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 a*b و {a,b,c}* و a+ 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**

هدف دستیابی به تئوری جامع برای تعریف یک زبان:
- الفبای زبان
- ساختار زبان
- روش تولید رشته های زبان (جمله)
- روش ارزیابی تعلق یک رشته به زبان
- روش درک معنا

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[۶] 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
[۴] The Generative Grammar of Modern Hebrew
[۵] Quine
[۶] 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 $u\in (\Sigma\cup V)+$

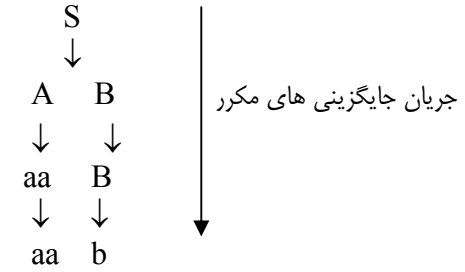
رشته $v\in(\Sigma\cup V)*$

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 :

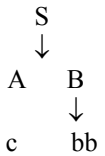
$G=(\Sigma,V,R,S)$; $\Sigma=\{a,b,c\}$; $V=\{S,A,B\}$

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



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



cbb جمله ای از زبان

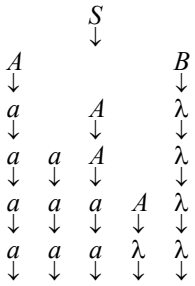
S را بعنوان تمامی رشته ها یا جمله های زبان تعریف می کنیم. جمله های زبان در یک زنجیره از جایگزینی‌های مکرر که بوسیله قوانین جایگزینی هدایت میشوند از S تولید می گردند و قوانین جایگزینی تضمین می کنند که جمله های تولید شده از ساختار خاص زبان تبعیت می کنند. گرامر یک زبان باید بتواند تولید همگی جمله های زبان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

- مثال ۲-۲ : گرامر زبان زیر را طراحی کنید

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

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

R: {S→AB,
A→aA,
A→λ,
B→bB,
B→λ}



- مثال۲-۳ :

R:

< جمله> → < فاعل> <مفعول> را <فعل> .

< مفعول> → < اسم >

< فعل > → خورد

< فعل > → خرید

< فاعل > → < اسم>

< اسم > → کتاب

< اسم > → حسن

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

[1] Recursive Variable

Σ = {حسن، کتاب، خرید، خورد، را ، .}

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

S={< جمله >}

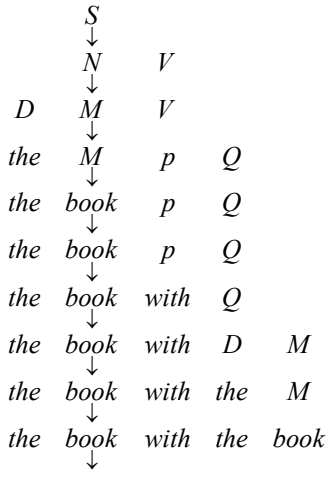
<جمله> → <فاعل> <مفعول> را <فعل> .
<فاعل> <مفعول> را خرید.
<فاعل> <اسم> را خرید.
<فاعل> کتاب را خرید.
<اسم> کتاب را خرید.
حسن کتاب را خرید.

- 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

∑={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}



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. abababbaabab

- مثال ۲-۷ :

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

- $\Sigma=\{a,b,c\}$, $L=(aUb)^*cc(aUb)^*$

$$\begin{cases} S\rightarrow ABA \\ B\rightarrow cc \\ A\rightarrow a|ba \\ A\rightarrow \lambda \end{cases}$$

- $L=\{\omega|Length(\omega)=3\}$, $\Sigma=\{a,b,c\}$

$$\begin{cases} S\rightarrow AAA \\ A\rightarrow a \\ A\rightarrow b \\ A\rightarrow c \end{cases} \equiv A\rightarrow a|b|c$$

- 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 [2] (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[1] (Type_0) T0

---

[1] Chomsky Hierarchy
[2] Regular Language
[3] Context Free Language
[4] Case Sensitive Language

