فصل سوم

زبان و گرامر

۳-۱ گرامر زبانها

زبانهای برنامه نویسی نیز همانند زبانهای محاوره‌ای مبتنی بر گرامر و قوانین خاص نحوی خود هستند. بـرای بیان قوانین گرامری زبانها از قوانین خاصی به نام Bacus Normal Form استفاده می‌شود، این فـرم کـه در اصطلاح یک Meta Language نامیده می‌شود، اولین برای بیان قوانین گرامری زبان Algoal استفاده شد. برای نمونه ساختار گرامری جملات را می‌توان به صورت زیر بیان کرد:

1) Statement › IfSt | WhileSt | RepeatSt | ForSt | CompoundSt | AssignmentSt | CallSt | CaseSt
2) IfSt › IF Cond THEN Statement ElsePart
3) ElsePart › ELSE Statement | **1**
4) WhileSt › WHILE Cond DO Statement
5) ForSt › FOR ID := Expression TO Expression DO Statement
6) Statement › Statement | Statement ‘:’ Statement
7) Assignment › ID:= Expression
8) CallSt › ID:’(‘ ActualParams ‘)’ | ID
9) CaseSt › CASE Expression OF CaseList END
10) CaseList › CaseLabel ‘:’ Statement | **1**
11) CaseLabel › Constant | CaseLabel , Constant
12) Expression › Expression ‘+’ Term | Expression ‘-‘ Term | Expression OR Term | Term
13) Term › Term ‘*’ Factor | Term ‘/’ Factor | Term AND Factor | Factor
14) Factor › ID | NO | ‘(‘ Expression ‘)’ | NOT Factor
15) Cond › Expression | Expression RelOp Expression
16) RelOp › < | <= | > | >= | <>
17) ActualParams › Expression | ActualParams ‘,’ Expression

برای بیان قواعد گرامر فوق از چهار نوع علامت استفاده شده است:

۱- علامت ‹ به مفهوم **هست** می‌باشد. برای نمونه قاعده شماره (۴) مشخص می‌کند که یک جمله While دارای قاعده گرامری به صورت ارائه شده است.

۲- علامت | به مفهوم **یا** است. برای نمونه (۳) مشخص می‌کند که قسمت ElsePart یا به صورت ELSE Statement است یا اصلاً وجود ندارد که با علامت λ نشان می‌دهند.

۳- علامت لامبدا λ به مفهوم تهی می‌باشد.

۴- علایم پرانتز (و) مهمولاً به عنوان جدا کننده به کار می‌روند. در گرامر فوق چون پرانتز بخشی از گرامر زبان است، لذا آن را در داخل کوتیشن قرار داده‌ایم.

گرامر فوق از تعدادی قاعده و یا در اصطلاح Production یا Rules تشکیل شده است. در سمت چپ هر قاعده یک ترم میانی قرار گرفته و سپس علامت **هست** یا ‹ و سپس گسترش آن ترم میانی در سمت چپ فلش مشخص شده است. برای مثال به قاعده شماره (۶) توجه کنید. در این قاعده Statement سرترم گرامر است. اصولاً سه نوع ترم در داخل گرامر زبانها مشاهده می‌شود:

۱- ترم میانی (Non Terminal):

ترم میانی به آن دسته از گرامرها اطلاق می‌شود که بنا به گرامر زبان دارای تعریف و قاعده باشد و یا به عبارت دیگر در سمت چپ لااقل یک قاعده قرار بگیرد. به ترم میانی، ترم واسطه نیز گفته می‌شود، چرا که واسطه‌ای برای بیان و مشخص نمودن قوانین گرامری است. برای مثال در زبان فارسی جملهٔ **اسنادی** یک ترم میانی و در واقع واسطه‌ای برای بیان دستورالعمل‌های زبان فارسی است. در گرامر فوق ترم‌هایی مثل Expression، Cond و IfSt که در سمت چپ قواعد قرار گرفته‌اند، ترم‌های میانی هستند.

۲- ترم‌های پایانی (Terminal Symbols):

این دسته از ترم‌ها اصولاً در سمت چپ قواعد ظاهر نمی‌شوند و در واقع لغاتی هستند که در داخل زبان ظاهر می‌شوند. کار تشخیص آنها به عهدهٔ تحلیلگر لغوی می‌باشد. از این جمله می‌توان شناسه‌ها، اعداد، جملات تفسیری کامنت و لغات کلیدی زبان را نام برد. برای مثال در گرامر فوق ترم‌های پایانی از قبیل ELSE و FOR با حروف بزرگ مشخص شده‌اند.

۳- سرترم گرامر (Start Symbol):

سرترم یا ترم آغازین گرامرها، ترمی است میانی که شروع کنندهٔ یک گرامر است و نهایتاً در تعریف آن تمامی ترم‌ها به نحوی گنجانده شده‌اند. برای نمونه در گرامر فوق Statement سرترم گرامر است و IfSt سرترم نیست زیرا در تعریف Statement ترمی مانند IfSt یا به طور غیر مستقیم ترمی مانند RelOp وجود دارد اما در تعریف IfSt همواره کلیه جملات باید با کلمه IF آغاز شوند. اصولاً گرامرها یا مستقل از متن هستند یا وابسته به متن می‌باشند.

