فصل سوم

زبان وگرامرزبان

3.1 - گرامر زبانها :

زبانهاي برنامه سازي نيز همانند زبانهاي محاوره اي مبتني بر گرامر و قوانين خاص نحوي خود هستند . براي بيان قوانين گرامري زبانها از قوانين خاصي بنام **Bacus Normal Form** استفاده مي شود. اين فرم كه دراصطلاح يك **Meta Language** ناميده ميشود ، اولين بار براي بيان قوانين گرامري زبان **Algoal** استفاده شد . براي نمونه ساختارگرامري جملات را مي توان بصورت زير بيان كرد:

1) Statement ⟶ IfSt | WhileSt | RepeatSt | ForSt | CompoundSt | AssignmentSt | callst | CaseSt
2) IfSt ⟶ IF cond THEN statement ELSE part
3) ElsePart ⟶ ELSE statement | ε
4) WhileSt ⟶ WHILE cond DO statement
5) ForSt ⟶ FOR id := expression TO expression DO statement
6) CompoundSt ⟶ BEGIN statements END.
7) statements ⟶ statement | statements ‘;’ statement
8) AssignmentSt ⟶ id := expression
9) CallSt ⟶ id ‘(‘ ActualParams ‘)’ | id
10) CaseSt ⟶ CASE expression OF CaseList End
11) CaseList ⟶ CaseLabel ‘:’ Statement | ε
12) CaseLabel ⟶ Constant | CaseLabel , Constant
10) expression ⟶ expression ‘+’ term | expression ‘-’ term | expression Or term | term
11) term ⟶ term ‘*‘ factor | term ‘/’ factor | term And factor | factor
12) factor ⟶ id | No | ‘(‘ expression ‘)’ | Not factor
13) cond ⟶ expression | expression relop expression
14) relop ⟶ < | <= | > | >= | <>
15)ActualParams ⟶ expression | ActualParams ‘,’ expression

براي بيان قواعد گرامر فوق از چهار نوع علامت استفاده شده است :
1- علامت " ⟶ " به مفهوم " هست " ميباشد. براي نمونه قاعده شماره (4) مشخص ميكند كه يك جمله While داراي قاعده گرامري بصورت ارائه شده است .

2- علامت "|" به مفهوم "يا" است . براي نمونه (3) مشخص ميكند كه قسمت ELSE يا else part يا بصورت Else statment است و يا ا صلاً وجود ندارد كه با علامت اپسيلون ε نشان ميدهند .

3- علامت اپسيلون "ε" به مفهوم تهي ميباشد .

4- علائم پرانتز "(" و ")" معمولاً بعنوان جداكننده استفاده ميشوند. درگرامر فوق چون پرانتز بخشي از گرامر زبان ميباشد لذا ، آنرا در داخل كوتيشن قرار داده اند .

گرامر فوق از تعدادي قاعده و يا در اصطلاح Productions يا Rules تشكيل شده است. در سمت راست هر قاعده يك ترم مياني قرار گرفته و سپس علامت هست يا "→" و سپس گسترش آن ترم مياني در سمت چپ فلش مشخص شده است . براي مثال به قاعده شماره (7) توجه كنيد. دراين قاعده Statment سرترم گرامر است. اصولاً سه نوع ترم در داخل گرامر زبانها مشاهده ميشود .

1- ترم مياني (Non terminal) :

ترم مياني به آن دسته از ترمها اطلاق ميشود كه بنا به گرامر زبان داراي تعريف و قاعده باشد و يا بعبارت ديگر در سمت چپ لااقل يك قاعده قرار بگيرد. به ترم مياني ، ترم واسطه نيز گفته مي شود چرا كه واسطه اي براي بيان و مشخص نمودن قوانين گرامري است. براي مثال در زبان فارسي جمله ً اسناديً يك ترم مياني و در واقع واسطه اي براي بيان دستورالعمل هاي زبان فارسي است. در گرامر فوق ترمهايي مثل Cond , Expression و If st كه در سمت چپ قواعد قرار گرفته اند، ترمهاي مياني هستند.

2- ترمهاي پاياني (Terminal seymbols) :

اين دسته از ترمها اصولاً در سمت چپ قواعد ظاهر نمي شوند و در واقع لغاتي هستند كه در داخل زبان ظاهر مي شوند . كار تشخيص آنها به عهده تحليلگر لغوي ميباشد. از اين جمله مي توان شناسه ها، اعداد، جملات تفسيري كامنت و لغات كليدي زبان را نام برد. براي مثال در گرامر فوق ترمهاي پاياني از قبيل ELSE و For با حروف بزرگ مشخص شده اند.