تفاوت این زبانها در ساختارشان است. ساختار زبانهای با قاعده بمراتب ساده تر است از سایر انواع زبانها .بـا توجـه به این نکته که ساختار زبان توسط قواعد جایگزینی بیان میشود بنابراین باید تفاوت این ٤ دسته را در شکل قـوانین جستجوکرد.همهٔ انواع زبان‌های باقاعده، مستقل از متن و و ابسته به متن زبان بدون محدوودیت محـسوب میـشوند 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 . $A \in V, a \in \sum$ $A \rightarrow a$
2 . $A, A' \in V, a \in \sum$ $A \rightarrow aA'$
3 . $S \rightarrow \lambda$ در صورتیکه λمتعلق به زبان باشد

- مثال ۲-۱۰:

$$\text{با قاعده نیست} \begin{cases} S \rightarrow aSb \\ S \rightarrow \lambda \end{cases}$$

- مثال ۲-۱۱ :

$$\text{با قاعده است} \begin{cases} S \rightarrow aS \\ S \rightarrow bS \\ S \rightarrow \lambda \end{cases}$$

- مثال۲-۱۲ :

$$\text{با قاعده نیست} \begin{cases} S \rightarrow Sb \\ S \rightarrow \lambda \end{cases}$$

برای هر زبان با قاعده حداقل یک گرامر با قاعده موجود است . عبارتهای باقاعده معادل با مجموعـه هـای بـا قاعده و زبانهای با قاعده ساختار ساده ای دارند و این ساختار امکان استفاده از ماشین را برای پـردازش زبانهـای بـا قاعده فراهم می سازد.

۲-۲-۲ زبان‌های مستقل از متن

قوانین زبانهای مستقل از متن به شکل زیر هستند:

1 - $A \in V, v \in (\sum \cup V)+$ $A \rightarrow v$
2 - $S \rightarrow \lambda$ 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 :

$$u \rightarrow v$$
$$u,v \in (\sum \cup V)+$$
$$\text{length}(u) \leq \text{length}(v) \quad \text{شرط تکمیلی}$$

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 :

$$u \rightarrow v \qquad \begin{cases} u \in (\sum \cup V)+ \\ v \in (\sum \cup V)+ \end{cases}$$

- **بحث کلی**

زبانهای بدون محدودیت ⊃ زبانهای مستقل از متن ⊃ زبانهای با قاعده

زبانهای بدون محدودیت ⊃ 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.

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

اشتقاق $(1,3,5,6)$

$$\begin{array}{c} S \\ 1\downarrow \\ ABS \\ 3\downarrow \\ aBS \\ 5\downarrow \\ abS \\ 6\downarrow \\ abab \end{array}$$

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 ۲-۱٤ : تعلق پذیری جملهٔ $(a^+b^+)^*ab$ را بررسی کنید.

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

و $S \rightarrow ab$

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

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

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

❑ مثال ۲-۱۵: زبان گرامر $\begin{cases} S \rightarrow aS \\ S \rightarrow \lambda \end{cases}$ چیست ؟ a^*

❑ مثال ۲-۱٦ : زبان گرامر

$$\begin{cases} S \rightarrow AS \\ A \rightarrow aa \\ A \rightarrow bb \\ A \rightarrow ab \\ A \rightarrow ba \\ S \rightarrow \lambda \end{cases}$$

چیست؟

تمامی دنباله های زوجی از ab

((a∪b)(a∪b))*

S→AS→AAS→aabaAS→aababbAS→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

$S \to ASBC$

$\to AASBCBC$

$\to AAASBCBCBC$

$\vdots$

$\to (A)^n S(BC)^n$

$\to (A)^n abc(BC)^n$

$\to a^{n+1}bcBCBC...BC$

$\to a^{n+1}bBcCBCB...BC$

$\to a^{n+1}bbccBCB...BC \to ...$

$\to a^{n+1}bb...bccc...c \to a^{n+1}b^{n+1}c^{n+1}$

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

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:

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

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

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

$$\overline{L} = \{\omega : \omega \in \sum^* \wedge \omega \notin L\} = \sum^* - L$$

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- ۵ 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 [٤] 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

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

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

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
[٤] Polynomial

۲-۵ درخت اشتقاق[۱]

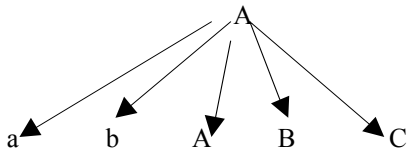
راه دوم نمایش اشتقاق صرف نظر از اینکه کدام قوانین مورد استفاده قرار می گیرند استفاده از درخت اشتقاق است. درخت اشتقاق یک درخت مرتب است که در آن گره ها با سمت چپ قوانین علامت گذاری می‌شوند و بچه های یک گره سمت راست آن قانون هستند و توسط ماشین براحتی قابل نمایش و استفاده می باشند.

