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- $A\to aA_1A_2\dots A_n$

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

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 آغازگربازگشتی*
۲- *حذف قوانین* λ
۳- *حذف قوانین به شکل* $A\to B$ *(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- $S\to\lambda$
۲- *وجود حلقه* $A\to B, B\to A$
۳- *بازگشت ازچپ* $A\to A\alpha$

۳-۱-۲ متغیربازگشتی [3]

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.

- **لم 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*

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

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 $G'=(\Sigma,V\cup\{S'\},R\cup\{S'\to S\},S')$ با ایجاد سمبل آغازین S' و افزودن قانون $S'\to S$ 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
[3] Recursive variable

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

1- S→aSb
2- S→MN
3,4 M→Am|m
5,6 A→Ma|a
7,8 N→Nn|n
9- S→λ

S→ aSb
S→MN→AmN→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→λ در این موقعیت امکانپذیر است. با حذف نماد آغازگر بازگشتی مطمئن میشویم که استفاده از قانون S→λ فقط در تولید جمله λ=ω 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 $G(\sum,V,R,S)$ که در آن S بازگشتی است به گرامر G_1 بقسمیکه $L(G_1)=L(G)$ به شکل زیر عمل میکنیم :

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

S′ را به مجموعه متغییرها اضافه میکنیم.

S′→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 .به حلقه نامتناهی در پارسر پائین به بالا منجر میشوند. البته این قوانین امکان تولید یک رابطه بین طول جمله ω و تعداد مراحل اشتقاق را به حداقل می رسانند.

۳-۱-٤-۱ افزایش مراحل اشتقاق

در قانون لامبدا (A→λ) A متغیر لامبدا نام دارد. یک متغیر 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 جایگزین 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 $A\to\alpha_1 A_1 \alpha_2 A_2 \dots \alpha_n A_n \alpha_{n+1}$ 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 :

$$\begin{cases} S\to aSb \\ S\to\lambda \end{cases} \longrightarrow \begin{cases} S'\to S \\ S\to aSb \\ S\to\lambda \end{cases} \longrightarrow \begin{cases} S'\to\lambda \\ S'\to S \\ S\to ab \\ S\to aSb \end{cases}$$

- 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

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

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 $A \to A$ 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 کـه توسط قوانین زنجیره بهم مربوط میشوند در خود داشته باشد سپس هر یک از مجموعه ها را ملاک اصـلی حـذف 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→AB
S→MN
A→B
B→M
M→N
N→nN|n
A→T
T→X
X→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:

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

- **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 میشوود که آنن عضوی از قوانین گرامر باشد .

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

[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 $r: A\to\alpha_1A_1\alpha_2A_2...\alpha_nA_n\alpha_{n+1}$ $,\alpha_i\in\sum^*_{1\le i\le n+1}$
If $A\notin T-G$ & $\underset{1\le i\le n}{A_i}$ $,A_i\in T-G$ *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 :

$$G:\begin{vmatrix} S\to AC\mid BS\mid B\\ A\to aA\mid aF\\ B\to CF\mid b\\ C\to cC\mid D\\ D\to aD\mid BD\mid C\\ E\to aA\mid BSA\\ F\to bB\mid b\end{vmatrix}$$

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'

$$V' = \{S, A, F, B, E\}$$
$$\sum' = \{a, b\}$$
$$R': \begin{cases} S \to BS \mid B \\ A \to aA \mid aF \\ B \to b \\ E \to aA \mid BSA \\ F \to bB \mid b \end{cases}$$

$$V' = V - non - T - G$$
$$R' = R - \{r \mid r : A \to \alpha \quad ,$$

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' ساده 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***