2- **سر ترم گرامر (Start Seymbol) :**

سر ترم يا ترم آغازين گرامرها، ترمي است مياني كه شروع كننده يك گرامراست و نهايتاً در تعريف آن تمامي ترمها به نحوي گنجانده شده اند. براي نمونه درگرامر فوق Statement سر ترم گرامر است و IfSt سرترم نيست زيرا ، در تعريف Statement ترمي مانند IfSt يا بطور غير مستقيم ترمي مانند Relop وجود دارد اما در تعريف Ifst همواره كليه جملات بايد با كلمه If آغاز شوند. اصولاً گرامر ها يامستقل از متن[1] هستند و يا وابسته به متن مي باشند .

گرامرفوق مستقل ازمتن است. به اين مفهوم كه براي يك ترم مياني مهم نيست در چه متني و يا دركنار كدام ترم مياني ظاهر شود. يك ترم مياني مثل IfSt هميشه داراي ساختاري ثابت بوده ، مستقل از متني كه در آن ظاهر شده ، مي باشد. امّا، درگرامرهاي وابسته به متن يا Context sensetive ساختار يك ترم مياني ممكن است وابسته به اين كه چه ترم هايي در اطراف آن قرارگرفته اند ، فرق كند. براي مثال جملات If يا While مستقلاً ساختار خود را دارند. امّا، اگر جمله If درکنار جمله While ظاهر شود جمله حاصل ساختار ديگري خواهد داشت. به همين دليل است كه در گرامرهاي وابسته به متن در سمت چپ قواعد، ممكن است بيش از يك ترم مياني ظاهر شود.

درگرامر هاي مستقل از متن هميشه يك ترم مياني در سمت چپ هر قاعده قرار ميگيرد. مسلماً براي هر ترم مياني يك گسترش مي بايست وجود داشته باشد. لذا ، درگرامر وابسته به متن اگر در سمت چپ قاعده ، دو ترم مياني وجود داشته باشد در سمت راست بايد تعداد ترمها دو يا بيشتر باشد. اگر تعداد ترمها در سمت چپ قاعده بتواند بيش از تعداد ترم ها در سمت راست باشد آن گرامر را ديگر وابسته به متن نمي نامند. اين نوع گرامر را *نوع صفر* مي گويند .

نوع ديگر *گرامر باقاعده* است. در اين نوع گرامرها قواعد يا بصورت خود بازگشتي چپ و يا بصورت خود بازگشتي راست ميتوانند ظاهر شوند. البته قواعد غيرخودبازگشتي هم مي تواند وجود داشته باشد. براي نمونه قاعده شماره (7) درگرامر فوق خود بازگشتي راست است.

3.2 - درختهاي تجزيه :

هدف از ايجاد درختهاي تجزيه تحليل نحوي جملات است . به عبارت ساده تر با ايجاد درخت تجزيه صحت جملات از لحاظ قوانين گرامري مورد آزمون قرار

[1] مستقل از متن يا Context Free به گرامري اطلاق مي شود كه ساختار نحوي هر ترم مياني مستقل از متن دربر گيرنده آن است.

ميگيرد . اين عمل با تجزيه جملات براساس عناصر آنها صورت ميپذيرد . براي نمونه گرامر زير را در نظر بگيريد :

$E \rightarrow E + T \mid E - T \mid E \text{ OR } T \mid T$
$E \rightarrow T * F \mid T / F \mid T \text{ AND } F \mid F$
$F \rightarrow id \mid NO \mid (E) \mid \text{NOT } F$

گرامر فوق مبين ساختار كلي عبارات است ، بنابراين هر يك از عبارات چهار عمل اصلي بايد بر اساس اين گرامر ايجاد شده باشند. براي نمونه عبارت زير را بايد بتوان بر اساس گرامر فوق ايجاد نمود.

