به نام خدا

طراحی زبان‌های برنامه‌سازی

مرجان عظیمی
Azimi.marjan@gmail.com

نحو زبان برنامه‌سازی

نحو، آرایش واژه‌ها به عنوان عناصری از یک دنباله است که رابطه بین آن‌ها را نشان می‌دهد.

دستور x=y+z در c معتبر است ولی x=yz+ معتبر نیست

برای توصیف یک زبان برنامه سازی به بیش از نحو آن زبان نیاز داریم.
نحو دستور x=5.1 + 2.2 مشخص نمی کند آیا xاعلان شده یا نه

معیار عمومی نحوی

◄ قابلیت خوانایی

اگر ساختار مربوط به الگوریتم و داده‌های استفاده شده در برنامه به خوبی روشن باشد خوانایی بالاست.

هر چه دستورات با جزئیات کامل تری نوشته شود قابلیت خوانایی بیشتر می‌شود.

```
Int a[3][2]={{1,2},{2,3},{3,4}};
Int a[3][2]={1,2,2,3,3,4};
```

معیار عمومی نحو

قابلیت نوشتن

قابلیت نوشتن،با استفاده از ساختارهای نحوی منظم و دقیق حاصل می شود.

قابلیت نوشتن و خواندن گاهی با هم در تضادند.

افزونگی:اگر نحوی، یک عنصر اطلاعاتی را به چند روش ارائه کند.

برخی از زبان‌ها تمایل دارند که افزونگی را کم کنند.(مثل فرترن.)

معیار عمومی نحو

◄ سهولت بازرسی

صحت برنامه یا سهولت بازرسی با قابلیت خوانایی و نوشتن در ارتباط است.

◄ سهولت ترجمه

خوانایی و نوشتن از معیار های مورد نیاز برای برنامه نویس و سهولت ترجمه معیار مترجم است.

کلید سهولت ترجمه منظم بودن ساختار است.

لیسپ مثالی از ساختار برنامه ای است که نه قابل خواندن است و نه قابل نوشتن بلکه ترجمه آن آسان است.

با افزایش تعداد ساختارهای نحوی ترجمه مشکل تر می شود.مثل کوبول

معیار عمومی نحو

◄ عدم وجود ابهام

ساختاری مبهم است که دو یا چند تفسیر از آن به دست آید.

به عنوان مثال در پاسکال والگول دو شکل مختلف از دستورات شرطی وجود دارد:

1) If boolean expression then statement else statement2
2) If boolean expression then statement1

وقتی این دو دستور با هم ترکیب می شوند دستور مبهم زیر به وجود می آید.

If boolean expression1 then[(if boolean expression2 then statement1)else statement2]

معیار عمومی نحو

رفع ابهام در الگول : قرار دادن دستورات شرطی در begin...end

رفع ابهام در ادا : هر دستور if با end if خاتمه می یابد.

رفع ابهام درc و پاسکال : آخرین else با نزدیکترین then قبل از آن مطابقت پیدا می کند.

عناصر نحوی زبان

◄ کاراکتر ها

انتخاب مجموعه کاراکتر،اولین کار در طراحی نحو زبان است و معمولا یک مجموعه کاراکتر استاندارد انتخاب می شود.

◄ شناسه ها

رشته ای از حروف و ارقام که با حرف شروع می شوند.

عناصر نحوی زبان

◄ نمادهای عملگر

اغلب زبان ها از کارکترهای + و- برای اعمال محاسباتی استفاده می کنند.

در بعضی از زبان ها برای اعمال اولیه از شناسه ها استفاده می کنند.

مثل PLUS,TIMES در لیسپ

◄ کلمات کلیدی و کلمات رزروی

کلمه کلیدی شناسه ای است که به عنوان بخش ثابتی از نحو یک دستور استفاده می شود.

کلمه کلیدی در صورتی کلمه رزروی است که نتواند به عنوان شناسه انتخاب شود.
مثل if در زبان c

عناصر نحوی زبان

◄ کلمات اضافی

کلمات اختیاری که برای افزایش خوانایی است.

مثال: go to label

◄ توضیحات

از توضیحات برای افزایش قابلیت خوانایی و برای document سازی استفاده می کنیم.

توضیحات به شکل های گوناگونی ظاهر می شوند.