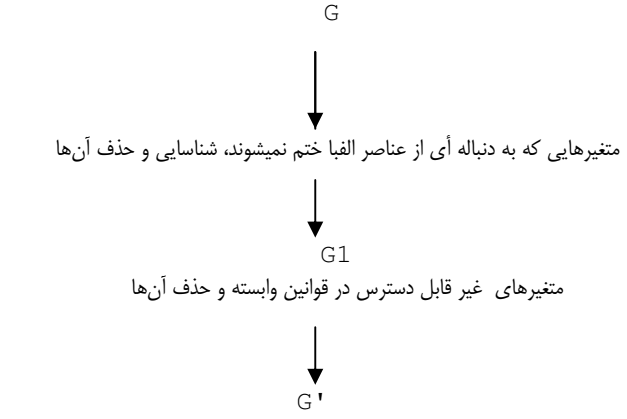
```
reachable={S}
repeat
 flag=false
 for every rule r : A → αAα1A1...αnAnαn+1   ,∀αi ∈ Σ*(0≤i≤n+1)   do
  if A∈ reachables  then
   for every Ai    ;0 ≤ i ≤ n  do
    if Ai ∉ reachables  then
      reachables = reachables ∪ {Ai}
      flag=true
    end_if
   end_for
  end_if
 end_for
until(NOT flag)
```

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

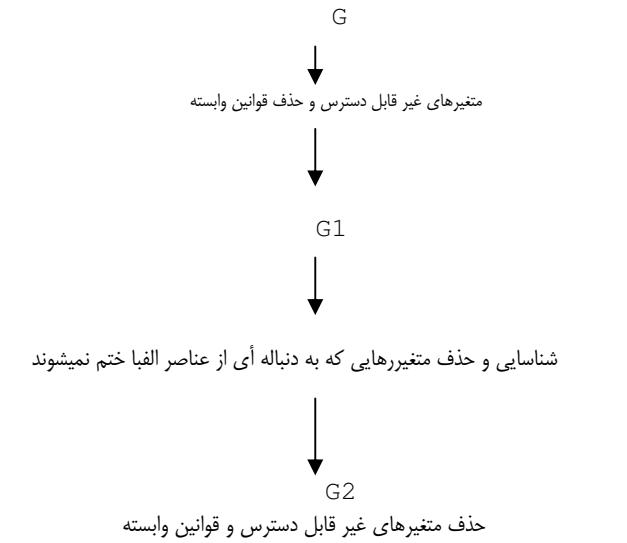
[1] Reachable

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:



و اما :



۳-۱-۷ مرحلهٔ پنجم: حذف بازگشت چپ [۱]

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.)

[۱] Removable of left recursion

- برای حذف بازگشت چپ باید متغیرهای بازگشت چپ را طوری در قوانین استفاده کنیم که عناصر الفبا در سمت چپ تولید نشوند:

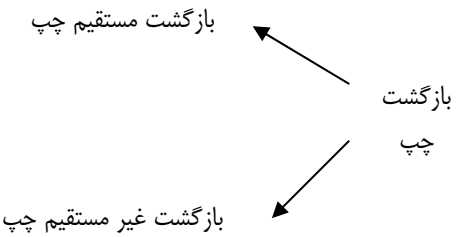
- مثال ۳-۸ گرامر زیر را در نظر بگیرید :

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

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 .



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 [1]

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 .

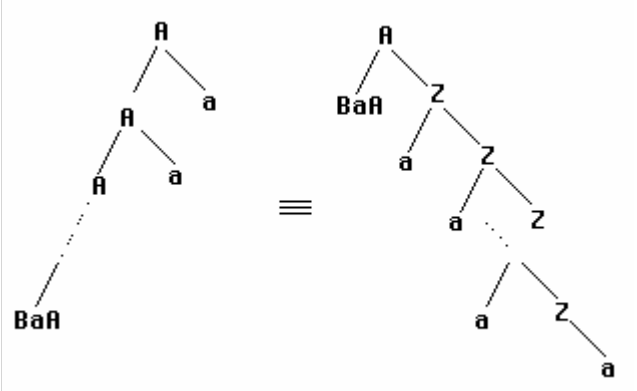
$$A \to A\alpha_1 \mid A\alpha_2 \mid \cdots \mid A\alpha_n$$
$$A \to B_1 \mid B_2 \mid B_3 \mid \cdots \mid B_m$$

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 :

$$A \to B_1 \mid B_2 \mid \cdots \mid B_k$$
$$A \to B_1 Z \mid B_2 Z \mid \cdots \mid B_k z$$
$$Z \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$$
$$Z \to \alpha_1 Z \mid \alpha_2 Z \mid \cdots \mid \alpha_n Z$$

$$A \to Aa$$
$$A \to BaA \mid a \qquad \text{مثلاً}$$
$$\begin{cases} A \to BaA \mid a \\ A \to BaAZ \mid aZ \\ Z \to a \mid aZ \end{cases}$$



[1] 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 $A \to A'\alpha$ اگر شمارهٔ A از شمارهٔ A' کوچکتر باشد $(\#A < \#A')$ در این‌صورت بازگشت چپ وجود ندارد.

