیر را تولید کند .

$$L = \{a^i b^j a^j b^i \mid i, j = 0,1,2,...\}$$

۲–۳۵ گرامری بنویسید که زبان زیر را تولید کند .

$$L = \{a^i b^i a^j b^j \mid i, j = 0,1,2,...\}$$

۲–۳۶ گرامری بنویسید که زبان زیر را تولید کند .

$$L = \{a^i b^i \mid i, j = 0,1,2,...\} \bigcup \{b^j a^j \mid j = 0,1,2,...\}$$

۲–۳۷ گرامر زبان زیر رابنویسید.

$$L = \{a^i b^j c^{i+j} \mid i, j = 0,1,2,...\}$$

فصل سوم: ساده سازی گرامرهای مستقل از متن و فرم‌های نرمال

مقدمه

تعریف یک گرامر مستقل از متن هیچ محدودیتی بر روی سمت راست قانون اعمال نمیکند. دیدیم برای اجتناب از حلقه های نامتناهی و حل برخی مشکلات لازم است بر روی گرامر ها محدودیتهایی اعمال کنیم و قوانینی مثلا به شکل $A\to\lambda$ یا $A\to B$ را حذف کنیم تا کار با گرامر آسانتر شود. در برخی جاها حتی لازم است محدودیتهای بیشتری هم بر روی گرامر اعمال شودبهمین دلیل لازم است که به روشهایی برای تبدیل گرامرهای مستقل از متن به گرامرهای معادلی که محدودیتهای خاصی را رعایت میکند دست یابیم. همچنین به فرمهای نرمال و گرامرهای مستقل از متن خواهیم پرداخت گرچه محدودیتهای اعمال شده آنقدر گستردگی دارد که بتوان یک معادل با فرم نرمال ساخت . معمولاً فرمهای نرمال چامسکی[1] (CNF) و گریباخ[2] (GNF) کاربردهای نظری و عملی بسیاری دارند که در این بخش معرفی خواهند شد.

۳-۱ روشهای تبدیل گرامر ها

ابتدا به مساله رشتهٔ تهی ؛ که مشکلاتی را پدید می آورد؛ می پردازیم. در اینجا ترجیح میدهیم که رشته تهی را کاملاً حذف کنیم وبه زبانهایی بپردازیم که λ در آنها نباشد .

گرامر های نرمال گرامرهای مستقل از متنی هستند که سعی میشود در آنها هیچکدام از قوانینی که منجر به تولید حلقه نامتناهی میشود وجود نداشته باشد.

در فرم نرمال گریباخ ؛ قوانین گرامر مستقل ازمتن میتوانند به شکل زیر باشند :

1- $A\to a$

2- $A\to aA_1A_2\ldots A_n$

$A_i\in V-\{S\}\quad 1\le i\le n$

3- $S\to\lambda\in R$ If $\lambda\in L(G)$

یعنی در اینجا نماد آغاز گر فقط و فقط در سمت چپ قوانین ظاهر میشود . مثلا $S\to aSb$ فرم نرمال گریباخ نیست هنگامی که از فرم نرمال گریباخ استفاده میکنیم شرط پایانی اینست که سطح گراف حداکثر برابر طول جمله باشد.

- در گرامر نرمال گریباخ تعداد مراحل اشتقاق (عمق درخت اشتقاق و گراف پارس) برای تولید جمله ای به طول n برابر m یا طول جمله میباشد.

---

[1] Chomsky Normal Form

[2] Greibach Normal Form

۳-۱-۱ مراحل تبدیل گرامر مستقل از متن $G=(\Sigma,V,R,S)$ به نوع گریباخ $G'=(\Sigma,V',R',S')$: (ترتیب مهم است)

۱- *حذف نماد آغازگر بازگشتی*
۲- *حذف قوانین λ*
۳- *حذف قوانین به شکل $A\rightarrow B$ (زنجیره[1])*
٤- *حذف متغییرهای غیرمفید[2] و قوانین وابسته*
٥- *حذف بازگشت چپ*
٦- *تبدیل نهایی به گریباخ*

➢ **عوامل حلقه های نامتناهی به طور کل عبارتند از :**

۱- $S\rightarrow\lambda$
۲- وجود حلقه $A\rightarrow B$, $B\rightarrow A$
۳- *بازگشت ازچپ* $A\rightarrow A\alpha$

۳-۱-۲ متغیربازگشتی [1]

تبدیل گرامر با اعمال محدودیت سمبل آغازین گرامر شروع میشود . گرامر G را در نظر بگیرید. در گرامر معادل G' نقش سمبل آغازین به مقداردهی اولیه اشتقاق محدود میشود . فرم اشتقاق بازگشتی به صورت $S\Rightarrow uSv$ سبب میشود کهسمبل آغازین در شبه جمله های گامهای بعدی اشتقاق ظاهر شود و این محدودیت زمانی برقرار است که سمبل آغازین G' یک متغییر غیر بازگشتی باشد.

➢ **لِم ۳-۱-** فرض کنید $G=(\Sigma,V,R,S)$ یک گرامر مستقل از متن باشد آنگاه گرامر $\exists\, G'=(\Sigma',V',R',S')$ بقسمیکه شرایط زیر برقرار باشد:

1- $L(G)=L(G')$
2- *قوانین R' به شکل زیر هستند*

$A\rightarrow\omega$
$A\in V'\ \&\ \omega\in((V-\{S'\})\cup\Sigma)^*$ *بقسمیکه*

اثبات : اگر سمبل آغازین S در سمت راست قوانین G واقع نباشد پس $G'=G$ واگر S یک متغیر غیر بازگشتی باشد خاصیت بازگشتی سمبل آغازین را باید حذف کرد . گرامر $G'=(\Sigma,V\cup\{S'\},R\cup\{S'\rightarrow S\},S')$ با ایجاد سمبل آغازین S' و افزودن قانون $S'\rightarrow S$ به مجموعه قوانین G پدید خواهد آمد. هر دو گرامر زبان یکسانی را تولید میکنند زیرا برای هر رشته u قابل اشتقاق در G توسط $S\Rightarrow u$ را میتوان با اشتقاق $S'\Rightarrow S\Rightarrow u$ در گرامر G' بدست آورد .

[1] Chain
[2] Usefullness Variable
[1] Recursive variable

تعریف : متغیر Aرا بازگشتی گوئیم اگر A پس از یک یا بیشتر از یک مرحله اشتقاق خودش را تولید کند.

1- $S \to aSb$

2- $S \to MN$

3,4 $M \to Am|m$

5,6 $A \to Ma|a$

7,8 $N \to Nn|n$

9- $S \to \lambda$

$S \to aSb$

$S \to MN \to AmN \to MamN$

اگر A در یک مرحله از اشتقاق خودش را تولید کند بازگشتی مستقیم[1] است در غیراینصورت بازگشتی غیرمستقیم[2] است.

۳-۱-۳ مرحله ۱ حذف نمادآغازگربازگشتی

اگر S در سمت راست قوانین ظاهر شود در طول مراحل اشتقاق هم ظاهر میشود پس استفاده از قانون $S \to \lambda$ در این موقعیت امکانپذیر است. با حذف نماد آغازگر بازگشتی مطمئن میشویم که استفاده از قانون $S \to \lambda$ فقط در تولید جمله $\omega=\lambda$ امکانپذیر است.

$$\begin{array}{lcl} S \to aSb & & S' \to S \\ & \Rightarrow & S \to aSb \\ S \to \lambda & & S \to \lambda \end{array}$$

- **روش کار**: برای تبدیل هر گرامر $G(\Sigma,V,R,S)$ که در آن S بازگشتی است به گرامر G_1 بقسمیکه $L(G_1)=L(G)$ به شکل زیر عمل میکنیم :

$G=(\Sigma,V\cup\{S'\},R\cup\{S'\to S\},S') \quad S'\notin V$

S' را به مجموعه متغییرها اضافه میکنیم.