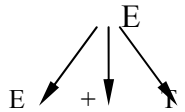
$a * (b - c) + d / (e - f)$

شايد روش تفكر يك معلم دستورالعمل زبان نيز به همين طريقي باشد كه توضيح داده خواهد شد. يعني اينكه جمله نوشته شده را تجزيه ميكند و مشخص مينمايد كه آيا براي مثال يك جمله انگليسي هست يا خير. معمولاً ذهن انسان به دو روش عمل مي نمايد. يك روش بالا به پائين است ، يعني اينكه با نگرش به جمله داده شده و اين نتيجه گيري كه جمله بايد يك عبارت باشد، عمل آغاز مي شود. عمل تجزيه از سر ترم گرامر يعني E آغاز مي شود. بناربراين در مرحله اول درخت تجزيه بصورت زير است :

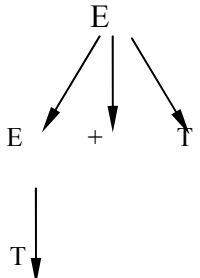
E

درخت تجزيه در مرحله 1

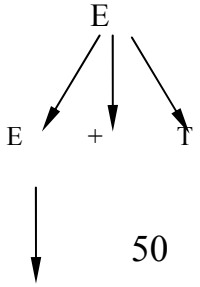
حالا با در نظر گرفتن اينكه عبارت داده شده حاصل جمع دو ترم ($a * (b - c)$ و $d / (e - f)$ است، سر ترم E بنابر قاعده $E \rightarrow E + T$ بصورت زير گسترش داده ميشود.

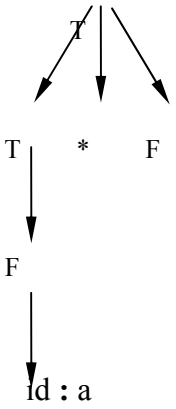


همانگونه كه در بالا توضيح داده شد عبارت داده شده حاصل جمع دو ترم است لذا ، بنا بر قاعده $E \rightarrow T$ ترم مياني E به T گسترش داده مي شود.

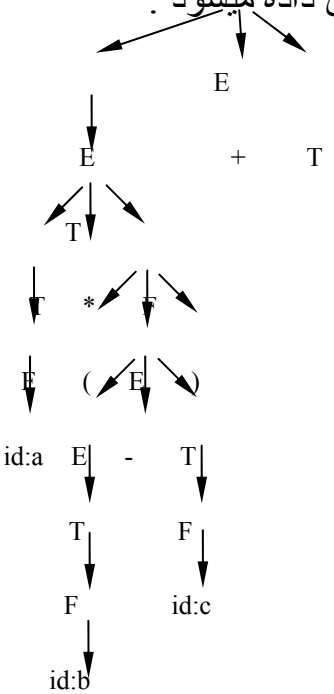


اكنون بايد بتوان از ترم مياني T عبارت $a * (b - c)$ را توليد كرد لذا ، درخت تجزيه فوق بصورت زير نوسعه داده ميشود.

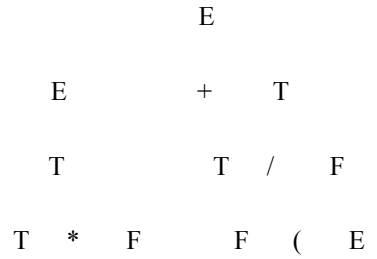


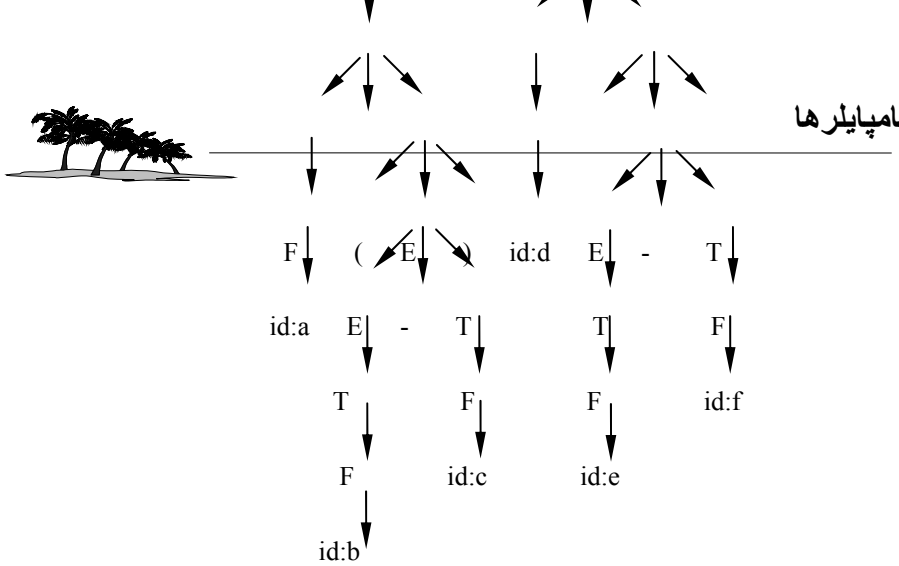


اکنون باید بتوان F را با (b - c) جایگزین کرد. بنابراین در خت تجزیه فوق بصورت زیر گسترش داده میشود :



