فرم باکوس نائور

- در علوم رایانه فرم باکوس نائور یا فرم بکوس-نائور یکی از روش‌های بیان قواعد است که برای گرامر مستقل از متن ارائه شده‌است.
- اغلب به عنوان دستور زبان رسمی در علوم رایانه مورد استفاده قرار می‌گیرد؛
- از این میان می‌توان به زبان‌های برنامه‌نویسی، قالب اسناد، دستورزبان دستورات و پروتکل‌های ارتباطی نام برد.

```
<symbol> ::= __expression__

<Non-Zero Digit> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<Digit>              ::= 0 | <Non-Zero Digit>

<DigitString> ::= <Digit> | <Digit> <DigitString>

<Positive Number> ::= <Non-Zero Digit> | <Non-Zero Digit> <DigitString>

<Program> :: = 'PROGRAM' <Identifier> 'BEGIN' <Full Sequence> 'END'.
 <Identifier> :: = <letter> <Restbezeichner>
 <Empty> :: = | <Letter or Digit> <Empty>
 <Letter or Digit> :: = <letter> | <digit>
 <letter> :: = A | B | C | D | ... | Z | a | b | ... | z *)
 <digit> :: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
 <Full Sequence> :: = ...
```

زبان های مستقل ازمتن وزبان های برنامه سازی

- گرامر مثال قبل با BNF بصورت زیر نمایش داده میشود:

<expression> ::= < term> | < expression > + < term >| < expression > - < term >

< term> ::= < factor> | < term> * < factor>| < term> / < factor>

$$E \rightarrow T|E+T\ |E-T$$
$$T \rightarrow F|T*F\ |T/F$$
$$F\ \rightarrow\ (E)|I$$
$$I\ \rightarrow a|b|c$$

- عبارت شرطی if در برنامه نویسی بصورت زیر تعریف میشود

< if-statement> ::= if < expression> then <then –clouse> else < else –clouse>

- عبارت حلقه تکرار while در برنامه نویسی بصورت زیر تعریف میشود

< while-statement> ::= while < expression> <statement>

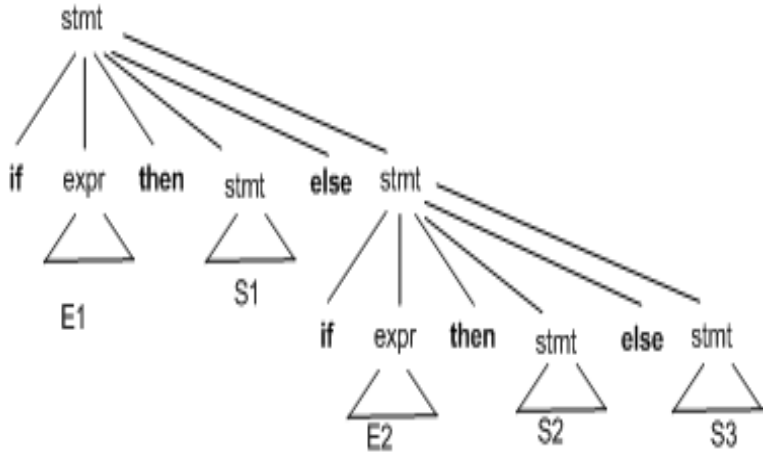
رفع ابهام

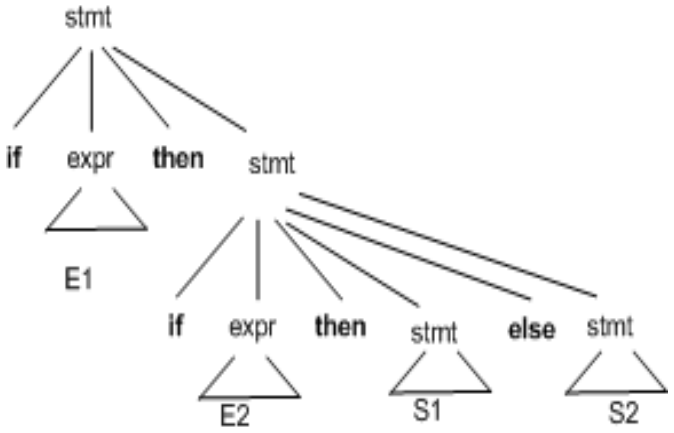
۳-**تغییر گرامر**: هر گرامری یک زبان را توصیف می‌کند. برای تولید یک زبان می‌توان از گرامرهای متعددی استفاده کرد.در نتیجه گرامر مبهم را می‌توان تغییر داد که همان زبان را تولید کند و مبهم نباشد.