$S'\to S$ را به مجموعه قوانین اضافه میکنیم .

نماد آغازگر S' خواهد شد.

۳-۱-٤ مرحله ۲ حذف قوانین λ

قوانین λ به دو دلیل تولید میشوند :

۱- حذف نمادآغازین بازگشتی

۲- بدلیل سهولت تعریف گرامر بوسیله طراح قوانین

[1] Direct

[2] Indirect

قوانین λ ممکن است به ابهام گرامر کمک کنند و همچنین درک گرامر را مشکل سازند .بـه حلقـه نامتنـاهی در پارسر پائین به بالا منجر میشوند. البته این قوانین امکان تولید یک رابطه بین طول جمله ω و تعداد مراحل اشـتقاق را به حداقل می رسانند.

۳-۱-٤-۱ افزایش مراحل اشتقاق

در قانون لامبدا ($A\to\lambda$) A متغیر لامبدا نام دارد. یک متغییر لامبدا ممکن است بطور غیر مستقیم شناسایی شود به این معنی که پس از چند مرحله از اشتقاق با λ جایگزین شود. λ و هر متغییری که بطور مستقیم یا غیر مستقیم λ را تولید میکند ؛ متغییر λ نام دارند.

۳-۱-٤-۲ خلاصه مراحل حذف λ

۱- *2^n قانون در گرامر جدید تولید میکنیم که در هر قانون متغییر های λ در یک ترکیب مشخص با λ جـایگزین شده اند.*

۲- *قوانینی که فاقد متغییر λ هستند عینا در گرامر جدید وارد میشوند.*

۳- *اگر $S\to\lambda$ تولید میشود باید به گرامر جدید منتقل شود.*

- **روش کار**: برای حذف قوانین λ به ازای هر قانون $A\to\alpha_1 A_1\alpha_2 A_2\ldots\alpha_n A_n\alpha_{n+1}$ که در آن A_i متغییر λ است به شکل زیر عمل میکنیم:

حالات مختلفی را که متغیرهای لامبدا میتوانند λ باشند ؛ در نظر میگیریم و مراحل مختلف را تولید میکنیم:

- مثال ۳-۱ حذف نماد آغازگر بازگشتی و متغیر لامبدا در زیر نشان داده شده است :

$$\left\{\begin{array}{l} S\to aSb \\ S\to\lambda \end{array}\right. \longrightarrow \left\{\begin{array}{l} S'\to S \\ S\to aSb \\ S\to\lambda \end{array}\right. \longrightarrow \left\{\begin{array}{l} S'\to\lambda \\ S'\to S \\ S\to ab \\ S\to aSb \end{array}\right.$$

- مثال ۳-۲ گرامر زیر را در نظر بگیرید :

$$\begin{cases} S\to AB \\ B\to MNaNA \\ M\to mM\mid m \\ N\to\lambda \\ N\to nN\mid n \\ A\to aA \\ A\to\lambda \end{cases}$$

N	N	A	B	
--	--	λ	MnaN	1
--	λ	λ	Mna	2
λ	--	--	MaNA	3
λ	--	λ	MaN	4
λ	λ	--	MaA	5
λ	λ	λ	Ma	6
--	--	--	MnaNA	7
--	λ	--	MnaA	8

$$
\begin{cases}
S \rightarrow AB & ------> \begin{cases} S \rightarrow B & 1 \\ S \rightarrow AB & 2 \end{cases} \\
B \rightarrow MNaNA & ------> \begin{cases} B \rightarrow MNaN & 3 \\ B \rightarrow MNa & 4 \\ B \rightarrow MaNA & 5 \\ B \rightarrow MaN & 6 \\ B \rightarrow MaA & 7 \\ B \rightarrow Ma & 8 \\ B \rightarrow MNaNA & 9 \end{cases} \\
M \rightarrow mM \mid m & 10,11 \\
N \rightarrow nN \mid n & ------> \begin{cases} N \rightarrow n & 12 \\ N \rightarrow nN & 13 \end{cases} \\
A \rightarrow \lambda & \\
A \rightarrow aA & ------> \begin{cases} A \rightarrow a & 14 \\ A \rightarrow aA & 15 \end{cases}
\end{cases}
$$

در گرامر زیر متغیرهای لامبدا حذف شده و گرامر از نو بازنویسی شده است.

$$\begin{cases} S \to B \\ S \to AB \\ B \to MNaN \\ B \to MNa \\ B \to MaNA \\ B \to MaN \\ B \to MaA \\ B \to Ma \\ B \to MNaA \\ B \to MNaNA \\ N \to n \\ N \to nN \\ A \to a \\ A \to aA \end{cases}$$

۳-۱-۵ مرحله ۳ حذف زنجیره ها [۱]

پس از آنکه گرامر G تبدیلات مربوط به حذف نماد آغازین بازگشتی حذف و قوانین λ را پشت سر گذاشت باید حذف زنجیره ها انجام شود:

➢ **عوامل تولید زنجیره ها**

۱- *حذف نماد آغازین بازگشتی و حذف قوانین λ*

۲- *طراحی طراح گرامر*

تنها مشکل ناشی از زنجیره ها افزایش تعداد مراحل اشتقاق در عمق درخت اشتقاق در گراف پارس است.

➢ **روش کار**

۱- *شناسایی قوانین A→A و حذف آنها از گرامر*

۲- *شناسایی زنجیره ها و حذف آنها*

➢ **فواید حذف زنجیره ها**

۱- *تعداد مراحل اشتقاق کاهش می یابد*

۲- *نزدیک تر شدن به فرم گریباخ*

[۱] Elimination Of chain rules

برای شناسایی زنجیره ها و حذف آنها ؛ ابتدا باید مجموعه هایی تشکیل دهیم که هر مجموعه ؛ متغیرهایی را که توسط قوانین زنجیره بهم مربوط میشوند در خود داشته باشد سپس هر یک از مجموعه ها را ملاک اصلی حذف قوانین زنجیره و تولید قوانین جدید قرار می دهیم.

❑ مثال ۳–۳ در گرامر زیر مجموعه زنجیره ها عبارتند از [A,T,X] , [A,B,M,N]

$$\begin{cases} A \to B \\ B \to M \\ M \to N \\ N \to a_1 | a_2 \\ A \to T \\ T \to X \end{cases}$$

❑ مثال ۳–٤ گرامر زیر را در نظر بگیرید :

$S \to AB$
$S \to MN$
$A \to B$
$B \to M$
$M \to N$
$N \to nN | n$
$A \to T$
$T \to X$
$X \to xX | x$

پس از شناسایی زنجیره ها و با توجه به اصل جایگزینی اقدام به حذف آنها میکنیم به این صورت که :

$$\begin{cases} A \to B \\ B \to u_1 | u_2 | \ldots | u_n \end{cases}$$

$$\downarrow$$

$$\begin{cases} A \to u_1 | u_2 | u_3 \ldots | u_n \\ B \to u_1 | u_2 | \ldots | u_n \end{cases}$$

و بلافاصله از A به جمله خواهیم رسید که در واقع ساده تر خواهد شد.