اکنون با توجه به اینکه پس از علامت جمع باید از ترم میانی T نهایتاً عبارت d / (e - f) حاصل شود درخت تجزیه بصورت زیر توسعه داده میشود:





همانگونه كه در شكلهاي فوق مشاهده ميشود با استفاده از روش بالا به پائين و تجزيه يك عبارت بر طبق گرامر نهايتاً درخت تجزيه كه در رأس آن سر ترم گرامر و در برگها ترمهاي پاياني قرار دارند ايجاد شده است .
چنانچه عبارت از لحاظ گرامري غلط مي بود، اين امكان وجود نداشت كه بتوان بر اساس گرامر درخت فوق را ايجاد كرد. مراحل تجزيه با ايجاد مرحله به مرحله درخت نمايش داده شد زيرا، با توجه به درخت تجزيه نهايي نمي توان مراحل تجزيه را مشخص نمود. در طي مراحل ايجاد درخت تجزيه ، عمل تجزيه ترمهاي مياني از چپ به راست انجام شده ، همواره سمت چپ ترين عنصر يا گره در داخل درخت گسترش داده شد تا نهايتاً بتوان به يك ترم پاياني بعدي در داخل جمله داده شده رسيد. به عبارت ديگر درخت تجزيه مشخص نميكند كه در طي چه مراحلي جمله داده شده از سر ترم گرامر مشتق يا نتيجه گيري شده است . مراحل تجزيه سر ترم تا رسيدن به ترمهاي پاياني درون جمله داده شده را در اصطلاح مراحل اشتقاق يا Deriivation مي گويند. مراحل اشتقاق سر ترم E تا رسيدن به جمله فوق به صورت زير است :

$$E \Rightarrow E+T \Rightarrow T+T \Rightarrow T*F+T \Rightarrow F*F+T \Rightarrow a*F+T \Rightarrow a*(E)+T \Rightarrow$$
$$a*(E-T)+T \Rightarrow a*(T-T)+T \Rightarrow a*(F-T)+T \Rightarrow a*(b-T)+T \Rightarrow$$
$$a*(b-F)+T \Rightarrow a*(b-c)+T \Rightarrow a*(b-c)+T/F \Rightarrow a*(b-c)+F/F \Rightarrow$$
$$a*(b-c)+d/F \Rightarrow a*(b-c)+d/(E) \Rightarrow a*(b-c)+d/(E-T) \Rightarrow$$
$$a*(b-c)+d/(T-T) \Rightarrow a*(b-c)+d/(F-T) \Rightarrow a*(b-c)+d/(e-T) \Rightarrow$$
$$a*(b-c)+d/(e-F) \Rightarrow \mathbf{a*(b-c)+d/(e-f)}$$

همانگونه كه مشاهده نموديد، عمل اشتقاق از سر ترم آغاز و نهايتا به جمله داده شده خاتمه مي يابد. در اين ميان فرمهاي جمله أي كه حاوي ترمهاي مياني و پاياني و يا فقط ترمهاي مياني هستند، ايجاد مي‌گردد. فرمهاي جمله اي را **Sentential form** نيز ميگويند .

3.4 تجزيه پائين به بالا

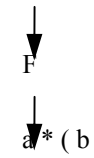
در روش تجزيه پائين به بالا بر خلاف روش بالا به پائين ، عمل تجزيه از پائين و از ترمهاي درون جمله آغاز و به سر ترم گرامر خاتمه مي يابد. لذا ، اين روش پائين به بالا است. براي نمونه عبارت زير را در نظر ميگيريم :

a * (b - c) + d

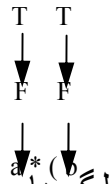
هدف تجزيه اين عبارت به روش پائين به بالا است . همانگونه كه توضيح داده شد در روش پائين به بالا عمل تجزيه از چپ به راست با جايگزينی ترمهاي پاياني با مياني بر اساس سمت چپ قواعد انجام مي شود و آنقدر عمل جايگزينی ادامه مي يابد تا درصورت صحت ، بتوان به سرترم داده شده رسيد .