مثل: /* */ در c یا ! در فرترن ۹۰

عناصر نحوی زبان

◄ فضای خالی(space)

قاعده استفاده از فضای خالی در زبان های مختلف متفاوت است.مثلا در فرترن فضای خالی معنای خاصی ندارد.در C فضای خالی حذف می شود اما نه همیشه(=+ یک عملگر ترکیبی است ولی = + منجر به خطای نحوی می شود.)

◄محدود کننده ها و محصور کننده ها

محصور کننده ها فاصله های جفتی هستند،مثل جفت های پرانتز یا begin...end

محدود کننده ها برای بالا بردن قابلیت خوانایی استفاده می شوند.

عناصر نحوی زبان

◄ فرمتهای آزاد و فرمتهای ثابت

فرمت آزاد یعنی دستورات برنامه می توانند از هر جایی از خط شروع شود.نحو فرمت ثابت از موقعیت های خاصی از خط استفاده می کند.

معمولا از فرمت نسبتا ثابت استفاده می شود،مثلا در فرترن ،۵ کاراکتر اول برای برچسب دستورات اختصاص می یابد.

◄ عبارت

عبارات توابعی هستند که به اشیای داده موجود در برنامه دسترسی دارند و مقداری را بر می گردانند.

عناصر نحوی زبان

دستورات

دستورات مهمترین جز نحوی در زبان های دستوری هستند.نحو دستورات تاثیر حیاتی بر روی نظم،قابلیت خوانایی و قابلیت نوشتن یک زبان دارد.

بعضی از دستورات از یک قالب دستور اصلی استفاده می کنند(دستور if) اما بعضی دیگر از نحوهای مختلفی برای هر نوع دستور استفاده می کنند(تعریف آرایه)

آن هایی که از یک فرمت استفاده می کنند(مثل اسنوبال) به نظم اهمیت می دهند،در حالی که بقیه به قابلیت خوانایی(مثل کوبول.)

ساختار برنامه و زیر برنامه

سازمان نحوی برنامه – زیربرنامه ،از زبانی به زبان دیگر متفاوت است.

این تفاوت ها عبارتند از:

◄ تعریف زیربرنامه ها به صورت جداگانه

تعریف هر زیربرنامه به عنوان یک واحد نحوی جداگانه در نظر گرفته می شود و هر زیر برنامه به طور جدا ترجمه می شود و هنگام بار کردن به هم پیوند زده می شوند.(فرترن چنین ساختاری دارد.)

ساختار برنامه و زیر برنامه

◄ تعریف داده ها به صورت جداگانه

تمام عملیاتی که شئ داده خاصی را دستکاری می کنند در یک زیربرنامه قرار می گیرند.

مانند : مفهوم کلاس ها در زبان c++

◄ تعریف زیربرنامه به صورت تودر تو

تعریف زیربرنامه های تودر تو در دوران اولیه الگول ،فرترن و پاسکال رایج بود.

اما در جاوا وc++ حذف شد.

تعریف زیربرنامه ها به صورت تودر تو ،محیط ارجاع غیرمحلی را برای زیربرنامه ها فراهم می کند.(تعریف توابع به صورت تو در تو.)

ساختار برنامه و زیربرنامه

◄ تعریف واسط مجزا

در این نوع زبان ها ،پیاده سازی زبان شامل چندین زیر برنامه است که باید با هم تعامل داشته باشند این تعامل توسط واسط انجام می شود.(مثل فایل های header(پسوند”.h”)در c که حاوی تعاریف این واسط است.)

◄ توصیف داده ها جدا از دستورات اجرایی است

برای مثال در کوبول . بخش داده ها ،بخش رویه ها(دستورات اجرایی.)

ساختار برنامه و زیربرنامه

◄ تعریف زیربرنامه ها به طور غیرمجزا

تمایز خاصی بین دستورات برنامه اصلی و زیربرنامه ها وجود ندارد مثل اسنوبال ۴

و بیسیک

مراحل ترجمه

- ترجمه ممکن است بسیار ساده باشد، مانند پرولوگ و لیسپ. اما اغلب پیچیده است مانند ادا
- پیاده سازی زبانها بصورت مفسری سرعت اجرای برنامه را پایین می آورد.
- اگر ساختار برنامه اجرایی متفاوت از ساختار برنامه منبع باشد، فرآیند ترجمه پیچیده خواهد بود.
- اجرای کارآمد مطلوب است وباید برنامه ها با ساختار اجرایی کارآمدی ترجمه شوند.