برای شناسایی زنجیره ؛ همانطور که مشاهده شد از الگوریتمی استفاده میکنیم که متغیرهایی را که توسط زنجیره ها با هم در ارتباط هستند در یک مجموعه قرار میدهد .

تعریف : مجموعه متغیرهایی که از طریق زنجیره ها با هم درارتباطند و سر منشا آنها A است Chain(A) یـا C(A) نامیده میشود. در مثال ۳–٤ زنجیره ها را میتوان به شکل زیر نوشت:

$$\begin{cases} C(A)=[A,B,M,N,T,X] \\ C(B)=[B,M,N] \\ C(S)=[S] \\ C(M)=[M,N] \\ C(N)=[N] \\ C(X)=[X] \\ C(T)=[T,X] \end{cases} \Rightarrow \begin{cases} S\to AB|MN \\ A\to nN|n|xX|x \\ B\to nN|n \\ M\to nN|n \\ N\to nN|n \\ T\to xX|x \\ X\to xX|x \end{cases}$$

➢ **الگوریتم ساختن C(A)**[1]

```
FOR EVERY VARIABLES A∈V DO
        C(A)=[A]
        REPEAT
                FLAG=FALSE
                FOR EVERY RULE  r:P→Q DO
                        IF P IS A MEMBER OF C(A)& Q IS NOT A
                 MEMBER  OF C(A) THEN
                                C(A)=C(A)∪[Q]
                                FLAG=TRUE
                        END-IF
                END-FOR
        UNTIL (NOT FLAG)
END-FOR
```

برای حذف زنجیره ها روی مجموعه زنجیره ای چند عضوی کار میکنیم .زنجیره ای مانند [B,M,N] از راست به چپ بررسی میشود یعنی برای N بررسی میکنیم که آیا اولا M→N در قوانین وجود دارد یا خیر و در صورت وجود زنجیره حذف میشود . حذف یک قانون هنگامی انجام میشود که آنن عضوی از قوانین گرامر باشد .

$$\begin{cases} S\to AB \\ S\to MN \\ B\to nN|n \\ A\to B & [A,B] \\ M\to nN|n \\ N\to nN|n \\ A\to T & [A,T,X] \\ T\to X \\ X\to xX|x \end{cases}$$

[1] Algorithm of construction of the set C(A)

- **بررسی متغیر N**

$$\begin{cases} M \to N \\ X \to N \\ T \to N \\ B \to N \\ A \to N \end{cases}$$

هیچیک از قوانین در گرامر وجود ندارند.

- **بررسی متغیر M**

$$\begin{cases} X \to M \\ T \to M \\ B \to M \\ A \to M \end{cases}$$

. هیچیک از قوانین در گرامر وجود ندارند

- **بررسی متغیر X**

$$\begin{cases} T \to X \Rightarrow \begin{cases} T \to xX \mid X \\ X \to xX \mid X \end{cases} \\ B \to X \\ A \to X \end{cases}$$

- **بررسی متغیر T**

$$\begin{cases} B \to T \\ A \to T \Rightarrow \begin{cases} A \to xX \mid X \\ T \to xX \mid X \\ X \to xX \mid X \end{cases} \end{cases}$$

- **بررسی متغیر B**

$$A \to B \Rightarrow \begin{cases} A \to nN \mid n \\ B \to nN \mid n \end{cases}$$

➢ با انجام حذف زنجیره گرچه تعدد قوانین وجود دارد ؛ مراحل اشتقاق کاهش یافته است.

۳-۱-٦ مرحلهٔ ٤ : حذف متغیرهای غیر مفید[1]

متغیرهای عیر مفید ممکن است بر اثر حذف زنجیره ها و یا خطای طراح گرامر پدید آیند. اینها متغیر هایی هستند که در تولید جمله های زبان شرکت نمیکنند و این عدم شرکت در تولید جمله ها به دو دلیل است:

۱- از نماد آغازین طی مراحل اشتقاق تولید نمیشوند. مثلاً در مثال ۳-٤ T اصلاً در گراف پارس دیده نمیشود.

۲- متغیرها قادر به تولید دنباله أی از عناصر الفبا نیستند.

بطور مثال اگر $\begin{cases} S \to EF \\ E \to fF \\ F \to eE \end{cases}$ هر چه این دنباله را بسط دهیم هیچگاه به رشتهٔ $\omega \in \sum^*$ نخواهیم رسید .

- **نقطه ضعف متغیرهای غیر مفید** : متغیرهای غیر مفید میتوانند علاوه بر افزایش فضای گراف پارس و زمان جستجوی گرامر، سبب ایجاد حلقه های نامتناهی در پارسرهای از بالا به پایین شوند .

بنابراین گرامر $G = (\sum, V, R, S)$ که در آن S بازگشتی نیست و تنها قانون لامبدا $S \to \lambda$ که $\lambda \in L(G)$ و فاقد زنجیره ها است. می توان به گرامر $G' = (\sum, V', R', S)$تبدیل گردد که فاقد مغیرهای عیر مفید است و $L(G) = L(G')$

مثال: در گرامر زیر متغیرهای A و X غیرمفید هستند و باید از گرامر حذف شوند.

$$\begin{cases} S \to ABMm \\ S \to PQy \\ A \to aA \mid xX \\ X \to xA \\ M \to mM \mid m \\ P \to Pp \mid p \\ Q \to q \\ B \to bB \mid b \end{cases}$$

۳-۱-٦-۱ مراحل تولید G'

- G' در دو مرحله تولید میشود:

۱- حذف متغیرهایی که قادر به تولید الفبا نیستند از گرامر G و تولید G'

۲- حذف متغیرهای غیر قابل دسترس[1] از نماد آغازگر در $G1$ و تولید G'

[1] Useless Variables

[1] Unreachable

مرحلهٔ یکم : تولید متغیرهایی که قادر به تولید دنباله أی از عناصر الفبا نیستند و حذف آنها و قوانین وابسته از گرامر G و تولید گرامر G'

الف) شناسایی متغیرهایی که قادر به تولید عناصر الفبا نیستند.

مجموعه متغیرهای غیر مفید در این مرحله (non Token Generator) non-T-G

ب) مجموعه متغیرهایی که قادر به تولید دنباله أی از عناصر الفبا هستند. T-G

مثلاً در مثال پیشین $T-G=\{B,Q,P,M\}\cup\{S\}$

بنابراین

non_T_G=V - T_G

الگوریتم ساختن مجموعه متغیرهائی که رشته های پایانی را تولید میکنند[۲] .