ابتدا سمت چپ ترين لغت از عبارت داده شده يعني a توسط تحليلگرلغوي خوانده مي شود. بر سر ورودي علامت * ظاهر مي شود. طبق قاعده $T \rightarrow T * F$ ، قبل از عملگر * فقط يك T مي تواند ظاهر شود. لذا ، با استفاده از قاعده $F \rightarrow id$ ترم پاياني a كه يك شناسه يا id است را با F و سپس طبق قاعده $T \rightarrow F$ ترم مياني F را با T مي توان جايگزين نمود. اكنون ترم بعدي يعني * از سر ورودي خوانده مي شود. طبق قاعده $T \rightarrow T * F$ پس از * ترم مياني F مي تواند ظاهر شود. لذا ، ترم بعدي بايستي F باشد .

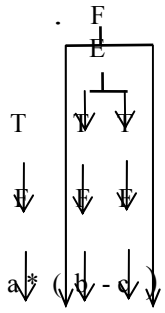
اكنون ورودي ' **(b - c) + d** '، است. بر سر ورودي لغت (وجود دارد. پس ازخواندن (و b درخت بصورت زير خواهد بود :



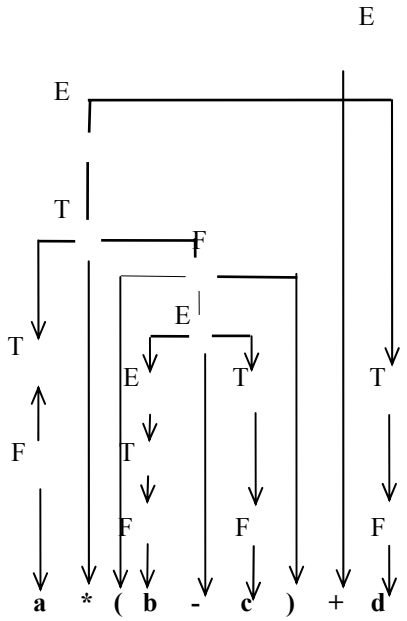
حال بر سر ورودي لغت منهاوجود دارد. طبق قاعده $E \rightarrow E - T$ قبل از عملگر - بايد يك عبارت E وجود داشته باشد. لذا، ترم پايانی b كه يك شناسه يا id است بنابر قاعده $F \rightarrow id$ با ترم مياني F و سپس طبق قواعد $T \rightarrow F$ و $E \rightarrow T$ نهايتاً با ترم مياني E جايگزين مي شود. بنابراين ، تا اين مرحله درخت تجزيه پائين به بالا بصورت زير خواهد بود.



اكنون ورودي ' d + (c - ' ، است. عملگرهاي - و سپس c از ورودي خوانده مي شوند. طبق قاعده $E \rightarrow E - T$ بايد پس از - در ورودي يك ترم ظاهر شود. لذا ، c با F و با T و نهايتاً E - T با E جايگزين خواهند شد . حال (از ورودي خوانده مي شود و به اين ترتيب درداخل درخت تجزيه (E) ظاهر مي شود و طبق قاعده $F \rightarrow (E)$ ميتوان آنرا با F جايگزين كرد.



اكنون ورودي ’d +‘ است. برسرورودي علامت + وجود دارد. قبل از + طبق قاعده$E \to E + T$ بايد يك E وجود داشته باشد. بنابراين T * F را كه اكنون در رأس درخت تجزيه وجود دارد طبق قاعده $T \to T * F$ با T و سپس طبق قاعده $E \to T$ ترم T را با E مي توان جايگزين نمود. نهايتاً درخت تجزيه به صورت زير تبديل مي شود :



همانگونه كه مشاهده نموديد عمل تجزيه از ترمهاي پاياني آغاز و به سر ترم گرامر خاتمه يافت . اين گونه تجزيه را در اصطلاح تجزيه پائين به بالا يا Bottom Up Parsing گويند .

طبق تعريف ، مراحل اشتقاق نمايانگر مراحل رسيدن از سر ترم گرامر به ترمهاي پاياني درون جمله مورد نظر است در تجزيه پائين به بالا ، مراحل اشتقاق درست بر خلاف مراحل تجزيه است چرا كه در اينجا نهايتا به سر ترم گرامر مي رسند در صورتي كه مراحل اشتقاق از سر ترم گرامر آغاز ميشود. به عبارت ديگر مراحل اشتقاق نشان مي دهد كه در طي چه مراحلي جمله داده شده از سر ترم گرامر مشتق شده است. مراحل مشتق كردن عبارت ’a * (b - c) + d‘ از سرترم E