مراحل ترجمه

- کامپایلر استاندارد دوگذره:
  - گذر تحلیل: برنامه را به اجزا تشکیل دهنده آن تجزیه می کند و اطلاعاتی از قبیل ، کاربرد نام متغیر را بدست می آورد.
  - گذر سنتز: با استفاده از این اطلاعات جمع آوری شده، برنامه مقصد را تولید می کند.

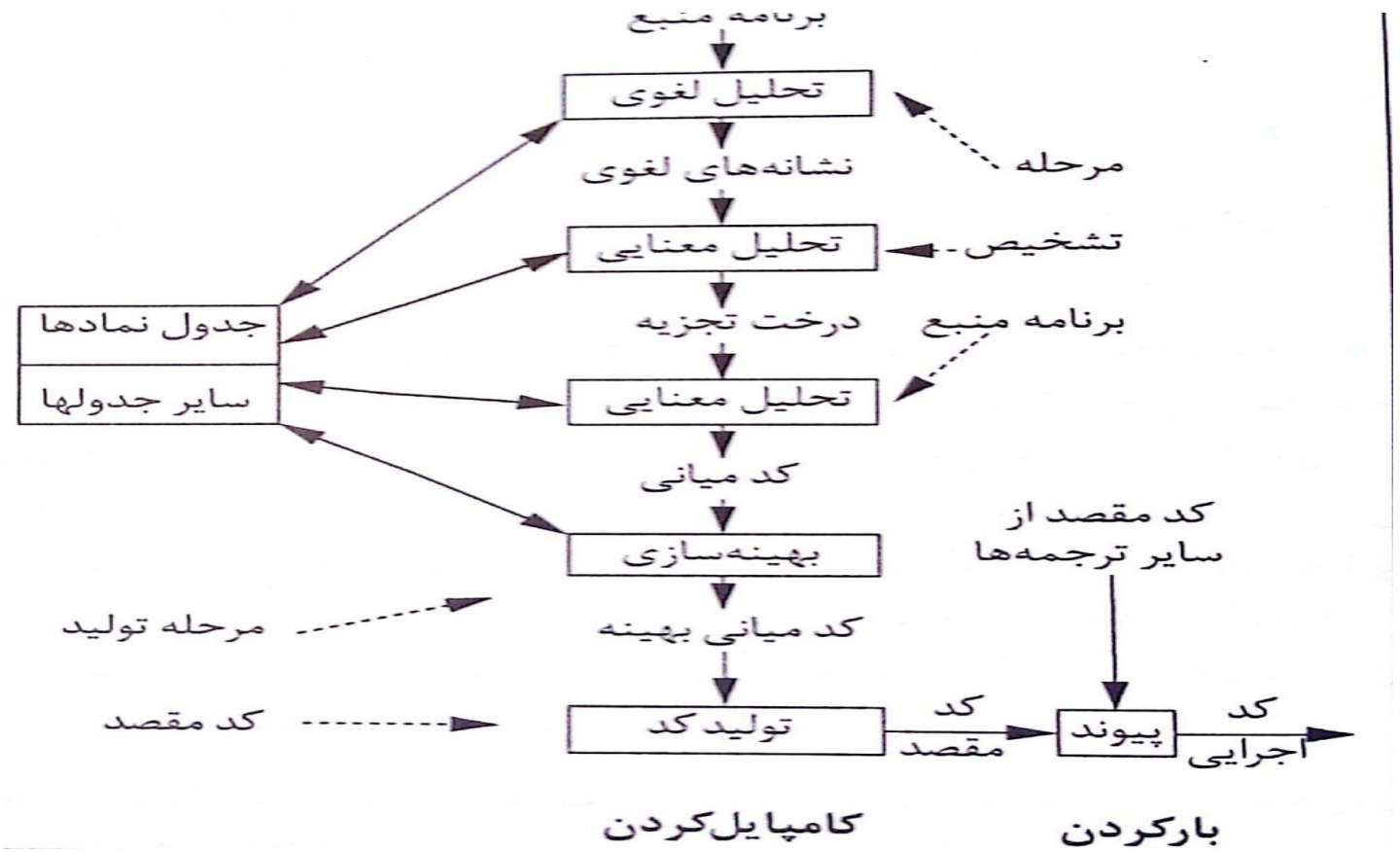
مراحل ترجمه

اگر سرعت ترجمه مهم باشد،راهبرد یک گذره مفید است، که پس از تحلیل، برنامه فورا به کد مقصد تبدیل می گردد. مانند پاسکال

اگر سرعت اجرا مهم باشد، کامپایلر دو یا چندگذره بهتر است .