Repeat
Flag=false
For every rule $r: A\rightarrow \alpha_1 A_1 \alpha_2 A_2 ... \alpha_n A_n \alpha_{n+1} \quad ,\alpha_i \in \sum\limits_{1\le i\le n+1}^{*}$
If $A\notin T-G \quad \& \quad \underset{1\le i\le n}{A_i} \quad ,A_i\in T-G$ *then*
$T-G=T-G\cup\{A\}$ *; add A to token-generator's set*
flag=tre
end_if
end_for
until (NOT flag)

❑ مثال ۳–۵ گرامر زیر را در نظر بگیرید :

$$G:\left|\begin{array}{l} S\rightarrow AC\,|\,BS\,|\,B \\ A\rightarrow aA\,|\,aF \\ B\rightarrow CF\,|\,b \\ C\rightarrow cC\,|\,D \\ D\rightarrow aD\,|\,BD\,|\,C \\ E\rightarrow aA\,|\,BSA \\ F\rightarrow bB\,|\,b \end{array}\right.$$

با اعمال الگوریتم فوق میتوان متغیرهایی از G را که رشته ترمینالی یا عناصر الفبا تولید میکنند به دست آورد:

[۲] Algorithm :Construction of the set of variables that derive Terminal Strigs

iteration	T-G
0	{B,F}
1	{B,F,A,S}
2	{B,F,A,S,E}
3	{B,F,A,S,E}

بنابراین میتوان گرامر $G' = \left(\sum ', V', R', S\right)$ را بگونه ای تعریف کرد که:

$i)$ $L(G') = L(G)$

ii) Every variabe in G' derives a terminal string in G'

$$V' = \{S, A, F, B, E\}$$
$$\sum ' = \{a, b\}$$
$$R' : \left| \begin{array}{l} S \to BS \mid B \\ A \to aA \mid aF \\ B \to b \\ E \to aA \mid BSA \\ F \to bB \mid b \end{array} \right.$$

$V' = V - non - T - G$

$R' = R - \{r \mid r : A \to \alpha$, حداقل یکی از عناصر non-T-G در آن وجود دارد

❑ مثال ۳–۷ گرامر زیر را در نظر بگیرید:

$$\begin{cases} S \to ABMm \\ S \to PQy \\ A \to aA \mid xX \\ X \to xA \\ M \to mM \mid m \\ P \to Pp \mid p \\ Q \to q \\ B \to bB \mid b \end{cases}$$

$$non - T - G = \{A, X\}$$
$$T - G = \{B, Q, M, P, S\}$$

G' سادهٔ شدهٔ گرامر G و به شکل زیر می‌باشد که در آن متغیرهای غیرمفید و قوانین وابسته حذف شده است:

$$G': \begin{cases} S \to PQy \\ M \to mM \mid m \\ P \to Pp \mid p \\ Q \to q \\ B \to bB \mid b \end{cases}$$

مرحلهٔ دوم: حذف متغیرهایی که از نماد آغازگر تولید نمیشوند و قوانین وابسته از گرامر G'

الف) شناسایی متغیرهای غیر قابل دسترس از S و تولید مجموعه متغیرهای غیر قابل دسترس

مجموعه متغیرهای قابل دسترس [1]

$$Unreachable = V' - Reachable$$

بطور مثال M در مثال ۳-۷ غیر قابل دسترس است .

- ***الگوریتم مربوط به تشخیص متغیرهای غیر قابل دسترس***

```
   reachable={S}
repeat
 flag=false
 for every rule r : A → αAα_1A_1...α_nA_nα_{n+1}    ,∀α_i ∈ Σ*_{0≤i≤n+1}  do
   if A ∈ reachables   then
    for every A_i    ;0 ≤ i ≤ n  do
     if A_i ∉ reachables   then
       reachables = reachables ∪ {A_i}
       flag=true
      end_if
    end_for
   end_if
 end_for
until(NOT flag)
```

در مثال ۳-۷ `reachables={ S, P, Q}` و `unreachables={B, M}`

[1] Reachable

ب) حذف هر قانونی در G1 که حداقل یکی از عناصر unreachables در آن ظاهر شده باشد و تولید G'

ترتیب اعمال این دو مرحله بسیار اهمیت دارد:

G

↓

متغیرهایی که به دنباله أی از عناصر الفبا ختم نمیشوند، شناسایی و حذف آن‌ها

↓

G1

متغیرهای غیر قابل دسترس در قوانین وابسته و حذف آن‌ها

↓

G'

و اما :

G

↓

متغیرهای غیر قابل دسترس و حذف قوانین وابسته

↓

G1

↓

شناسایی و حذف متغیررهایی که به دنباله أی از عناصر الفبا ختم نمیشوند

↓

G2

حذف متغیرهای غیر قابل دسترس و قوانین وابسته

۳-۱-۷ مرحلهٔ پنجم: حذف بازگشت چپ [1]

دقت در این مرحله بسیار ضروری است زیرا:

- بازگشت چپ به معنای متغیرهای بازگشتی است که خود را بعنوان سمت چپ ترین عنصر در شبه جمله تولید میکنند $A \overset{*}{\Rightarrow} A\cdots$ (عامل حلقهٔ نامتناهی در پارسرهای بالا به پایین)
- از سمت چپ token تولید نمیکنند و این بر خلاف فلسفهٔ فرم نرمال گریباخ گریباخ است (تولید token از سمت راست انجام میگیرد در رحالیکه ما میخواهیم از سمت راست صورت بگیرد.)

[1] Removable of left recursion

➢ برای حذف بازگشت چپ باید متغیرهای بازگشت چپ را طوری در قوانین استفاده کنیم کـه عناصـر الفبـا در سمت چپ تولید نشوند:

❑ مثال ۳–۸ گرامر زیر را در نظر بگیرید :

$$S \to AB$$
$$A \to BaA \mid a$$
$$B \to bBa \mid Ab \mid b$$

در این گرامر بازگشت چپ به چشم میخورد :

$$A \Rightarrow BaA \Rightarrow AbaA$$
$$B \Rightarrow Ab \Rightarrow BaAb$$

پس ابتدا باید بازگشت چپ را تشخیص بدهیم و سپس آنها را حذف کنیم .

بازگشت چپ ← بازگشت مستقیم چپ

بازگشت چپ ← بازگشت غیر مستقیم چپ

برای حذف بازگشت چپ از گرامر G که فاقد نماد آغازین بازگشتی ، قوانین لامبدا $S \to \lambda$ و زنجیره ها و متغیرهای غیر مفید است و تولید گرامر G' به صورتی‌که $L(G)=L(G')$ و G' فاقد بازگشت چپ است به طریقهٔ زیر عمل میکنیم :

۱– حذف بازگشت مستقیم چپ

۲– تبدیل بازگشت غیر مستقیم چپ به نوع مستقیم

۳– حذف بازگشت مستقیم چپ

مثلاً $A \to Aa$ از نوع بازگشتی مستقیم چپ است .

۳-۱-۷-۱ حذف بازگشت مستقیم چپ [۱]

الف) شناسایی متغیرهای بازگشتی چپ مستقیم

اگر متغیرهای گرامر را به نحوی شماره گذاری کنیم که نماد آغازگر کوچکترین شماره را داشته باشد در اینصورت به ازای هر قانون $A \to A'\cdots$ در صورتیکه شمارهٔ A' = شمارهٔ A بازگشت مستقیم وجود دارد.

ب) حذف بازگشت چپ

اگر A متغیر بازگشتی چپ باشد ، تمامی قوانین جایگزین کنندهٔ A را از گرامر جدا میکنیم .

$$A \to A\alpha_1 \mid A\alpha_2 \mid \cdots \mid A\alpha_n$$
$$A \to B_1 \mid B_2 \mid B_3 \mid \cdots \mid B_m$$

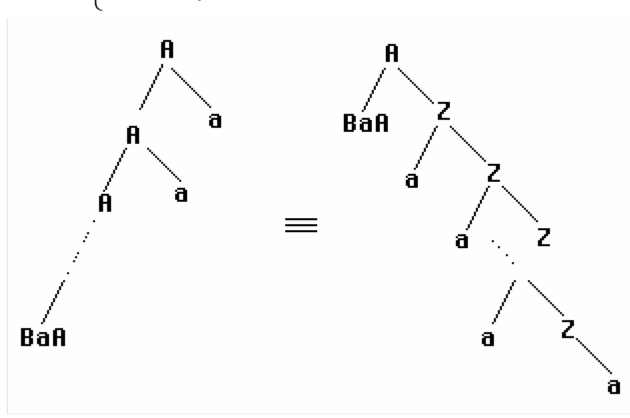
که A سمت چپ ترین عنصر نیست .