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

مثال: در گرامر روبرو

$$\begin{cases} A \to B\alpha \\ B \to C\beta \\ C \to A\delta \end{cases}$$

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)

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

[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.

$$S \to AB$$
$$A \to bBaaA \mid AbaA \mid baA \mid a$$
$$B \to bBa \mid Ab \mid b$$
$$A \to bBaaA \mid AbaAZ \mid baAZ \mid aZ$$
$$Z \to a$$
$$Z \to aZ$$

و بالاخره بازگشت مستقیم از چپ حذف میشود:

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

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

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. $A \to A_1A_2 \quad ; A \prec\succ A_1$
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 $A \to a\alpha \quad ; \alpha \in \left(\sum, V\right)^+$ 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 $A \to A'\alpha \quad ; \alpha \in \left(\sum, V\right)^+$ که $length(\alpha) \underset{=}{>} 2$ 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 $A \to a_1 a_2$ مجموعه قوانین جدید را با استفاده از دو متغیر جدید T_i و T_j تولید میکنیم :

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

۴. به ازای هر قانون به فرم $A \to a_1 A'$ با استفاده از متغیر T_i قوانین جدید جایگزین قوانین فوق میگردند:

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

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.

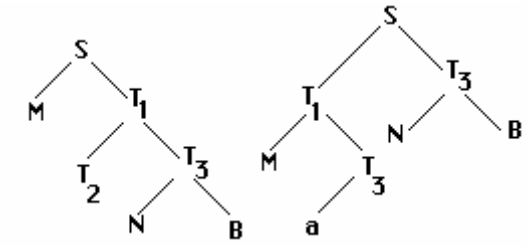
$$\begin{cases} S \to MaNB \\ S \to anb \\ M \to MaB \\ N \to nNBa \\ M \to ma \\ N \to n \\ B \to bBB \\ B \to b \end{cases}$$

حل:

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

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

$$\begin{vmatrix} S \to T_1T_2 \\ T_1 \to MT_3 \\ T_2 \to NB \\ T_3 \to a \end{vmatrix}$$



بروش دیگری نیز میتوان گرامر مستقل از متن را کـه تبـدیلات قبلـی را پـشت سـر گذاشـته اسـت بـه گرامـر چامـــسکی تبـــدیل کـــرد . بـــه ایـــن صـــورت کـــه بـــه ازای هـــر قـــانون غیـــر چامـــسکی بـــه فـــرم $A \to \delta_1\delta_2 \cdots \delta_n \qquad ;\delta_i \in \left(\sum \cup V\right)$ سمت راست قانون را از میان بطور نسبی تجزیه میکنیم یعنی:

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

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

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

در صورتیکه گرامر به روش جدید تولید گردد درخت اشتقاق جملهٔ ω بطول n تقریباً عمقـی برابـر $\lceil \ell og_2^n \rceil$ 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:

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

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:

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

$$R:\begin{cases} A\to a \\ B\to b \\ C\to c \\ S\to XY \\ X\to AD \mid AE \\ D\to XE \\ E\to BB \\ Y\to YC \mid c \end{cases}$$

مثال: فرم نرمال چامسکی زبان زیررا بنویسید:

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

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 $A \Rightarrow aA \Rightarrow aaA \Rightarrow \cdots \Rightarrow a \cdots aA \Rightarrow a^+A \Rightarrow a^+a$

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

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

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

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

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:

$$S \to bA \mid aB$$
$$A \to a \mid aS \mid bAA$$
$$B \to b \mid bS \mid aBB$$

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؟