بصورت زير است. بايد توجه داشته باشيد كه مراحل اشتقاق درست بر خلاف مراحل تجزيه است . بنابراين اگر از آخرين مرحله ايجاد درخت تجزيه به مراحل قبلي برگشت شود مراحل اشتقاق بصورت زير معين مي شود :

$$E \Rightarrow E+T \Rightarrow E+F \Rightarrow E+d \Rightarrow T+d \Rightarrow T*F+D \Rightarrow T*(E)+d \Rightarrow T*(E-T)+d \Rightarrow T*(E-F)+D \Rightarrow T*(E-c)+d \Rightarrow T*(T-c)+d \Rightarrow T*(F-c)+d \Rightarrow T*(b-c)+d \Rightarrow F*(b-c)+d \Rightarrow a*(b-c)+d$$

همانگونه كه مشاهده مي شود در طي مراحل اشتقاق بر خلاف اشتقاق چپ كه قبلاً توضيح داده شد، همواره سمت راست ترين ترمها تا رسيدن به ترم پايانی درون جمله داده شده جايگزين مي گردند لذا اين گونه اشتقاق را *اشتقاق راست* گويند . اولين ترم پايانی كه در طي اين مراحل اشتقاق در فرمهاي جمله اي ظاهر شد ترم پايانی d در فرم جمله اي $T+d$ بود و سپس از راست به چپ ترم پايانی بعدي يعني c در فرم جمله أي $T*(E-c)+d$ ظاهر گرديد لذا همانگونه كه مشاهده مي كنيد جايگزينی از سمت راست به چپ انجام شده است .

3.5 گرامرهاي مبهم

چنانچه بتوان براي جمله داده شده بر اساس گرامر زبان ، بيش از يك درخت تجزيه ايجاد نمود، آن گرامر را مبهم مي نامند. در حالت كلي با نگرش به گرامرها نمي توان ابهام را تشخيص داد. اما بايد ديد كه چرا اگر بتوان بيش از يك درخت تجزيه براي جمله داده شده ايجاد نمود، گرامر ابهام دارد . اين بايد يك مزيت باشد، چرا آنرا ابهام ميخوانند ؟ به عنوان يك مثال ساده به گرامر زبانهاي C يا پاسكال براي جملات شرطي if توجه نماييد :

statement → ifSt | whileSt | compoundSt | assignmentSt | callSt |
ifSt → IF cond THEN statement | IF cond THEN statement ELSE statement
cond → exp | exp Relop exp

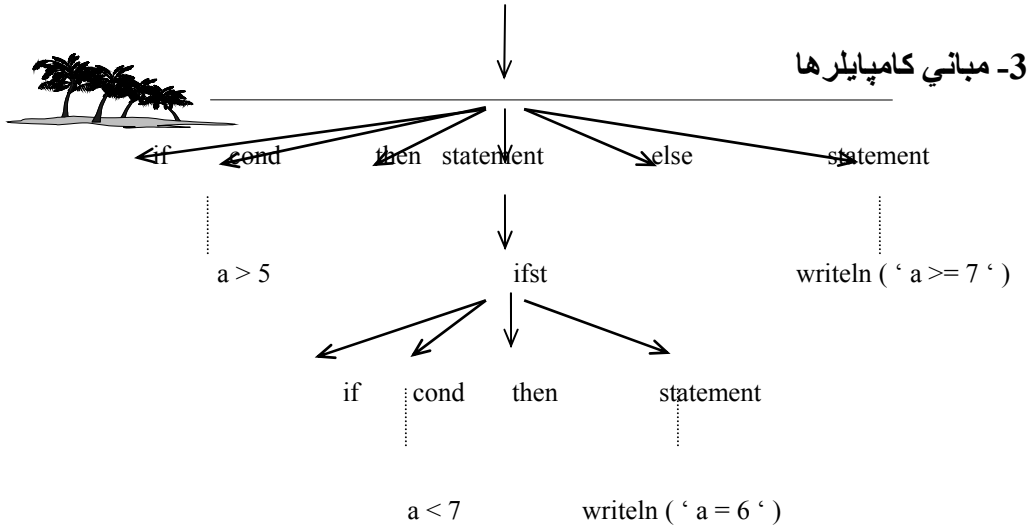
براي نمونه به جمله زير توجه نماييد :

IF a > 5 THEN IF a < 7 THEN writeln (‘ a = 6 ‘) ELSE writeln (‘ a >= 7 ‘)

بنا بر گرامر فوق درخت تجزيه بصورت زير مي تواند باشد :