آنچه در سرعت ترجمه واجرا تاثیرمی گذارد، پیچیدگی زبان است، نه تعدادگذرها

مراحل ترجمه(ساختار یک کامپایلر)



تحلیل برنامه منبع

تحلیل ساختار برنامه در حین ترجمه ، کاراکتر به کاراکتر انجام می

شود:

تحلیل لغوی

اولین مرحله ترجمه، دسته بندی دنباله ای از کاراکترها در اجزای بنیادی است .
مانند شناسه ها، فاصله‌ها، عملگرها، کلمات کلیدی، فضای خالی، توضیحات و...

تحلیل لغوی

تحلیلگرلغوی، خطوط برنامه ورودی را می خواند، به نشانه هایی تبدیل می کند و نوع هر عنصر لغوی (عدد، شناسه، فاصل و...) را تعیین

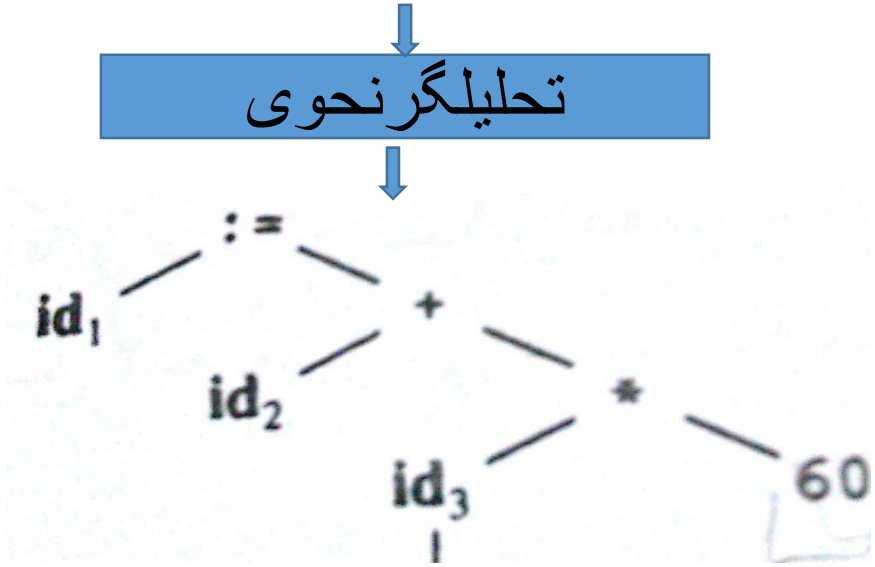
می کند و آنها را به مراحل بعدی ترجمه تحویل می دهد.

مثال : position = initial rate * 60

تحلیل نحوی

مرحله دوم ترجمه است که ساختارهای بزرگ برنامه(دستورات، اعلانها، عبارات و...)
با استفاده از عناصر لغوی که توسط تحلیلگر لغوی تولید شدند،شناسایی می شوند.

Id1 = id2 + id3 * 60



تحلیل معنایی

ساختارهای نحوی که توسط تحلیلگر نحوی تشخیص داده شدند پردازش می شوند تا از نظر مشخصات آن زبان مجاز و یک واحد معنادار باشند و ساختار کد مقصد اجرایی شکل می‌گیرد.

✓دراین مرحله خطاهای معنایی بررسی می شود.

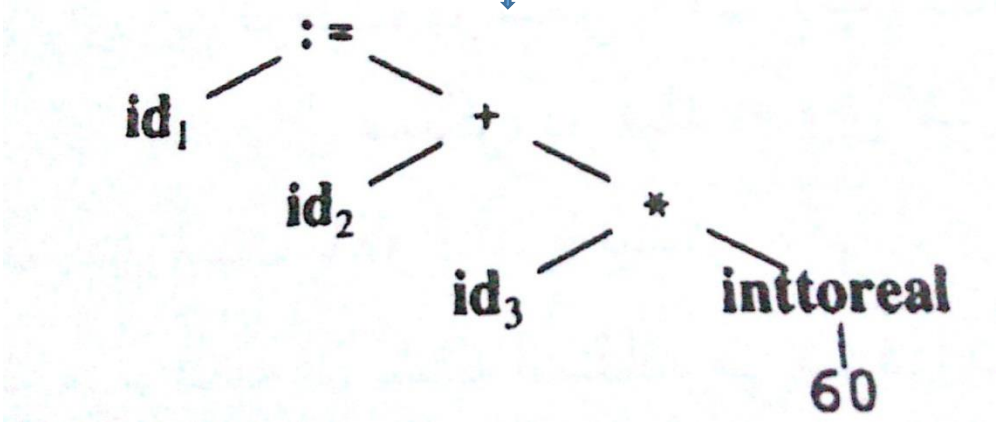
مثلا : بسیاری از کامپایلرها اگر از اعداد اعشاری برای اندیس آرایه استفاده شود، خطایی را گزارش می دهند.