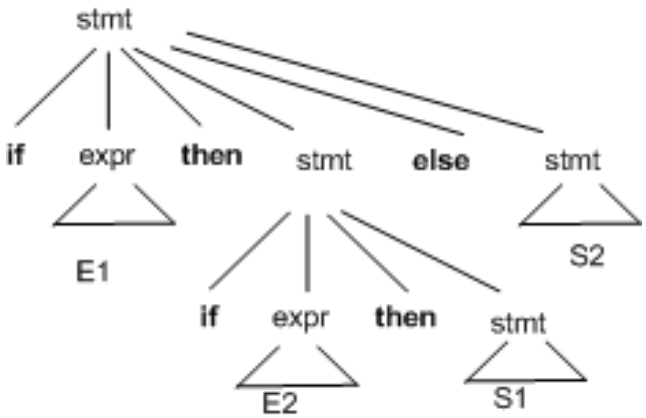
stmt ⟶ **If expr then stmt**

| **If expr then stmt else stmt**

| **other**







بازگشتی چپ

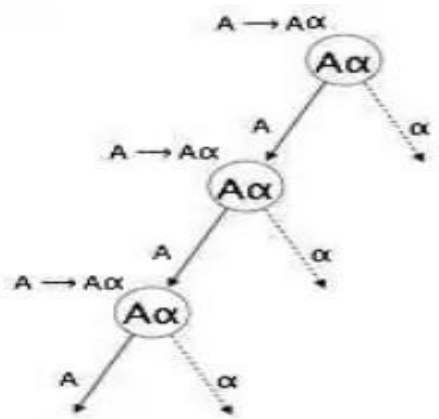
یک گرامر زمانی بازگشتی چپ نامیده می‌شود که نماد غیر پایانی Aرا داشته باشد که اشتقاق آن شامل خود Aبه عنوان چپ‌ترین نماد باشد.(ظاهر شدن غیر پایانه سمت چپ در سمت راست قانون بعنوان اولین عنصر)

گرامر چپ‌ترین یک موقعیت دردسرساز برای تجزیه‌کننده‌های بالا به پایین تلقی می‌شود. تجزیه‌کننده‌های بالا به پایین از نماد آغازین شروع به تجزیه می‌کنند که خود یک نماد غیر پایانی است. از این رو وقتی تجزیه‌کننده با همان نماد غیر پایانی در اشتقاقش مواجه می‌شود، دیگر نمی‌تواند در مورد این که کجا باید تجزیه را متوقف کند تصمیم بگیرد و لذا وارد حلقه نامتناهی می‌شود.

مثال ۱ نمونه‌ای از **بازگشتی بی درنگ چپ** است که در آن Aمی‌تواند هر نماد غیر پایانی و آلفا نشان دهنده یک رشته از نمادهای غیر پایانی باشد.

۲-مثال ۲ نمونه‌ای از **بازگشتی چپ غیرمستقیم** است. تجزیه‌کننده بالا به پایین، ابتدا Aرا تجزیه می‌کند که به نوبه خود رشته‌های شامل خود Aارائه می‌دهد و تجزیه‌کننده ممکن است وارد حلقه بی نهایت شود.

(1)A => Aα | β
(2)S => Aα | β
(3)A => Sd



حذف بازگشتی چپ

یک روش برای حذف بازگشتی چپ، استفاده از تکنیک زیر است:

A => Aα | β

به ترکیب‌های زیر تبدیل می‌شود

A => βA’

A'=> αA' | λ

این کار تأثیری بر رشته‌های اشتقاق یافته از گرامر ندارد، اما بازگشتی چپ بی درنگ را حذف می‌کند. روش دوم استفاده از الگوریتم زیر است که همه بازگشتی‌های چپ مستقیم و غیرمستقیم را حذف می‌کند.

START
Arrange non-terminals in some order like A1, A2, A3,…, A_n
for each i from 1 to n
{
for each j from 1 to i-1
{
replace each production of form A_i ⟹Aj?
with A_i ⟹ δ1? | δ2? | δ3? |…| ?
where A_j ⟹ δ₁ | δ₂|…| δ_n are current A_j productions
}
}
eliminate immediate left-recursion
END

مثال

مجموعه ترکیب زیر:

$S \Rightarrow A\alpha \mid \beta$

$A \Rightarrow Sd$

پس از بکار بردن الگوریتم فوق باید به صورت زیر در آید:

$S \Rightarrow A\alpha \mid \beta$

$A \Rightarrow A\alpha d \mid \beta d$

و سپس بازگشتی چپ بی درنگ را با استفاده از تکنیک نخست حذف می‌کنیم:

$A \Rightarrow \beta dA'$

$A' \Rightarrow \alpha dA' \mid \lambda$

اینک هیچ یک از ترکیب‌ها دیگر، بازگشتی چپ مستقیم یا غیرمستقیم ندارند.

فاکتورگیری چپ

اگر بیش از یک قاعده ترکیب گرامری، **رشته پیشوندی مشترکی** داشته باشد در این **صورت تجزیه‌کننده بالا به پایین نمی‌تواند** تصمیم بگیرد که کدام یک از ترکیب‌ها باید رشته موجود را تجزیه کنند.