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 :

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

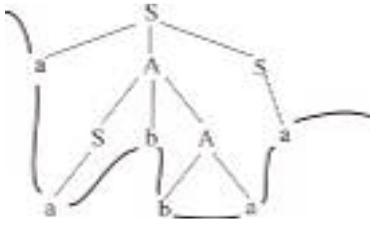
پیمایش برگهای درخت بدون توجه به سطح آنها از چپ به راست .یعنی پیمایش برگهای درخت از چپ براست و مستقل از سطح آنها جمله تولید می شود.

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

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

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

حل:



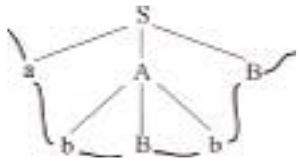
$$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$$

[۱] Derivation Tree

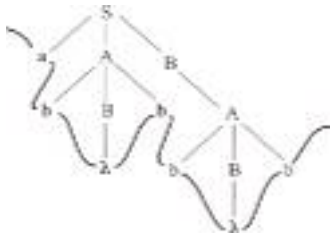
مثال ۲-۲۳: گرامر G با قوانین زیر مفروض است: درخت اشتقاق را برای شبه جملهٔ abBbB رسم کنید و سپس درخت اشتقاق کامل را برای جملهٔ aabbbb رسم کنید.

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

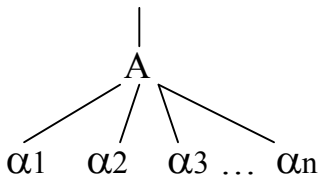
حل:



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 $A\to \alpha1,\alpha2,\ldots,\alpha n$ را انتخاب کرده، زیر درخت زیر تولید میشود:



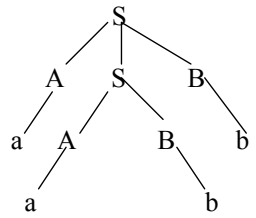
اگر همیشه سمت چپ ترین متغییر بسط داده نشده در درخت اشتقاق (مستقل از سطح) برای بسط انتخاب شود در اینصورت درخت اشتقاق درخت اشتقاق چپ[۱] نام دارد.

حال اگر با اشتقاق راست[۲] برویم ترتیب تولید عناصر جمله ها متفاوت خواهد شد ولی درخت اشتقاق آن کاملاً شبیه درخت اشتقاق چپ خواهد شد.

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

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

[۱] Left Derivation
[۲] 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 :

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

S→ABS و هیچ تفاوتی ندارد که نخست کدام را جایگزین سازیم چون مستقل از متن است.
مثال: گرامر زیر مفروض است:

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

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

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

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

۲-٦ 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.

۲-٦-۱ روش‌های متداول انتخاب متغییر از شبه جمله

در اشتقاق چپ[۱] همواره سمت چپ ترین متغییر در شبه جمله بسط داده می شود.

در **اشتقاق راست**[۲] 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.

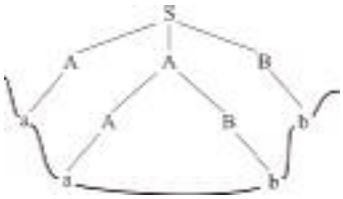
- مثال ۲-۲۵: گرامر زیر مفروض است. اشتقاق راست و چپ را برای جملهٔ aabb نشان دهید و سپس درخت اشتقاق را برای این جمله رسم کنید.

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

حل:

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

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



[۱] Left Most Derivation=L.M.D
[۲] 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 ۲-۲۶ : 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.

۲-۶ ابهام[1] 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.