تحلیل معنایی

مثال: عملگرinttoreal عدد صحیح ۶۰ را به عدد اعشاری تبدیل می کند .

تحلیلگر نحوی

تحلیلگر معنایی



متداولترین اعمالی که تحلیلگرمعنایی انجام میدهد

نگهداری جدول نماد:

جدول نماد ساختمان داده ای است شامل یک رکورد برای هرشناسه ومیدانهایی برای صفات آن شناسه مانند:نوع آن(متغیر،آرایه و...)، نوع مقادیر(صحیح، حقیقی و...)، محیط ارجاع و...

درج اطلاعات ضمنی:

اغلب در برنامه منبع،اطلاعات ضمنی اند و باید در برنامه مقصد به صورت صریح بیان شوند، مثلا درفرترن نوع متغیرها اعلان نمی شود.

متداولترین اعمالی که تحلیلگرمعنایی انجام میدهد

کشف خطا

در هر مرحله ازترجمه، خطاهای متعددی ممکن است تشخیص داده شوند، مانند استفاده از دو اندیس بجای سه اندیس درآرایه ای که بصورت سه بعدی تعریف شده .

✓ تحلیلگر معنایی باید خطاها را تشخیص دهد وکاری کند که تحلیلگرنحوی به کارش ادامه دهد.

متداولترین اعمالی که تحلیلگرمعنایی انجام میدهد

پردازش ماکرو وعملیات زمان ترجمه

ماکرو قطعه برنامه ای است که به طور جدا تعریف می شود و هنگام ترجمه برنامه، با فراخوانی ماکرو، در برنامه درج میشود.

همه زبانها ویژگی ماکرو و یا عملیات زمان اجرا ندارند. درصورت وجود، پردازش آنها توسط تحلیلگر معنایی انجام می گیرد.

ماکرو می تواند رشته ای مانند مقدار 3.1416 برای pi باشد که هر مراجعه به pi موجب می شود به‌جای آن 3.1461 قرار گیرد.

پردازش ماکرو

عملیات زمان ترجمه

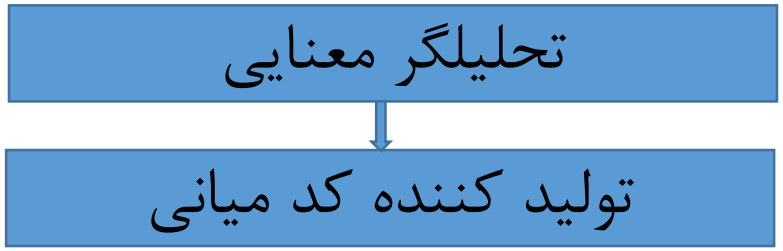
عملی است که باید در زمان ترجمه انجام شود تا فرآیند ترجمه را کنترل کند.

به عنوان مثال: دستور #define در c موجب می شود تا قبل از ترجمه برنامه، ثوابت یا عباراتی ارزیابی شوند.

ترکیب برنامه مقصد

❑ بهینه سازی :

کد میانی تولید شده در تحلیلگر معنایی قبل از تولید کد مقصد، بهینه سازی می شود.




```
Temp1 = inttoreal(60)
Temp2 = id3 * temp1
Temp3 = id2 + temp2
Id1 = temp3
```

بهینه سازی(ادامه)