گرامر فوق مستقل از متن است. به این مفهوم که برای یک ترم میانی مهم نیست در چه متنی آمـده و یـا در کنار کدام ترم میانی ظاهر شود. یک ترم میانی مثل IfSt همیشه دارای ساختاری ثابت بوده و مستقل از متنـی است که در آن ظاهر شده است. اما در گرامرهای وابسته به متن یا Context Sensetive سـاختار یـک تـرم میانی ممکن است وابسته به این که چه ترم‌هایی در اطراف آن قرار گرفته‌اند، فرق کند. برای مثال جمـلات If یا While مستقلاً ساختار خاص خود را دارند. اما اگر جمله If در کنار جمله While ظـاهر شـود، جملـه حـاصل ساختار دیگری خواهد داشت. به همین دلیل است که در گرامرهای وابسته به متن در سمت چپ قواعد ممکن است بیش از یک ترم میانی ظاهر شود.

در گرامرهای مستقل از متن همیشه یک ترم میانی در سمت چپ هر قاعده قرار می‌گیـرد. مسـلماً بـرای هـر گسترش ترم میانی باید یک گسترش وجود داشته باشد. لذا در گرامر وابسته به متن اگر در سمت چپ قـاعده، دو ترم میانی وجود داشته باشد در سمت راست باید تعداد ترم‌ها دو یا بیشتر باشد. اگر تعـداد ترم‌هـا در سـمت چپ قاعده بتواند بیش از تعداد ترم‌ها در سمت راست باشد، آن گرامر را دیگر وابسته به متـن نمی‌نـامند. ایـن نوع گرامر را نوع صفر می‌گویند. نوع دیگر گرامر باقاعده است. در این نوع گرامرها قواعد یا بـه صـورت خـود بازگشتی چپ و یا به صورت خود بازگشتی راست می‌توانند ظاهر شوند. البته قواعد غـیر خـود بـازگشـتی هـم می‌تواند وجود داشته باشد. برای نمونه قاعده شماره (۶) در گرامر فوق خود بازگشتی راست است.

۲-۳ درخت های تجزیه

هدف از ایجاد درخت های تجزیه، تحلیل نحوی جملات است. به عبارت ساده‌تر با ایجاد درخت تجزیه صحت جملات از لحاظ قوانین گرامری مورد آزمون قرار می‌گیرد. این عمل با تجزیه جملات بر اسـاس عنـاصر آنـها صورت می‌پذیرد. برای نمونه گرامر زیر را در نظر بگیرید:

E › E + T | E – T | E OR T | T
T › T * F | T / F | T AND F | F
F › ID | NO. | (E) | NOT F

گرامر فوق مبین ساختار کلی عبارات است، بنابراین هر یک از چهار عمل اصلی باید بر اساس این گرامر ایجاد شده باشند. برای نمونه عبارت زیر را باید بتوان بر اساس گرامر فوق ایجاد نمود.

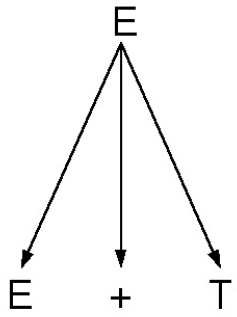
a*(b-c)+d/(e-f)

شاید روش تفکر یک معلم دستورالعمل زبان نیز به طریقی باشد که توضیح داده خواهد شد. یعنی اینکه جملـه نوشته شده را تجزیه می‌کند و مشخص می‌نماید که آیا برای مثال یک جمله انگلیسی هست یا خـیر. معمـولاً ذهن انسان به دو روش عمل می‌نماید. یک روش بالا به پایین است، یعنی اینکه با نگرش به جملهٔ داده شـده

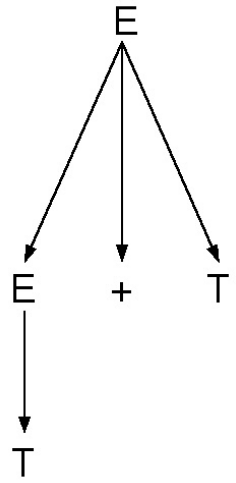
و این نتیجه گیری که جمله باید یک عبارت باشد، عمل آغاز می‌شود. عمل تجزیه از سـرترم گرامـر یعنـی E آغاز می شود. بنابراین در مرحله اول درخت تجزیه بصورت زیر است:

E

حالا با در نظر گرفتن اینکه عبارت داده شده حاصل جمع دو ترم d/(e-f) و a*(b-c) است، سر ترم E بنابر قاعدهٔ E › E + T به صورت زیر گسترش داده می شود:

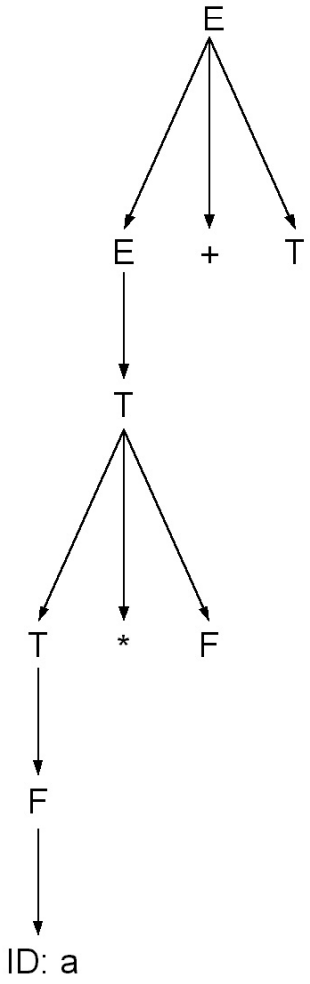


همانگونه که در بالا توضیح داده شد، عبارت داده شده حاصل جمع دو ترم است لذا بنابر قاعده E› T ترم میانی E به T گسترش داده می شود.

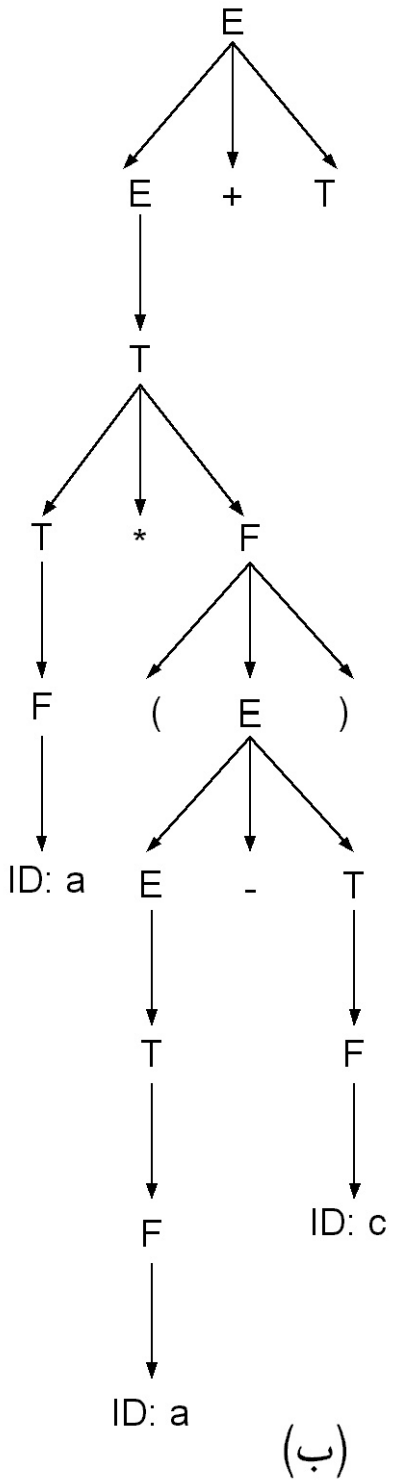


اکنون باید بتوان از ترم میانی T عبارت a*(b-c) را تولید کرد لذا، درخت تجزیه فوق بصورت (الف) توسعه داده می‌شود.