از متغیر کمکی Z که عنصری از V نیست کمک میگیریم و قوانین جدید را تولید کرده، جایگزین قوانین R' در گرامر G میکنیم :

$$A \to B_1 \mid B_2 \mid \cdots \mid B_k$$
$$A \to B_1Z \mid B_2Z \mid \cdots \mid B_k z$$
$$Z \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$$
$$Z \to \alpha_1Z \mid \alpha_2Z \mid \cdots \mid \alpha_nZ$$

مثلاً
$$A \to Aa$$
$$A \to BaA \mid a$$

$$\begin{cases} A \to BaA \mid a \\ A \to BaAZ \mid aZ \\ Z \to a \mid aZ \end{cases}$$



---

[۱] Direct left recursion

۳-۱-۷-۲ حذف بازگشت غیر مستقیم چپ[۱]

الف) شناسایی متغیرهای بازگشتی غیر مستقیم چپ

با استفاده از مجموعه متغیرهای شماره گذاری شده به ازای هر قانون $A \rightarrow A'\alpha$ اگر شمارهٔ A از شمارهٔ A' کوچکتر باشد $(\#A < \#A')$ در این‌صورت بازگشت چپ وجود ندارد.

$$
\begin{aligned}
&S \rightarrow AB \\
&A \rightarrow BaA \mid a \\
&B \rightarrow bBa \mid Ab \mid b \\
&A \rightarrow BaAZ \mid aZ \\
&Z \rightarrow a \\
&Z \rightarrow aZ
\end{aligned}
$$

مثال: در گرامر روبرو

$$
\left|\begin{array}{l}
A \rightarrow B\alpha \\
B \rightarrow C\beta \\
C \rightarrow A\delta
\end{array}\right.
$$

A<B<C<A و در هر سه متغیر بازگشت غیر مستقیم چپ وجود دارد:

$$A \Rightarrow B\alpha \Rightarrow C\beta\alpha \Rightarrow A\delta\beta\alpha \Rightarrow B\alpha\delta\beta\alpha \Rightarrow C\beta\alpha\delta\beta\alpha$$

اگر هنگام شماره گذاری به رابطهٔ $A_0 < A_1 < A_2 < \cdots < A_i < \cdots < A_n < A$ برسیم در اینصورت $A_0, A_1, \cdots, A_n$ بازگشتی غیر مستقیم از چپ هستند و باید حذف شوند. A تا A_n را در لیستی به شکل زیر قرار میدهیم ، سپس با استفاده از اصل جایگزینی بازگشت غیر مستقیم را به نوع مستقیم تبدیل میکنیم (مشابهٔ حذف زنجیره ها)

$$
\begin{cases}
S \rightarrow AB \\
A \rightarrow BaA \mid a \\
B \rightarrow bBa \mid Ab \mid b \\
A \rightarrow BaAZ \mid aZ \\
Z \rightarrow a \\
Z \rightarrow aZ
\end{cases}
$$

---

[۱] Indirect left recursion

$$\Rightarrow \begin{cases} S \to AB \\ A \to bBaaA \mid AbaA \mid baA \mid a \\ B \to bBa \mid Ab \mid b \\ A \to bBaaAZ \mid AbaAZ \mid baAZ \mid aZ \\ Z \to a \\ Z \to aZ \end{cases}$$

$[A, A_1, \cdots, A_i, \cdots, A_n]$ که A_i سمت راست ترین عنصری است کـه علامـت نخـورده اسـت و آنـرا علامـت میزنیم . سپس به ازای هر A_j که j<I است به دنبال قوانینی به شکل $A_j \to A_i\alpha$ میگردیم و A_i را با قـوانین مربوطه – در صورت وجود – جایگزین میکنیم .

۳-۱-۷-۳ **حذف بازگشت مستقیم چپ** مجدداً به همان شکل پیشین میباشد .

پس از این مرحله تبدیلات زیر را در گرامر اعمال میکنیم :

٤- حذف زنجیره ها

٥- حذف متغیرهای غیر مفید

٦- تبدیل نهایی به فرم نرمال گریباخ

گرامر G را که تبدیلات قبلی را کاملاً پشت سر گذاشته است میتوان با استفاده از اصل جایگزینی به گرامر گریبـاخ تبدیل کرد.

الف) متغیرهای V را طوری شماره گذاری میکنیم که S کوچکترین شماره را داشته باشد و به ازای هر قانون $A \to A'\alpha$ شمارهٔ A از شمارهٔ A' کوچکتر باشد، سپس متغیرها را بر اساس شمارهٔ آنها در یک لیست به شکل صعودی مرتب میکنیم.

$$\begin{aligned} & S \to AB \\ & A \to bBaaA \mid AbaA \mid baA \mid a \\ & B \to bBa \mid Ab \mid b \\ & A \to bBaaA \mid AbaAZ \mid baAZ \mid aZ \\ & Z \to a \\ & Z \to aZ \end{aligned}$$

و بالاخره بازگشت مستقیم از چپ حذف میشود:

$$\begin{array}{|l} S \to AB \\ A \to bBaaA \mid baA \mid a \mid baAZ \mid aZ \mid bBaaAZ \\ A \to bBaaAZ_1 \mid baAZ_1 \mid aZ_1 \mid baAZZ_1 \mid aZZ_1 \mid bBaaAZZ_1 \\ Z_1 \to AbaAZ_1 \mid AbaAZZ_1 \mid AbaA \mid AbaAZ \\ B \to bBZ \mid Ab \mid b \\ Z \to aZ \mid a \end{array}$$

$\to [S, Z1, B, A, Z]$

و بجای A قوانین مربوطه را جایگزین میکنیم :

برای A جایگزینی وجود دارد:

$$\begin{cases} B \to Aa \\ Z_1 \to Aa \end{cases}$$

برای Z جایگزینی وجود ندارد:

$$\begin{cases} A \to Z\alpha \\ B \to Z\alpha \\ Z_1 \to Z\alpha \\ S \to Z\alpha \end{cases}$$

$[S, A_1, A_2, \cdots, A_i, \cdots, A_n]$ که A_i سمت راست ترین عنصر علامت نخورده است . سپس به ازای هر قانون به شکل $A_j \to A_i$ که j<I است A_i را با استفاده از قوانین مربوطه جایگزین میکنیم و در نهایت قوانین جدید تولید میشود . پس از پایان این مرحله هر قانون گرامر به شکل $A \to a\alpha$ در میآید که $a \in \sum, \alpha \in \left(\sum \cup V\right)^*$

مرحلهٔ پایانی:

$$B \to bBaaA \xrightarrow{Geribach} \begin{cases} T_1 \to a \\ T_2 \to b \\ B \to bBT_1T_1AT_2 \end{cases}$$

سپس به ازای هر عنصر الفبا a_i در α $[\alpha = \cdots \alpha_i \cdots]$ از متغیر جدید T_i استفاده میکنیم و قانون $T_i \to a_i$ را به گرامر اضافه و a_i در α را با T_i جایگزین میکنیم.