خروجی کد میانی به بهینه ساز داده میشود

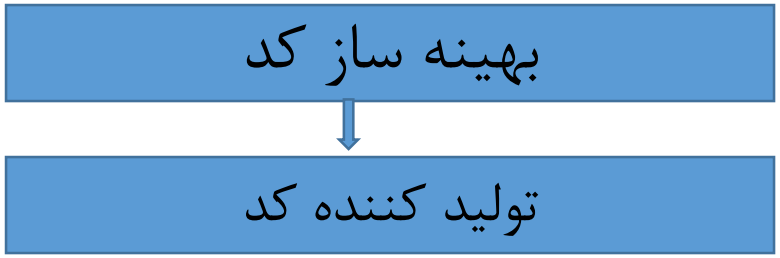
بهینه ساز کد

```
Temp1 = id3  *  60.0

Id1  =  id2  + temp1
```

تولید کد :

وقتی نمایش داخلی برنامه ترجمه شده، بهینه شد، باید به صورت دستورات زبان اسمبلی، کد ماشین یا شکل دیگری از برنامه مقصد تبدیل شود.




```
Movf id3,r2
Mulf #60.0,r2
Movf id2,r1
Addf r2,r1
movf r1,id1
```

پیوند زدن و بارکردن

قطعاتی از کد که از ترجمه زیربرنامه های جدا به وجود آمده اند، در برنامه اجرایی نهایی با هم پیوند زده می شوند.

راه اندازی خودکار(خودرانی)

- اغلب مترجم زبان جدید، به همان زبان نوشته می شود، این عمل را **خودرانی** گویند

برای این کار می توان کامپایلر را به صورت دستی ترجمه کرد.

مدل های رسمی ترجمه

تعریف رسمی نحو زبان برنامه نویسی، گرامر نام دارد و گرامر متشکل از مجموعه ای از قواعد(بنام مولدها) است که ترتیب کاراکترها (عناصر لغوی) را مشخص می کند.

گرامرهای BNF

گرامر BNF توسط جان باکوس برای تعریف نحوی الگول ایجاد شد تقریبا در همان زمان،گرامر مستقل از متن توسط نوام چامسکی برای تعریف نحو زبان های طبیعی ارائه شد که مشابه گرامر BNF بود، تفاوت آن ها فقط در نشانه گذاری هاست.

- ✓ گرامر BNF همان گرامر مستقل از متن است.

نحو

- **تعریف زبان**

هرزبان توسط یک گرامر تعریف می شود .

نحو در ارتباط با شکل است نه معنا .

در نحو زبانهای برنامه سازی :

زبان هر مجموعه ای از رشته های کاراکتری(با طول متناهی) است که از الفبای محدودی از نمادها انتخاب می شود.

- مجموعه ای از تمام دستورات انتساب C یک زبان است.
- مجموعه ای شامل دنباله‌های a,b

نمایش یک زبان

مثال:

به کمک عبارت زیر

Digit :=0|1|2|3|4|5|6|7|8|9

که در آن ارقام ۰،۱،۲و...را ترمینال و digit راغیر ترمینال می نامیم، زبان اعداد صحیح را بصورت زیر معرفی می کنیم