مثال اگر یک تجزیه‌کننده بالا به پایین با ترکیبی مانند زیر مواجه شود:

$A \Longrightarrow \alpha\beta \mid \alpha? \mid \ldots$

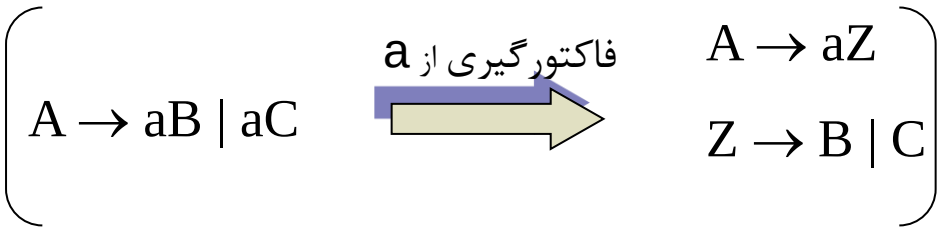
در این صورت نمی‌تواند تصمیم بگیرد که از کدام ترکیب برای تجزیه رشته استفاده کند، زیرا هر دو ترکیب‌ها از نماد پایانی (یا غیر پایانی) یکسانی آغاز می‌شوند. برای رفع این سردرگمی از تکنیکی استفاده می‌کنیم که **فاکتورگیری چپ** نام دارد.

فاکتورگیری چپ گرامر را تبدیل می‌کند تا آن را برای تجزیه‌کننده‌های بالا به پایین مناسب سازد. در این تکنیک یک ترکیب برای هر یک از پیشوندهای مشترک انتخاب می‌کنیم و برای اشتقاق با ترکیب‌های جدید افزوده می‌شود. اینک تجزیه‌کننده تنها یک ترکیب برای هر پیشوند دارد و راحت‌تر می‌تواند تصمیم‌گیری کند.

مثال ترکیب‌های فوق را می‌توان به صورت زیر نوشت:

$A \Rightarrow \alpha A'$

$A' \Rightarrow \beta \mid ? \mid \ldots$



خطای نحوی

الف سطوح خطا

۱- لغوی. مانند دیکته غلط شناسه، کلمه کلیدی یا عملگر

۲- نحوی. مانند عبارت محاسباتی با پرانتزهای نامتعادل

۳- معنایی. مانند استفاده از عملگر با عملوندهای ناسازگار

۴- منطقی. مانند فراخوانی بازگشتی بی نهایت

ب- ویژگی اداره کننده خطای نحوی

- توانایی گزارش حضور خطاها را با وضوح و با دقت
- پوشش هر خطا با سرعت کافی به جهت امکان آشکارسازی خطاهای بعدی
- عدم کاهش بیش از حد سرعت پردازش برنامه های صحیح

اهداف پوشش خطا

- ✓ وجود خطا واضح و صحیح گزارش شود.
- ✓ هر خطا سریعتر پوشش داده شود تا خطاهای بعدی تشخیص داده شوند.
- ✓ کمترین سربار به پردازش برنامه های صحیح اعمال شود.

ج- استراتژی های پوشش خطای نحوی

۱- Panic Mode

نمادهای ورودی یکی یکی دور ریخته شوند تا به یکی از کاراکترهای همگام سازی مشخص شده برسد. ساده ترین روش پوشش، قابل استفاده اکثر روش‌های تجزیه، وارد حلقه بی نهایت نمی‌شود.

۲- Phrase level

پوشش سطح عبارت(پیشوندی از ورودی باقیمانده با رشته ای جایگزین می شود که اجازه دهد پارسر به کارش ادامه دهد.) استفاده از تصحیح موضعی، عدم ورود به حلقه بی نهایت با دقت در انتخاب جایگزینی، ضعف در برخورد با خطاهای اصلی قبل از نقطه تشخیص، قادر به تصحیح هر رشته ورودی

۳- Error production

مولدهای خطا، اضافه نمودن ساختارهای مولد خطا به زبان از قبل تشخیص آنها در زمان تجزیه در رشته ورودی - به گرامر، مولدهایی اضافه شود که ساختارهای خطا را تولید کنند.

۴- Global correction

تصحیح سراسری، انتخاب الگوریتم های تصحیح خطا با قابلیت ایجاد کمترین تغییرات در ورودی برای رفع خطا -رخ دادن حداقل تعداد درج ها، حذف‌ها در رشته ورودی - انتخاب دنباله مینیمم تغییرات برای به دست آوردن اصلاح با کمترین هزینه

محدودیت‌های تحلیل‌گرهای نحوی

تحلیل‌گرهای نحوی ورودی‌های خود را به شکل توکن‌هایی از تحلیل‌گرهای نحوی می‌گیرند.

تحلیل‌گرهای نحوی مسئولیت اعتبارسنجی یک توکن ارائه شده از تحلیل‌گر نحوی را بر عهده دارند. تحلیل‌گرهای نحوی معایب زیر را دارند:

تحلیلگرهای نحوی:

- ✓ نمی‌توانند تشخیص دهند آیا یک توکن معتبر است یا نه
- ✓ آیا یک توکن پیش از استفاده شدن اعلان شده است یا نه
- ✓ آیا یک توکن پیش از استفاده، مقداردهی اولیه شده یا نه
- ✓ آیا یک عملیات که بر روی نوعی از توکن اجرا شده معتبر است یا نه.

همه این‌ها وظایفی هستند که به وسیله **تحلیل‌گر معنایی** انجام می‌یابند