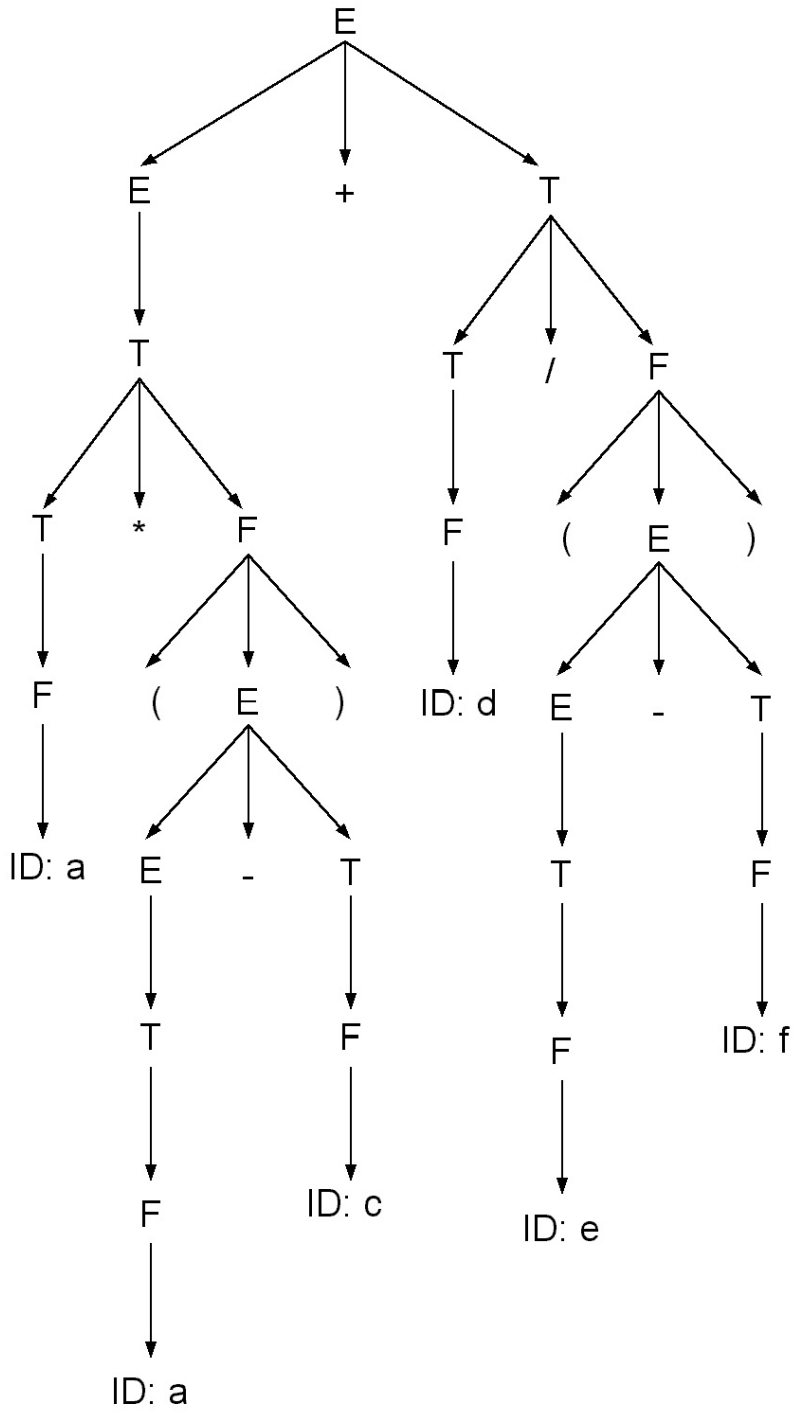
اکنون باید بتوان F را با (b-c) جایگزین کرد. بنابراین درخت تجزیه فوق بصورت (ب) گسترش داده می‌شود. اکنون با توجه به اینکه پس از علامت جمع باید از ترم میانیT نهایتاً عبارت d/(e-f) حاصل شود درخت تجزیه به صورت (ج) توسعه داده می‌شود.



(الف)



(ب)



(ج)

همانگونه که در شکلهای فوق مشاهده می‌شود با استفاده از روش بالا به پایین و تجزیه یک عبارت بـر طبـق گرامر نهایتاً درخت تجزیه که در رأس آن سرترم گرامر و در برگها ترم‌های پایانی قرار دارند، ایجاد شده است.

چنانچه عبارت از لحاظ گرامری غلط می‌بود، این امکان وجود نداشت که بتوان بر اساس گرامر درخت فـوق را ایجاد کرد. مراحل تجزیه با ایجاد مرحله به مرحله درخت نمایش داده شد. زیرا با توجه به درخت تجزیه نـهایی نمی‌توان مراحل تجزیه را مشخص نمود. در طی مراحل ایجاد درخت تجزیه، عمل تجزیه ترم های میـانی از چپ به راست انجام شده، همواره سمت چپ‌ترین عنصر یا گره در داخل درخت گسـترش داده شـد، تـا نـهایتاً بتوان به یک ترم پایانی بعدی در داخل جمله داده شده از سر ترم گرامر مشتق یـا نتیجـه گیـری شـده اسـت. مراحل تجزیه سر ترم تا رسیدن به ترم های پایانی درون جمله داده شـده را در اصطـلاح مراحـل اشـتقاق یـا Derivation می‌گویند. مراحل اشتقاق سر ترم E تا رسیدن به جمله فوق به صورت زیر است:

E› E+T› T+T› T*F+T› F*F+T› a*F+T› a*(E)+T›
A*(E−T)+T› a*(T−T)+T› a*(T−T)+T› a*(F−T)+T›
a*(b−T)+T› A*(b−F)+T› a*(b−c)+T› a*(b−c)+T/F› a
(b−c)+F/F› A(b−c)+d/F› a*(b−c)+d/(E)› a
(b−c)+d/(E−T)› A(b−c)+d/(T−T)› a
(b−c)+d/(F−T)› a(b−c)+d/(e−T)› A
(b−c)+d/(e−F)› a(b−c)+d/(e−f)

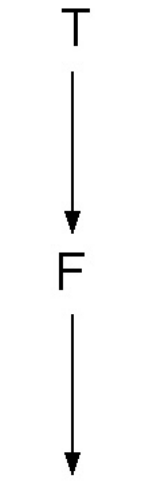
همانگونه که مشاهده نمودید، عمل اشتقاق از سرترم آغاز و نهایتاً به جمله داده شده خاتمه می یـابد. در ایـن میان فرمهای جمله‌ای که حاوی ترمهای میانی و پایانی و یا فقـط ترم‌هـای میـانی هسـتند، ایجـاد مـی‌گردد. فرمهای جمله‌ای را Sentential form نیز می‌گویند.