Digit Digit *

مثالی از نحو

نحو دسته ای از دستورات ساده انتساب با گرامر زیر تعریف می شود

<assignment statement> ::= <variable> = <arithmetic expression>

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

<arithmetic expression> - <term>

<term> ::= primary> | <term> × <primary>| <term> /<primary>

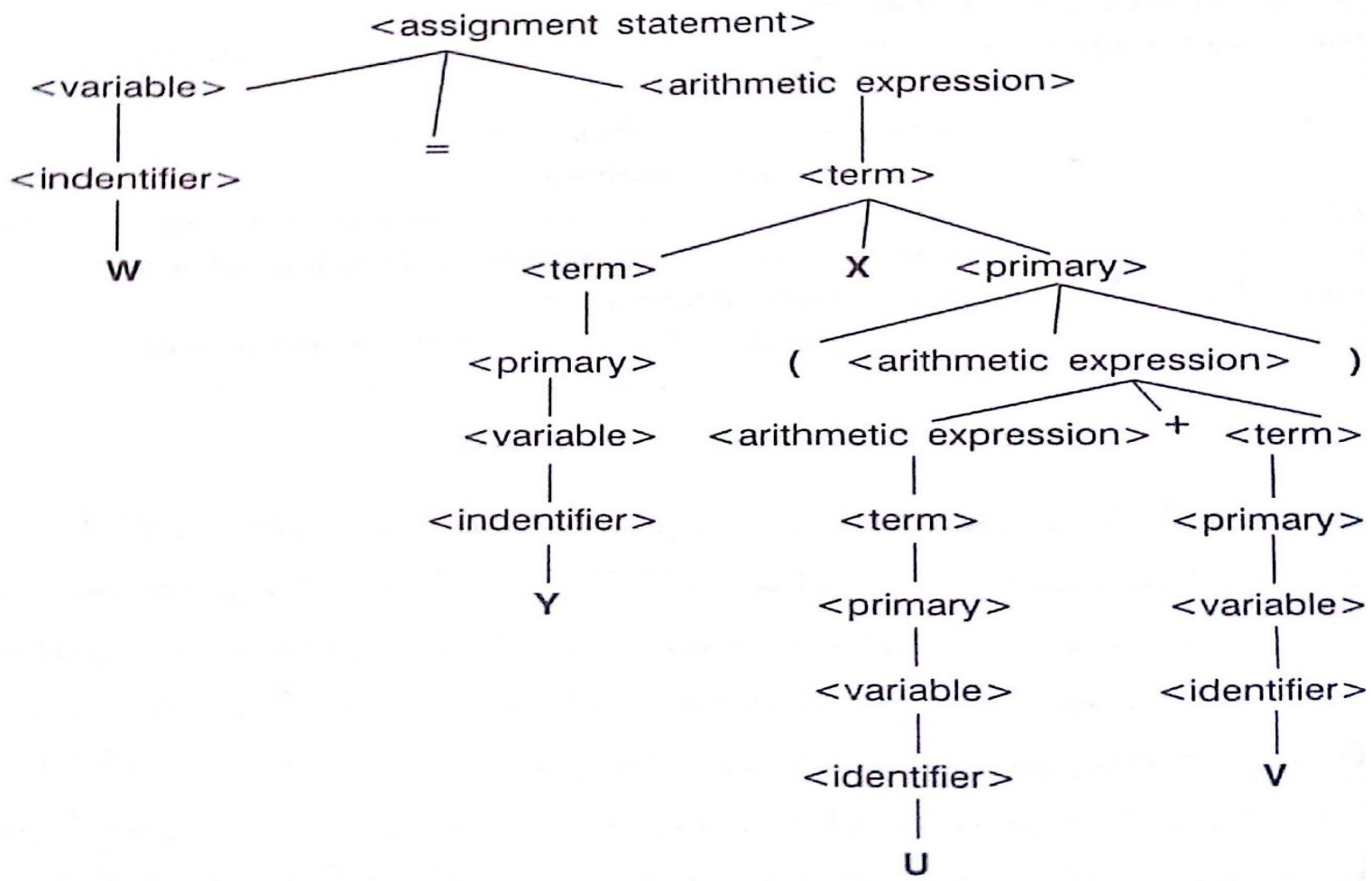
<primary> ::= <variable> | <number> | (<arithmetic expression>)

درخت تجزیه

برای تعریف اینکه آیا رشته ای در یک زبان تعریف شده،توسط گرامر BNF ،نمایش دهنده برنامه ای است که از نظر نحوی معتبر است،باید با استفاده از قواعد گرامری، تحلیل نحوی یا تجزیه رشته را انجام دهیم. اگر این رشته با موفقیت تجزیه شود، در زبان وجود دارد.

- ✓ **درخت تجزیه،** ساختار نحوی شهودی را برای یک برنامه آماده می کند.

مثال: درخت تجزیه ای تحلیل نحوی دستور $W = Y \times (U + X)$



..... درخت های تجزیه

هر گرامرساختاری را برای یک برنامه مشخص می کند:

- برنامه نویس با مطالعه گرامر می تواند ساختارهایی را شناسایی کند که همه با ترکیب شدن با یکدیگر، برنامه های درستی را ایجاد می کنند.

- یک زبان ممکن است با گرامرهای مختلفی تعریف شود .