[1] 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 زیرا 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) $\langle If\ -St\rangle \rightarrow If\ \langle Bexp\rangle then\ \langle St\rangle$

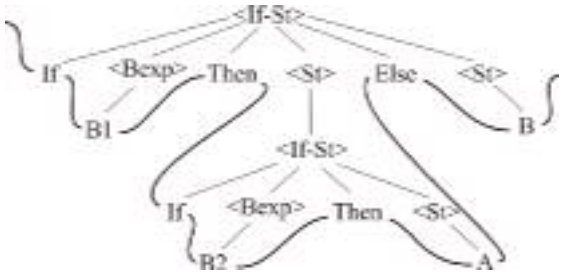
$\langle St\rangle \rightarrow \langle If - St\rangle$

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

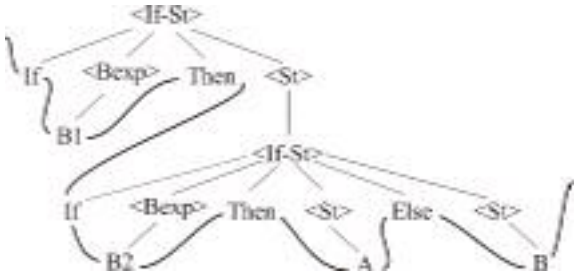
$\langle Bexp\rangle \rightarrow B\,1|B\,2$

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

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

β_1 **پیشوند:** ω

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

ω **پیشوندی است** $\lambda = \beta_1$

$n+1$ **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 کارآ نیست ولی با هر نوع گرامری کار می‌کند.

۲- روش‌های بالا به پائین [۳]

۳- روش‌های پائین به بالا[۳]

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
[۳] 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 [۶] 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

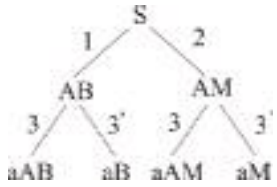
الگوریتم Pars1 گراف پارس را از بالا به پائین یعنی از S به سمت جمله مورد نظر شکل میدهد. پس الگوریتم تولید گراف پارس از بالا به پائین نامیده میشود. از طرف دیگر گراف پارس سطر به سطر شکل میگیرد بنابراین الگوریتم Pars1 تولید کننده گراف پارس در سطح از بالا به پائین است. در این الگوریتم قدیمی ترین گره در گراف 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∈Σ*  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.

---

[۶] Back -tracking
[۷] Predictive
[۸] Greedy



1: $S \to AB$

2: $S \to AM$

3,3′: $A \to aA|a$

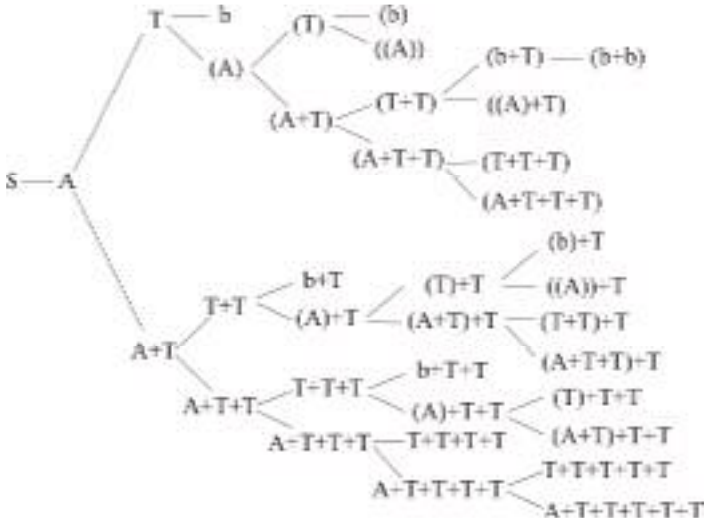
4,4′: $B \to bB|b$

5: $M \to mM|m$

❑ 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.