۲-۳ گرامر نرمال گریباخ

1. $A \to a$
2. $A \to aA_1A_2 \cdots A_n$
3. $S \to \lambda \quad if \quad \lambda \in L(G)$

۳-۳ گرامر چامسکی

1. $A \to a$
2. $A \to A_1A_2$
3. $S \to \lambda \quad if \quad \lambda \in L(G)$

- **گرامر نرمال چامسکی**

1. $A \to a$
2. $A \to A_1A_2 \qquad ; A \prec\succ A_1$
3. $S \to \lambda \quad if \quad \lambda \in L(G)$

خلاصهٔ تبدیل های مورد نیاز برای تولید گرامر نرمال چامسکی از هر گرامر مستقل از متن

۱. حذف نماد آغازگر بازگشتی
۲. حذف قوانین لامبدا
۳. حذف زنجیره ها
۴. حذف متغیرهای غیر مفید
۵. حذف بازگشت چپ
۶. تبدیل نهایی به فرم چامسکی

تبدیل نهایی $G = \left(\sum, V, R, S\right)$ که فاقد قوانین لامبدا، فاقد قوانین آغازگر بازگشتی در گرامر نرمال، فاقد زنجیره ها، فاقد متغیرهای غیر مفید ، فاقد بازگشت چپ در گرامر نرمال است به گرامر $G' = \left(\sum, V', R', S\right)$ به شکل زیر تبدیل میشود:

۱. به ازای هر قانون به فرم $A \to a\alpha \qquad ; \alpha \in \left(\sum, V\right)^+$ دو متغیر جدید T_i و T_j را تعریف می کنیم و مجموعه قوانین زیر را جایگزین قانون فوق میکنیم :

$$\begin{cases} A \to T_iT_j \\ T_i \to a \\ T_j \to \alpha \end{cases}$$

۲. به ازای هر قانون به فرم $A \to A'\alpha \qquad ; \alpha \in \left(\sum, V\right)^+$ که $length(\alpha) \geq 2$ از متغیر جدید T_i استفاده میکنیم و قوانین جدید را جایگزین قوانین فوق میکنیم :

$$\begin{cases} A \to A'T_i \\ T_i \to \alpha \end{cases}$$

۳. به ازای هر قانون به فرم $A \to a_1a_2$ مجموعه قوانین جدید را با استفاده از دو متغیر جدید T_i و T_j تولید میکنیم :

$$\begin{cases} A \to T_iT_j \\ T_i \to a_1 \\ T_j \to a_2 \end{cases}$$

۴. به ازای هر قانون به فرم $A \to a_1A'$ با استفاده از متغیر T_i قوانین جدید جایگزین قوانین فوق میگردند:

$$\begin{cases} A \to T_iA' \\ T_i \to a_1 \end{cases}$$

۵. به ازای هر قانون به فرم $A \to A'a$ با استفاده از متغیر جدید T_i مجموعه قوانین را جایگزین قانون فوق میکنیم:

$$\begin{cases} A \to A'T_i \\ T_i \to a \end{cases}$$

پنج روش فوق را برای مجموعهٔ R آنقدر تکرار میکنیم تا هیچ قانونی که فرم چامسکی را نقض میکند در گرامر ظاهر نشود. در پایان این مرحله G' تولید شده است .

گرامر زیر را که ۵ مرحله تبدیل را پشت سر گذاشته است به فرم نهائی نرمال چامسکی تبدیل کنید.

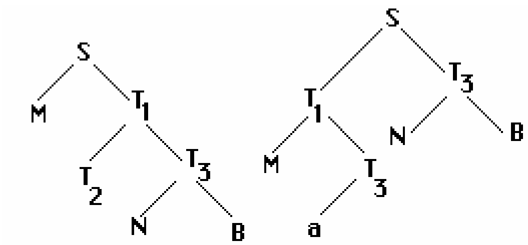
$$\begin{cases} S \to MaNB \\ S \to anb \\ M \to MaB \\ N \to nNBa \\ M \to ma \\ N \to n \\ B \to bBB \\ B \to b \end{cases}$$

حل:

$$\begin{cases}
S \to MaNB \quad --> \begin{cases} S \to MT_1 \\ T_1 \to aNB \end{cases} --> \begin{cases} S \to MT_1 \\ T_1 \to T_2T_3 \\ T_2 \to a \\ T_3 \to NB \end{cases} \\
S \to anb \quad ---> \begin{cases} S \to T_2T_4 \\ T_4 \to nb \end{cases} ---> \begin{cases} S \to T_2T_4 \\ T_4 \to T_5T_6 \\ T_5 \to n \\ T_6 \to b \end{cases} \\
M \to MaB \quad --> \begin{cases} M \to MT_7 \\ T_7 \to aB \end{cases} --> \begin{cases} M \to MT_7 \\ T_7 \to T_2B \end{cases} \\
N \to nNBa \quad --> \begin{cases} N \to T_5T_8 \\ T_8 \to NBa \end{cases} ---> \begin{cases} N \to T_5T_8 \\ T_8 \to NT_9 \\ T_9 \to Ba \end{cases} --> \begin{cases} N \to T_5T_8 \\ T_8 \to NT_9 \\ T_9 \to BT_2 \end{cases} \\
M \to ma \quad ---> \begin{cases} M \to T_{10}T_2 \\ T_{10} \to m \end{cases} \\
N \to n \\
B \to bBB \quad ---> \begin{cases} B \to T_6T_{11} \\ T_{11} \to BB \end{cases} \\
B \to b
\end{cases}$$

اگر $S \to MaNB$ را از میان تقسیم کنیم $S \to Ma \vdots NB$ به یک درخت متوازن می‌رسیم که برای گرامر مناسب میباشد .

$$\left| \begin{array}{l} S \to T_1T_2 \\ T_1 \to MT_3 \\ T_2 \to NB \\ T_3 \to a \end{array} \right.$$



بروش دیگری نیز میتوان گرامر مستقل از متن را که تبدیلات قبلی را پشت سر گذاشته است به گرامر چامسکی تبدیل کرد . به این صورت که به ازای هر قانون غیر چامسکی به فرم $A \to \delta_1\delta_2\cdots\delta_n \qquad ;\delta_i \in \left(\sum \cup V\right)$ سمت راست قانون را از میان بطور نسبی تجزیه میکنیم یعنی:

$$\delta_1\delta_2\cdots\delta_i \vdots \delta_{i+1}\cdots\delta_n \qquad i = \left\lfloor \frac{n+1}{2} \right\rfloor$$

سپس با استفاده از متغیرهای کمکی در صورت نیاز اقدام به تولید قوانین جدید میکنیم :

$$\begin{array}{l} A \to T_1T_2 \\ T_1 \to \delta_1\cdots\delta_i \\ T_2 \to \delta_{i+1}\cdots\delta_n \end{array}$$

در صورتیکه گرامر به روش جدید تولید گردد درخت اشتقاق جملهٔ ω بطول n تقریباً عمقی برابر $\lceil \ell og_2^n \rceil$ خواهد داشت و طول این درخت در بدترین حالت n خواهد بود.

۳-۳ گرامرهای چامسکی – گریباخ

قوانین این گرامرها باید بیکی از اشکال زیر باشد :

1. $A \to a$
2. $A \to aA_1A_2\cdots A_n$
3. $A \to A_1A_2\cdots A_n \qquad ;n \geq 2$
4. $S \to \lambda \qquad if \qquad \lambda \in L(G)$

وجود و یا عدم وجود بازگشت چپ مهم نیست زیرا ما نمیخواهیم از این گرامر برای پارسرهای جامع استفاده کنیم.