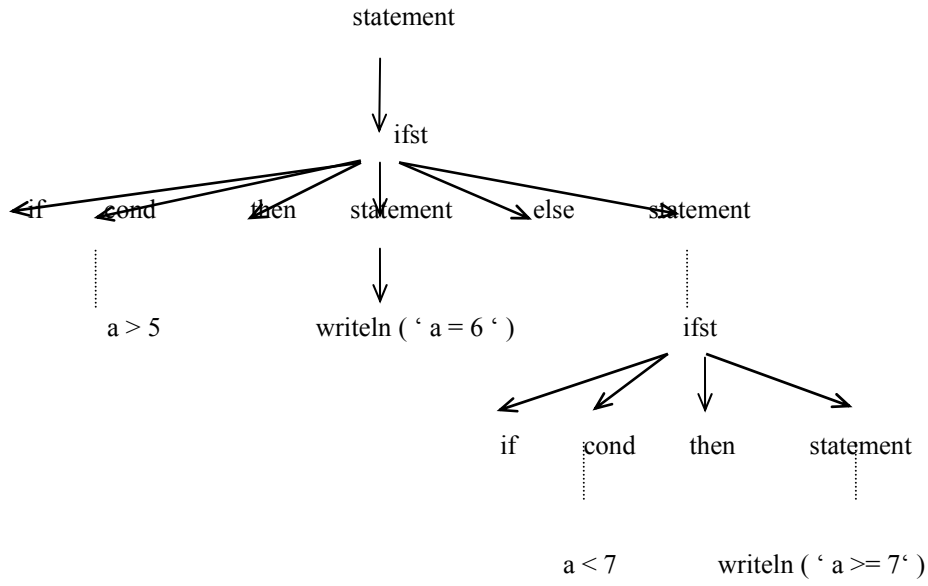
statement

ifst



شكل 3.1- درخت تجزيه براي جمله if

درخت تجزيه را بصورت زير نيز مي توان توليد كرد :



شكل 3.2- درخت تجزيه براي جمله if

بنابر گرامرفوق دو درخت تجزيه با دو مفهوم متفاوت ايجاد شده است . در درخت تجزيه شكل 3و5 ، else وابسته به if خارجي ميباشدودر شكل 3و6 ، else وابسته به if داخلي است . براي رفع اين مشكل در زبانهاي C و پاسكال ، براي تكميل گرامر بطور ضمني و نه بر طبق گرامر ، تعيين كرده اند كه else به نزديكترين if وابسته است . بنابراين درخت تجزيه شكل 3و6 توسط كامپايلر پاسكال ايجاد ميشود. در زبان فورترن 77 با افزايش Endif اين مشكل حل شده و گرامر بصورت زير تبديل گرديده است :

ifst → IF cond THEN statement ENDIF
| IF cond THEN statement ELSE statement ENDIF

بنابراين اگرقراراست كه درجمله داده شده else وابسته به if خارجي باشد، به صورت زير عمل ميشود :

IF a > 5 THEN IF a < 7 THEN writeln (‘ a = 6 ‘) ENDIF ELSE writeln (‘ a >= 7 ‘) ENDIF

بالعكس اگر else وابسته به if داخلي باشد ، جمله بصورت زير نوشته مي شود :

IF a > 5 THEN IF a < 7 THEN writeln (‘ a = 6 ‘) ELSE writeln (‘ a >= 7 ‘) ENDIF ENDIF

اصولاً درحالت كلي ابهام در گرامر ها را نمي توا ن نشان داد و اثبات نمود اما نكته قابل توجه اين است كه اگر گرامري بصورت خود بازگشتي چپ وخود بازگشتي راست براي يك ترم مياني قواعدي داشته باشد آن گرامر مبهم است.

1) $A \rightarrow A\ \alpha \mid \beta\ A$

همچنين اگر در يك گرامر نهايتاً بتوان طي مراحل استنتاج يا اشتقاق از يك ترم مياني به خود آن ترم مياني رسيد، گرامر مبهم ميباشد.

2) $A \Rightarrow \alpha \Rightarrow \ldots \Rightarrow A$

اگر در گرامري قواعد بصورت خود بازگشتي راست يا چپ براي يك ترم مياني وجود داشته باشد و علاوه بر آن قاعده ديگر براي اين ترم مياني بصورتي باشد كه حالت خود بازگشتي در وسطح دو ترم در سمت راست آن قاعده براي آن ترم مياني وجود داشته باشد و در اصطلاح قاعده بصورت Self Embeding ياخودگردان باشد ، بازهم گرامر مبهم خواهد بود.