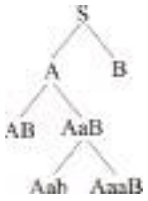
ورودی خوانده شده توسط پارسر	اشتقاق
λ	$S \to A$
λ	$\to T$
(	$\to (A)$
(	$\to (A+T)$
(	$\to (T+T)$
(b +	$\to (b+T)$
(b + b)	$\to (b+b)$

[1] Additive Expression

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

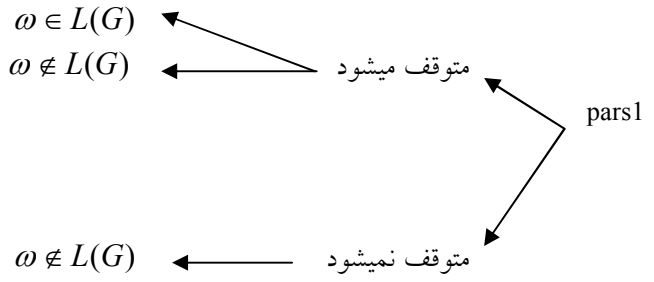
$$\begin{cases} S \to AB \\ A \to Aa \\ A \to a \\ B \to b \end{cases}$$

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:



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

حل:

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

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=ω or Is_empty(st))

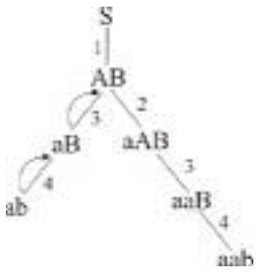
[1] Stack strcture
[2] Top to bottom

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

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

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*



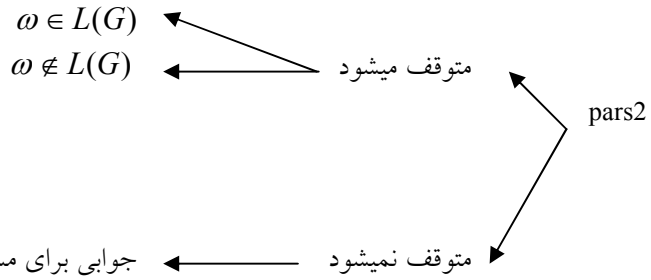
ستون اول بیانگر شبه جمله ها ست که نهایتاً به جمله ω=aab ختم میشود و ستون دوم نشاندهندهٔ شماره قوانین بکار نرفته (یا شمارهٔ قوانین به‌کار رفته) است .

- 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

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

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:



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:

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

۲-۸-۲ تجزیه از پائین به بالا

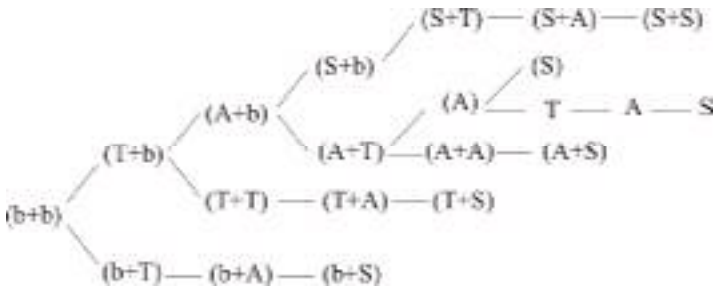
پارسرهای ۱و۲ تولید کنندهٔ گراف از نماد آغاز گر به جمله $(S \overset{*}{\Rightarrow} \omega)$ هستند و در پارسرهائی که در زیر درباره آنها در زیر بحث خواهد شد دیدگاه کمی تغییر پیدا میکند ؛ یعنی فرض براینست که از جمله بخواهیم به‌طور معکوس به نماد آغازگر برسیم.

```
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.

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

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→λ ؛ 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<>λ 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=λ 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

$\alpha_{n-1}\alpha_n$

.

.

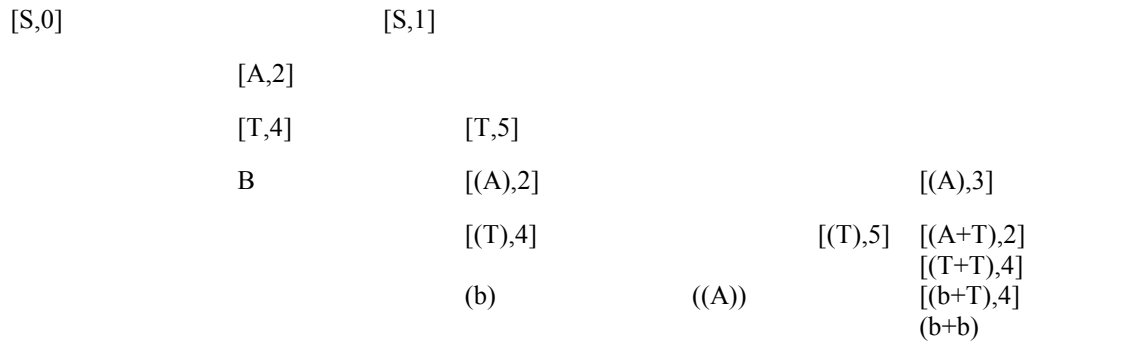
U

❑ 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 ۲-۳٦ فرض کنیم با الگوریتم پارس ٤ و با بکارگیری قوانین مثال ۲-۲۵ بخواهیم اشتقاق (b+b) را بسازیم



اشتقـــاق	پشتـــــه
S⇒A	[S,1]
⇒T	[A,2]
⇒(A)	[T,5]
⇒(A+T)	[(A),3]
⇒(T+T)	[(A+T),2]
⇒(b+T)	[(T+T),4]
⇒(b+b)	[(b+T),4]

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}
Σ= {b, +, (,)}
R :{
1- S→A,
2- A→T,
3- A→A+T,
4- T→b,
5- T→(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	)	Reduction
[(A+b,4,)] [(T,2,+b)] [(b,4,+b)]	(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 محدودیتهائی در زبانهای 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 . این رشته ها را می‌توان مشابهٔ این محدودیت در زبانهای برنامه سازی فرض کرد که تعریف متغیرها باید قبل از استفاده از آنها باشد، به این 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 هائی به شکل $a^+ b^+ c^+ d^+$ ایجاد میکند که در آنها تعداد تکرار a با c برابر است و تعداد تکرار b با d یکی است . این زبان مشابهٔ این محدودیت در زبانها برنامه سازی است که تعداد پارامترها در تعریف یک روال بایستی با 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 زیر قابل توصیف میباشد:

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

- مثال ۲-۴۲ : زبان $L = \{a^n b^n c^m d^m \mid m \geq 1 \wedge n \geq 1\}$ 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 $L = \{a^n b^n \mid n \geq 1\}$ 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 [۱] نام دارد . لم تزریق یکی از قویترین ابزار در تعیین ویژگی منظم بودن زبان است . و همچنین در تکامل الگوریتمهائی وابسته به اتومتای متناهی[۲] 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 $\forall i \geq 0 \quad uv^i w \in L$ (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[۳].

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 تعییین 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 :

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

- 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 .

---

[۱] Pumping Lemma

[۲] 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 :

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

یعنی uv^2w مربع کامل نیست ؛ بنابراین :

$$uv^2w \notin L$$

❑ 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.

۲-۵ 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 بنوییسید 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 .

$$\Sigma = \{a, b, c\}$$
$$V = \{A, B, C\}$$
$$S = A$$
$$R: \begin{cases} A \to aABC \mid abc \\ CB \to BC \\ bB \to bb \\ bC \to bc \\ cC \to cc \end{cases}$$

۲-۱٦ زبان گرامر زیر را بنویسید.

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

۲-۱۷ زبان گرامر زیر را بنویسید.

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

۲-۱۸ گرامر زیر مفروض است :

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

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

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

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

۲-۲۱ گرامر زبان زیر را بنویسید :

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

۲-۲۲ گرامر زبانهای زیر را بنویسید:

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

۲-۲۳ 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

۲-۲۵ 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 .

$$G = (\{a, b\}, \{S, A\}, R, S\}$$
$$R: \begin{vmatrix} S \to aAS \mid a \\ A \to SbA \mid SS \mid ba \end{vmatrix}$$

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.

$$L = \{a^i b^i \mid i \geq 1\}$$

۳۲-۲ زبان گرامر زیر را بنویسید .

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

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

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

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

۳٤-۲ گرامری بنویسید که زبان زیر را تولید کند .

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

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

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

۳٦-۲ گرامری بنویسید که زبان زیر را تولید کند .

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

۳۷-۲ گرامر زبان زیر رابنویسید.

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