مثال: گرامر زیر را به فرم نرمال چامسکی بنویسید:

$$\begin{cases} S \to xABz \\ A \to xA \mid \lambda \\ B \to yB \mid A \end{cases}$$

حل:

در آغاز باید دو قانون $B \to A$, $A \to \lambda$ را از گرامر حذف کرد پس با حذف ایندو قانون گرامر به شکل زیر تبدیل خواهد شد:

$$\begin{cases} S \to xBz \mid xz \mid xAz \\ A \to x \\ B \to y \mid xA \mid x \end{cases}$$

اکنون گرامر را به شکل زیر در می آوریم:

$$\begin{cases} S \to XABZ \mid XBZ \mid XAZ \mid XZ \\ A \to XA \mid x \\ B \to YB \mid y \mid XA \mid x \\ X \to x \\ Y \to y \\ Z \to z \end{cases}$$

در آخر میتوان قوانین را به قوانینی با تنها دو متغیر تبدیل کرد:

$$\begin{cases} S \to XP \mid XQ \mid XR \mid XZ \\ P \to AQP \\ A \to XA \mid x \\ B \to XB \mid y \mid XA \mid x \\ R \to AZ \\ X \to x \\ Y \to y \\ Z \to z \\ Q \to BZ \end{cases}$$

مثال: گرامر زیر را در نظر بگیرید:

$$G : \begin{cases} S \to SaB \mid aB \\ B \to bB \mid \lambda \end{cases}$$

آنرا به فرم نرمال چامسکی و گریباخ تبدیل کنید و سپس در هر سه شکل گرامر، اشتقاق جملهٔ w=abaaba را بنویسید.

حل: مراحل کار در زیر نشان داده شده است:

۱- حذف نماد آغازگر بازگشتی

$$S' \to SaB \mid aB \mid Sa \mid a$$
$$S \to SaB \mid Sa \mid aB \mid a$$
$$B \to bB \mid \lambda$$

۲- حذف لامبدا

$$S' \to SaB \mid aB \mid Sa \mid a$$
$$S \to SaB \mid aB \mid Sa \mid a$$
$$B \to bB \mid b$$

۳- حذف بازگشت چپ

$$S' \to ST \mid SA \mid AB \mid a$$
$$S \to AZ \mid aZ \mid AB \mid a$$
$$Z \to TZ \mid AZ \mid T \mid a$$
$$B \to CB \mid b$$
$$T \to AB$$
$$A \to a$$
$$C \to b$$

۴- تبدیل به فرم نرمال چامسکی(CNF)

$$S' \to ST \mid SA \mid AB \mid a$$
$$S \to ST \mid SA \mid AB \mid a$$
$$B \to CB \mid b$$
$$T \to AB$$
$$A \to a$$
$$C \to b$$

۵- تبدیل به فرم نرمال گریباخ(GNF)

$$S' \to aBZT \mid aZT \mid aBT \mid aT \mid aBZA \mid aZA \mid aBA \mid aA \mid aB \mid a$$
$$S \to aBZ \mid aZ \mid aB \mid a$$
$$Z \to aBZ \mid aZ \mid aB \mid a$$
$$B \to bB$$
$$T \to aB$$
$$A \to a$$
$$C \to b$$

اشتقاق جملهٔ w=abaaba در هر سه گرامر در زیر نشان داده شده است:

$\underline{G}$	$\underline{CNF}$	$\underline{GNF}$
$S \Rightarrow SaB$	$S' \Rightarrow SA$	$S' \Rightarrow aBZA$
$\Rightarrow SaBaB$	$\Rightarrow STA$	$\Rightarrow abZA$
$\Rightarrow SaBaBaB$	$\Rightarrow SATA$	$\Rightarrow abaZA$
$\Rightarrow aBaBaBaB$	$\Rightarrow ABATA$	$\Rightarrow abaaBA$
$\Rightarrow abBaBaBaB$	$\Rightarrow aBATA$	$\Rightarrow abaabA$
$\Rightarrow abaBaBaB$	$\Rightarrow abATA$	$\Rightarrow abaaba$
$\Rightarrow abaaBaB$	$\Rightarrow abaTA$	
$\Rightarrow abaabBaB$	$\Rightarrow abaABA$	
$\Rightarrow abaabaB$	$\Rightarrow abaaBA$	
$\Rightarrow abaaba$	$\Rightarrow abaabA$	
	$\Rightarrow abaaba$	

مثال: گرامر نرمال چامسکی زبان زیر را بنویسید:

$$L(G)=\{a^n b^{2n} c^k \mid n,k\ge 1\}$$

حل: گرامری که زبان L را توصیف میکند عبارتست از:

$$G:\begin{cases} S\to XY \\ X\to aXbb \mid abb \\ Y\to Yc \mid c \end{cases}$$

فرم نرمال چامسکی گرامر G به شکل زیر است:

$$G=(\{A,B,C,D,E,X,Y\},\{a,b,c\},S,R)$$

$$R:\left|\begin{array}{l} A\to a \\ B\to b \\ C\to c \\ S\to XY \\ X\to AD \mid AE \\ D\to XE \\ E\to BB \\ Y\to YC \mid c \end{array}\right.$$

مثال: فرم نرمال چامسکی زبان زیررا بنویسید:

$$L(G)=\{a^k b^m c^n \mid n,m,k\ge 1, n=2k\}$$

حل: گرامر مستقل ا ز متن G عبارتست از:

$$G:\begin{cases} S\to aS \mid aXc \mid aXcc \\ X\to aXc \mid aXcc \mid Y \\ Y\to Yb \mid b \end{cases}$$

مثال: فرم نرمال چامسکی گرامر G در زیر نشان داده شده است:

$$G=(\{a,b,c\},\{A,B,C,D,E,X,Y\},S,R\}$$

$$R:\begin{cases} A\to a \\ B\to b \\ C\to c \\ S\to AS \mid AE \\ X\to AE \mid BY \mid b \\ D\to CC \\ E\to XC \mid XD \\ Y\to YB \mid b \end{cases}$$

مثال: گرامر زیر را در نظر بگیرید:

$$G':\begin{cases} S\to AB \\ A\to Cb \\ C\to B \\ B\to BCa \\ B\to \lambda \end{cases}$$

گرامر G'' را طوری بنویسید که $L(G'')=L(G')-\lambda$

حل: کافیست تمامی تولیدات لامبدا را از گرامر حذف کنیم:

$$1.S \to AB \begin{cases} B \to \lambda \end{cases} \Rightarrow \begin{cases} S \to AB \\ S \to A \end{cases}$$

$$2.A \to Cb \begin{cases} C \to B \\ B \to \lambda \end{cases} \Rightarrow \begin{cases} A \to b \\ A \to Cb \end{cases}$$

$$3.C \to B$$

$$4.B \to BCa \begin{cases} B \to \lambda \end{cases} \Rightarrow \begin{cases} B \to Ca \\ B \to a \\ B \to Ba \\ B \to Bca \end{cases}$$

گرامر G'' فاقد متغیر لامبدا است یعنی $L(G'') = L(G') - \lambda$

$$G'': \begin{cases} S \to AB \\ A \to Cb \mid b \\ C \to B \\ B \to BCa \mid Ca \mid Ba \mid a \end{cases}$$