3) $A \rightarrow A\alpha \mid \beta A\ \delta$

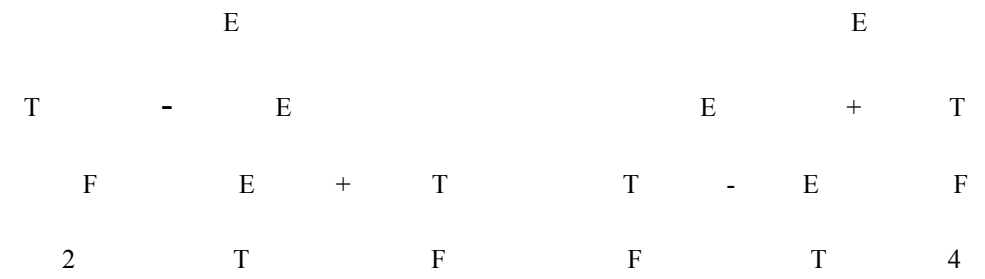
براي نمونه به گرامرمبهم عبارات توجه كنيد :

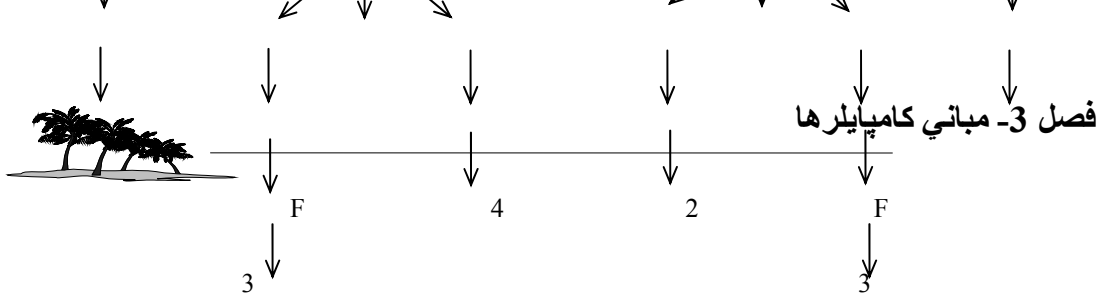
$E \rightarrow E + T \mid T - E \mid T$

$T \rightarrow T * F \mid T / F \mid F$

$F \rightarrow (E) \mid id \mid NO$

مي خواهيم براي جمله 2 - 3 + 4 درخت تجزيه ترسيم نماييم :





الف- درخت با حاصل: $5 = 4 + (2 - 3)$ **ب-** درخت با حاصل: $-5 = (4 + 3) - 2$

شكل 3.3- دو درخت تجزيه متفاوت براي يك عبارت

همانگونه كه مشاهده ميكنيد ايجاد دو نتيجه متفاوت 5 و 5- براي عبارت فوق بخاطر مبهم بودن گرامر و در واقع وجود قاعده بصورت خود بازگشتي چپ و خود بازگشتي راست براي ترم مياني E است .

3.6 تمرين

تمرين 1- مراحل استنتاج را براي عبارت **(a * b - c) / (d - e)** در روشهاي تجزيه بالا به پائين و پائين به بالا مشخص نماييد .

تمرين 2- گرامر ساختار ليست را در نظر بگيريد:

$S \rightarrow$ '(' L ')' | a
$L \rightarrow L$ ',' S | S

درخت تجزيه را براي جملات زير ترسيم نمائيد :

الف - (a , a)

ب - (a, (a, a))

ج - (a, ((a, a), (a, a))

تمرين 3- نشان دهيد كه گرامر زير مبهم است :

$S \rightarrow a\ S\ b\ S \mid b\ S\ a\ S \mid \varepsilon$

تمرين 4- گرامر زير را در نظر بگيريد :

bexpr $\rightarrow$ bexpr *or* bterm | bterm
bterm $\rightarrow$ bterm *and* bfactor | bfactor
bfactor $\rightarrow$ *not* bfactor | (bexpr) | *true* | *false*

الف - درخت تجزيه براي عبارت not(true or false) ايجاد كنيد.
ب - آيا اين گرامر مبهم است.

تمرين 5- چرا اگر در يك گرامر براي يك ترم مياني قواعد به دو صورت خودبازگشتي چپ و خودبازگشتي راست وجود داشته باشند آن گرامر مبهم است.