٤-٣ تجزیه پایین به بالا

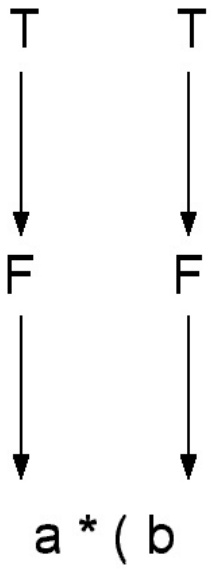
در روش تجزیه پایین به بالا برخلاف روش پایین به بالا، عمل تجزیه از پایین و از ترم‌های درون جمله آغاز و به سرترم گرامر خاتمه می یابد. لذا این روش پایین به بالا است. برای نمونه عبارت زیر را در نظر میگیریم:
a*(b-c)+d
هدف تجزیه این عبارت به روش پایین به بالا است. همانگونه که توضیح داده شد در روش پایین به بالا عمل تجزیه از چپ به راست با جایگزینی ترم‌های پایانی با میانی بر اساس سمت چپ قواعد انجام می شود و آنقـدر عمل جایگزینی ادامه می‌یابد تا در صورت صحت، بتوان به سرترم داده شده رسید.
ابتدا سمت چپ‌ترین لغت از عبارت داده شده یعنی a توسط تحلیلگر لغـوی خوانـده می‌شـود. بـر سـر ورودی علامت * ظاهر می شود. طبق قاعده T*F ›T، قبل از عملکرد * فقط یک T می تواند ظاهر شود . لذا ، با استفاده از قاعده F› id ترم پایانی a که یک شناسه یا id است را با F و سپس طبق قـاعده T› F تـرم میانی F را با Tمی‌توان جایگزین نمود. اکنون ترم بعدی یعنی * از سر ورودی خوانـده می‌شـود. طبـق قـاعده T*F ›T پس از * ترم میانی F می‌تواند ظاهر شود. لذا، ترم بعدی بایستی F باشد.

اکنون ورودی ’(b-c)+d‘ است. بر سر ورودی لغت) وجود دارد. پس از خواندن) و b درخت بصورت (الف) خواهد بود.

حال بر سر ورودی لغت منها وجود دارد. طبق قاعده E› E-T قبل از عملگر – باید یک عبارت E وجود داشته باشد. لذا ترم پایانی b که یک شناسه یا id است. بنابر قاعده id ‹ F با ترم میانی F و سپس طبـق قواعـدT F ‹ و T ‹ E نهایتاً با ترم میانی E جایگزین می‌شود. بنابراین، تا این مرحله درخت تجزیه پایین به بالا بـه صورت (ب) خواهد بود.

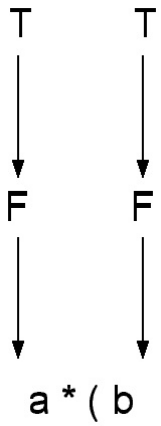


(الف)

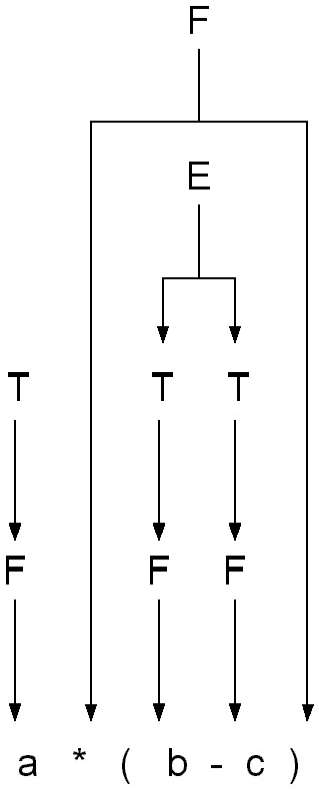


(ب)

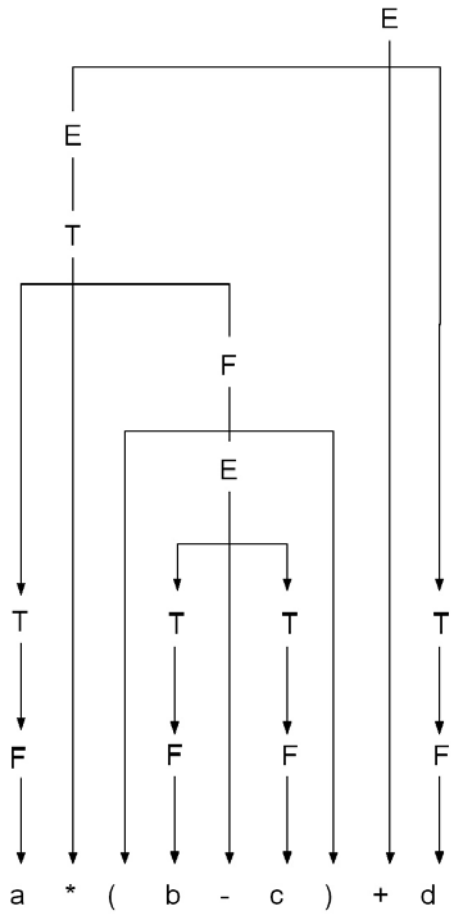
اکنون ورودی d + (c – است. عملگرهای – و سپس c از ورودی خوانده می‌شود. طبق قـاعده E› E-T باید پس از – در ورودی یک ترم ظاهر شود. لذا، c با F وF با T و نهایتاً T – E با E جایگزیـن خواهنـد شـد. حال (از ورودی خوانده می‌شود و به این ترتیب در داخل درخت تجزیه (E) ظاهر می‌شود و طبق قاعده F (E) n می‌توان آن را با F جایگزین کرد.