مثال: اگر $L = a^*b^*$ آیا گرامر $G: \begin{cases} S \to AB \\ A \to aA \\ A \to a \\ B \to bB \\ B \to b \end{cases}$ برای L صحیح و کامل است و یا خیر ؟

در اینجا باید یادآوری کرد که منظور از گرامر صحیح آنست که هر جمله أی که از G تولید میشود به L متعلق باشد و گرامر کامل یعنی اینکه تمامی جمله های L تولید گردد . پس باید ثابت کنیم L(G)=L

برای این منظور باید سعی کنیم فرم کلی عباراتی را که توسط گرامر G تولید میشوند پیدا کرده و بـا آن L(G) را بسازیم . سپس باید ثابت کنیم $L \subset L(G)$ است و همچنین $L(G) \subset L$ میباشد پس ثابت میشود L(G)=L

استفاده از خاصیت بازگشتی A $\quad A \Rightarrow aA \Rightarrow aaA \Rightarrow \cdots \Rightarrow a\cdots aA \Rightarrow a^+A \Rightarrow a^+a$

عدم استفاده از خاصیت بازگشتی A $\quad A \Rightarrow a$

استفاده از خاصیت بازگشتی B $\quad B \Rightarrow bB \Rightarrow bbB \Rightarrow \cdots \Rightarrow b\cdots bB \Rightarrow b^+B \Rightarrow b^+b$

عدم استفاده از خاصیت بازگشتی B $\quad B \Rightarrow b$

$$S \Rightarrow A \mid B \Rightarrow (a \cup a^+a) \mid (b \cup b^+b) \Rightarrow a^+b^+$$

پس $L \neq L(G)$

تمرینهای فصل سوم

۳-۱ گرامر زیر را به فرم نرمال چامسکی بنویسید:

$$\begin{cases} S \to xABz \\ A \to xA \mid \lambda \\ B \to yB \mid A \end{cases}$$

۳-۲ گرامر نرمال چامسکی زبان زیر را بنویسید:

$$L(G) = \{a^n b^{2n} c^k \mid n, k \geq 1\}$$

۳-۳ فرم نرمال چامسکی زبان زیررا بنویسید:

$$L(G) = \{a^k b^m c^n \mid n, m, k \geq 1, n = 2k\}$$

۳-٤ گرامر زیر را در نظر بگیرید:

$$G' : \begin{cases} S \to AB \\ A \to Cb \\ C \to B \\ B \to BCa \\ B \to \lambda \end{cases}$$

گرامر G'' را طوری بنویسید که $L(G'') = L(G') - \lambda$

۳-۵ فرم نرمال چامسکی (CNF) زبان $L = \{a^n b^n \mid n \geq 1\}$ را بنویسید.

۳-٦ گرامر زیر مفروض است . گرامر معادل آنرا که شامل متغیرهای بازگشتی آغازین و زنجیره نباشد بنویسید.

$$\begin{cases} S \to ABC \mid \lambda \\ A \to aA \mid a \\ B \to bB \mid A \\ C \to cC \mid \lambda \end{cases}$$

۳-۷ گرامر زیر را به فرم نرمال چامسکی تبدیل کنید:

$$\begin{cases} S \to aAc \mid bbb \\ A \to aA \mid bb \end{cases}$$

۳-۸ در گرامر زیر زنجیره ها را حذف کنید:

$$G: \begin{cases} S \to AB \mid C \\ A \to aA \mid B \\ B \to bB \mid C \\ C \to cC \mid a \mid A \end{cases}$$

۳-۹ در گرامر زیر متغیرهای غیر مفید را حذف کنید:

$$\begin{cases} S \to ACH \mid BB \\ A \to aA \mid aF \\ B \to CFH \mid b \\ C \to aC \mid DH \\ D \to aD \mid BD \mid Ca \\ F \to bB \mid b \\ H \to dH \mid d \end{cases}$$

۳-۱۰ فرم نرمال چامسکی گرامر زیر را بنویسید:

$$\begin{cases} S \to AB \mid C \\ A \to aA \mid B \\ B \to bB \mid C \\ C \to cC \mid a \mid A \end{cases}$$

۳-۱۱ گرامر زیر را در نظر بگیرید:

$$\begin{aligned} &S \to bA \mid aB \\ &A \to a \mid aS \mid bAA \\ &B \to b \mid bS \mid aBB \end{aligned}$$

فرم نرمال چامسکی آنرا بنویسید:

۳–۱۲ گرامر زیر را به فرم نرمال گریباخ تبدیل کنید:

$$G = (\{A_1, A_2, A_3\}, \{a,b\}, R, A1)$$

که در آن
$$R: \begin{cases} A1 \to A2A3 \\ A2 \to A3A1 \\ A2 \to b \\ A3 \to A1A2 \\ A3 \to a \end{cases}$$

۳–۱۳ در گرامر زیر متغیرهای غیر مفید و زنجیره ها را حذف کنید:

$$\begin{cases} S \to Aa \mid B \\ B \to A \mid bb \\ A \to a \mid bc \mid B \end{cases}$$

۳–۱٤ فرم نرمال چامسکی گرامر زیر را بنویسید:

$$G: \begin{cases} S \to ABa \\ A \to aab \\ B \to Ac \end{cases}$$

۳–۱۵ در گرامر زیر قانون لامبدا را حذف کنید:

$$G: \begin{cases} S \to aS_1b \\ S_1 \to aS_1b \mid \lambda \end{cases}$$

۳–۱٦ CNF (فرم نرمال چامسکی)گرامر زیر را بنویسید:

$$\begin{cases} S \to T+S \mid S+T \mid T \\ T \to F*T \mid T*F \mid F \\ F \to n \mid (S) \end{cases}$$

۳–۱۷ گرامری بنویسید که عبارتهای جبری زیر را که شامل +,(,) است تولید کند اشتقاق $S \overset{*}{\Rightarrow} a+b(c+a)$، را بسازید و سپس آنرا به فرم CNF درآورید.

۳–۱۸ گرامر زیر را در نظر بگیرید:

$$G: \begin{cases} S \to SaB \mid aB \\ B \to bB \mid \lambda \end{cases}$$

آنرا به فرم نرمال چامسکی و گریباخ تبدیل کنید و سپس در هر سه شکل گرامر ، اشتقاق جملهٔ w=abaaba را بنویسید.

۱۹-۳ فرم نرمال گریباخ گرامر زیر را بنویسید:

$$\{S \to aSa \mid bSb \mid \lambda$$

۲۰-۳ CNF گرامر زیر را بنویسید:

$$\begin{cases} S \to ABC \mid ab \\ A \to aA \mid b \\ B \to bB \mid a \\ C \to ABC \mid c \end{cases}$$

۲۱-۳ گرامر زیر را به فرمCNF تبدیل کنید:

$$\begin{cases} S \to AAS \mid a \\ A \to SSA \mid b \end{cases}$$

۲۲-۳ فرم نرمال گریباخ (GNF) گرامر زیر را بنویسید:

$$\begin{cases} A \to ASS \mid a \\ S \to SAA \mid b \end{cases}$$

۲۳-۳ اگر G گرامر نرمال چامسکی و رشتهٔ $\omega \in L(G)$ بطول k باشد . برای اشتقاق w بچند مرحله نیاز است؟

۲٤-۳ اگر G گرامر نرمال گریباخ و رشتهٔ $\omega \in L(G)$ بطول k باشد . برای اشتقاق w حداقل بچند مرحله نیاز است؟