اکنون ورودی ’d+(c-‘ است. عملگرهای – و سپس c از ورودی خوانده می‌شوند. طبق قاعدهE-T ‹ E باید پس از - در ورودی یک ترم ظاهر شود. لذا c با F و F با T و نهایتاً E-T با E جایگزین خواهند شد. حال (از ورودی خوانده می‌شود و به این ترتیب در داخل درخت تجزیه (E) ظاهر می‌شود و طبق قاعدهٔ (E) ‹ F می‌توان آن را با F جایگزین کرد.



اکنون ورودی ’d+‘ است. بر سر ورودی علامت + وجود دارد. قبل از + طبق قاعده E › E + T باید یک E وجود داشته باشد. بنابراین T * F را که اکنون در رأس درخت تجزیه وجود دارد طبق قاعده T › T * F با T و سپس طبق قاعده E › T ترم T را با E می‌توان جایگزین نمود. نهایتاً درخت تجزیه به صورت زیر



تبدیل می‌شود:

همانگونه که مشاهده نمودید عمل تجزیه از ترمهای پایانی آغاز و به ترم گرامر خاتمه یافت. این گونه تجزیه را در اصطلاح تجزیه پایین به بالا یا Bottom Up Parsing گویند.

طبق تعریف، مراحل اشتقاق نمایانگر مراحل رسیدن از سرترم گرامر به ترم‌های پایانی درون جمله مورد نظراست، در تجزیه پایین به بالا، مراحل اشتقاق درست برخلاف مراحل تجزیه است چرا که در اینجا نهایتاً به سر ترم گرامر می رسند در صورتی که مراحل اشتقاق از سر ترم گرامر آغاز می‌شود. به عبارت دیگر مراحل اشتقاق نشان می‌دهد که در طی چه مراحلی جمله داده شده از سر ترم گرامر مشتق شده است. مراحل مشتق کردن a*(b-c)+d از سرترم به صورت زیراست. باید توجه داشته باشید که مراحل اشتقاق درست برخلاف مراحل تجزیه است. بنابراین اگر از آخرین مرحله ایجاد درخت تجزیه به مراحل قبلی برگشت شود مراحل اشتقاق بصورت زیرمعین می شود :

E › E+T › E+F › E+d › T+d › T*F+d › T*(E)+d › T*(E-T)+d ›
T*(E-F)+d › T*(E-c)+d › T*(E-c)+d › T*(T-c)+d › T*(F-c)+d ›
T*(b-c)+d › F*(b-c)+d › a*(b-c)+d

همانگونه که مشاهده می‌شود در طی مراحل اشتقاق برخلاف اشتقاق چپ که قبلاً توضیح داده شد، همواره سمت راست‌ترین ترم‌ها تا رسیدن به ترم پایانی درون جمله داده شده جایگزین می‌گردند. لذا این گونه اشتقاق را **اشتقاق راست** گویند. اولین ترم پایانی که در طی این مراحل اشتقاق درفرمهای جمله‌ای ظاهر شد ترم پایانی d در فرم جمله‌ای T+d بود و سپس از راست به چپ ترم پایانی بعدی یعنی c درفرم جمله ای T* (E +d) –c) ظاهرگردید لذا همانگونه که مشاهده می‌کنید جایگزینی ازسمت راست به چپ انجام شده است.

۳-۵ گرامرهای مبهم

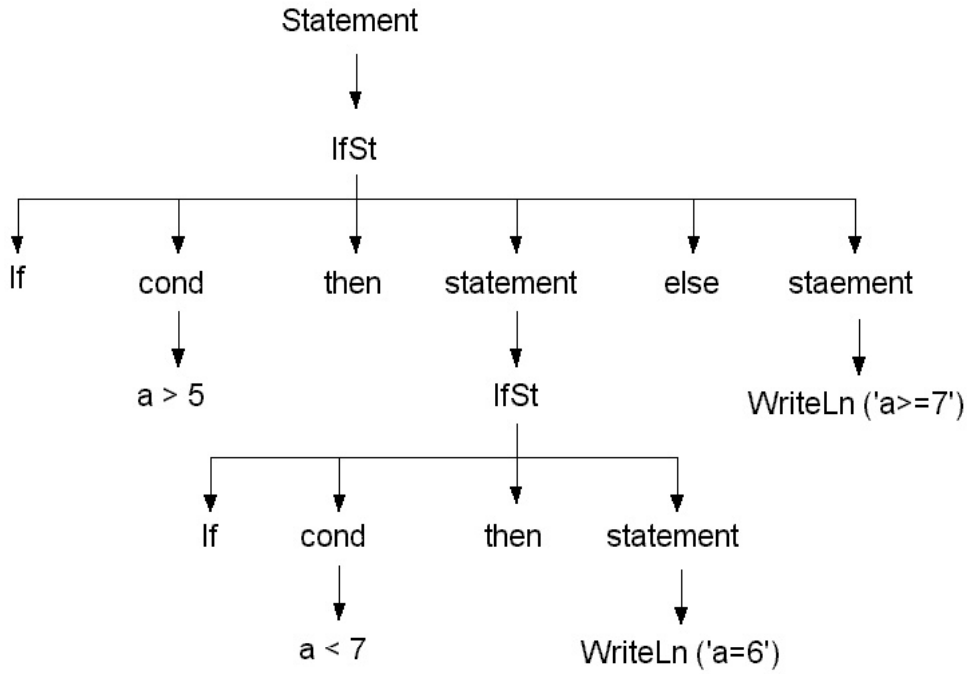
چنانچه بتوان برای جمله داده شده براساس گرامر زبان، بیش ازیک درخت تجزیه ایجاد نمود، آن را مبهم می‌نامند. در حالت کلی با نگرش به گرامرها نمی‌توان ابهام را تشخیص داد. اما باید دید که چرا. اگربتوان بیش ازیک درخت تجزیه برای جمله داده شده ایجاد نمود، گرامر ابهام دارد. این باید یک مزیت باشد، چرا آن را ابهام می‌خوانند؟ به عنوان یک مثال ساده به گرامر زبانهای C یا پاسکال برای جملات شرطی if توجه نماید:

Statement › IfSt | WhileSt | CompoundSt | AssignmentSt | CallSt
IfSt › IF Cond THEN statement | IF Cond THEN Statement ELSE Statement
Cond › Exp | Exp Relop Exp

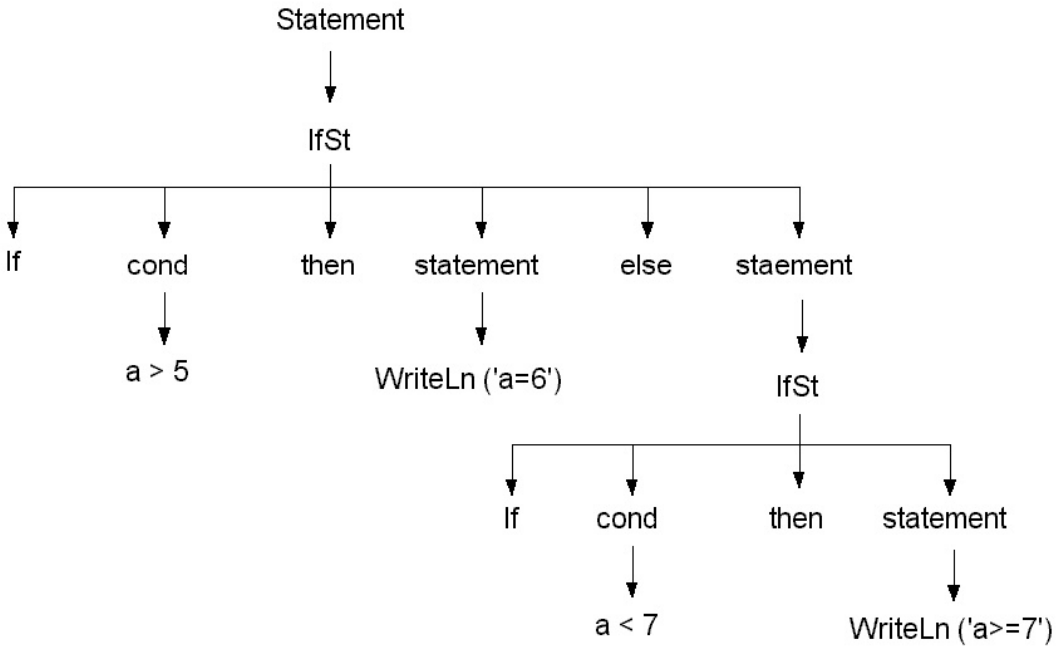
برای نمونه به جمله زیر توجه نمایید:

```
IF a>5 THEN IF a<7 THEN WriteLn ('a=6')
     ELSE WriteLn ('a>=7')
```

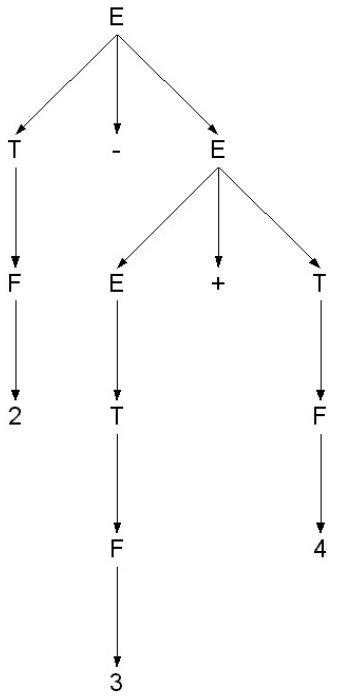
بنابراین گرامر فوق درخت تجزیه بصورت زیر می تواند باشد:



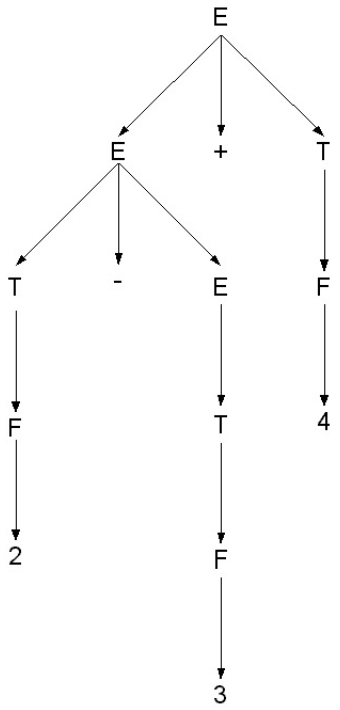
درخت تجزیه را به صورت زیر نیز می‌توان تولید کرد:



می‌خواهیم برای جملهٔ 4+3-2 درخت تجزیه ترسیم نماییم:



درخت با حاصل 2-(3+4) = -5



درخت با حاصل (2-3)+4 = 3

همانگونه که مشاهده می‌کنید ایجاد دو نتیجه متفاوت ۵- و ۳ برای عبارات فوق به خاطر مبهم بودن گرامر و در واقع قاعده به صورت خودبازگشتی چپ و خودبازگشتی راست برای ترم میانی E است.

بنابرگرامر If دو درخت تجزیه با دو مفهوم متفاوت ایجاد شده است. در درخت تجزیه شکل اول، Else وابسته به If خارجی می باشد و درشکل دوم، Else وابسته بهIf داخلی است. برای رفع این مشکل درزبان‌های C و پاسکال، برای تکمیل گرامر بطور ضمنی و نه بر طبق گرامر، تعیین کرده‌اند که Else به نزدیکترین If وابسته است. بنابراین درخت تجزیه شکل دوم توسط کامپایلر پاسکال ایجاد می‌شود. درزبان فورترن ۷۷ با افزایش Endif این مشکل حل شده و گرامر به صورت زیر تبدیل گردیده است:

If › IF Cond THEN Statement ENDIF | IF Cond THEN Statement ELSE Statement ENDIF

بنابراین اگر قرار است که در جمله یادشده Else وابسته به If خارجی باشد، به صورت زیر عمل می‌شود:

```
If a>5 THEN If a<7 THEN WriteLn('a=6') ENDIF ELSE
WriteLn('a>=7') ENDIF
```

بالعکس اگر Else وابسته به If داخلی باشد، جمله به صورت زیر نوشته می‌شود:

```
If a>5 THEN If a<7 THEN WriteLn('a=6') ELSE
WriteLn('a>=7') ENDIF
```

اصولاً در حالت کلی، ابهام در گرامرها را نمی‌توان نشان داد و اثبات نمود، اما نکته قابل توجه این است که اگر گرامری به صورت خودبازگشتی چپ و خودبازگشتی راست برای یک ترم میانی قواعدی داشته باشد، آن گرامر مبهم است.

1) A › Aa | bA

همچنین اگر در یک گرامر نهایتاً بتوان طی مراحل استنتاج یا اشتقاق از یک ترم میانی به خود آن ترم رسید، گرامر مبهم می‌باشد.

2) A › a › . . . . › A

اگر در گرامری قواعد به صورت خودبازگشتی راست یا چپ برای یک ترم میانی وجود داشته باشد و علاوه بر آن قاعده دیگر برای این ترم میانی به صورتی باشد که حالت خودبازگشتی در دو سطح دو ترم در سمت راست آن قاعده برای آن ترم میانی وجود داشته باشد و در اصطلاح قاعده به صورت Self Embeding یا خودگردان باشد، باز هم گرامر مبهم خواهد بود.

3) A › Aa | bAd

برای نمونه به گرامر مبهم عبارات توجه کنید:

E › E + T | T – E | T
T › T * F | T / F | F
F › ID | NO. | (E)

۶-۳ تمرین

تمرین ۱- مراحل استنتاج را برای عبارت (a*b-c)/(d-e) در روش‌های تجزیه بالا به پایین و پایین به بالا مشخص نمایید.

تمرین ۲- گرامر ساختار لیست را در نظر بگیرید،

S › '(' L ')' | a
L › L ',' S | S

درخت تجزیه را برای جملات زیر ترسیم نمایید:

الف – (a , a)

ب - (a , (a , a))

ج - (a , ((a , a) , (a , a)))

تمرین ۳- نشان دهید که گرامر زیر مبهم است.

S › aSbS › bSaS | l

تمرین ۴- گرامر زیر را در نظر بگیرید :

bexpr › bexpr or bterm › bterm
bterm › bterm and bfactor | bfactor
bfactor › not bfactor | (bexpr) | true | false

الف ـ درخت تجزیه برای عبارت not (true or false) ایجاد کنید.

ب ـ آیا این گرامر مبهم است؟

تمرین ۵ ـ چرا اگر در یک گرامر برای یک ترم میانی قواعد به دو صورت خود باز گشتی چپ و خود بازگشتی راست وجود داشته باشد آن گرامر مبهم است.