ابهام

گاهی ابهام صفت یک گرامر است و به زبان مربوط نمی شود

که تمام رشته های دودویی را تولید می کند، مبهم است: G1به عنوان مثال، گرامر

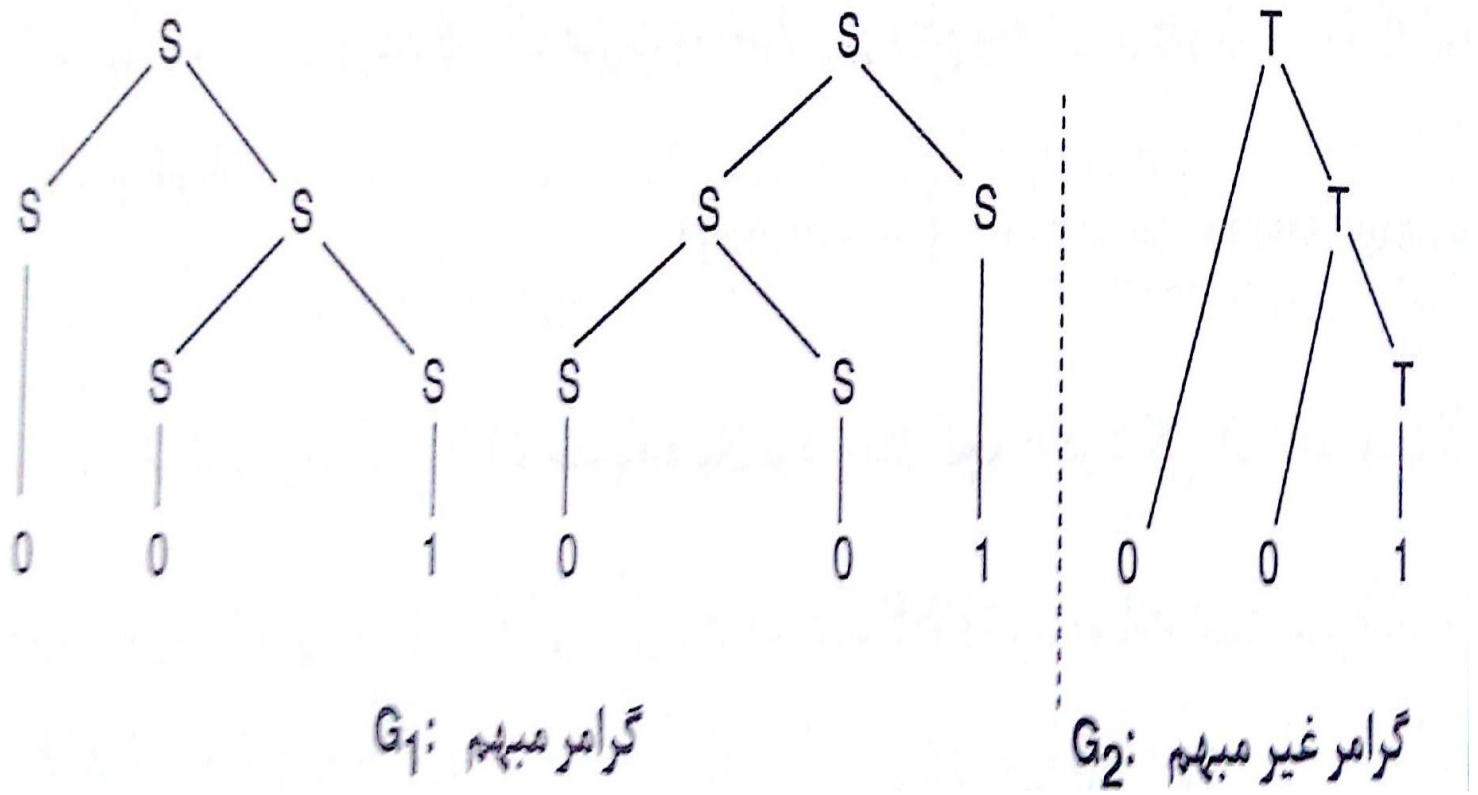
$$G1 : S \rightarrow SS \mid 0 \mid 1$$

زیرا رشته ای در زبان وجود دارد که دارای دو درخت تجزیه است.

زبانی از رشته های دودویی، ماهیتا مبهم نیست، زیرا گرامر غیرمبهم زیر نیز همان رشته ها را تولید می کند:

$$G2 : T \rightarrow 0T \mid 1T \mid 0 \mid 1$$

.....ابهام



G_1: گرامر مبهم

G_2: گرامر غیر مبهم

بسط نشانه های BNF

گرامر های BNF علی رغم قدرت،سهولت و دقت نمی توانند قواعد نحو زبان برنامه نویسی را به خوبی با برنامه نویسی حرفه ای انتقال دهند.

به عنوان مثال : برای بیان نحو "عدد صحیح علامت دار،دنباله ای از ارقام است که قبل از آن ها یک علامت – و+ قرار دارد."قاعده بازگشتی BNF زیر را می نویسیم.

<singed integer>::= +<integer> | -<integer>

<integer>::= <digit> | <integer> <digit>

نشانه گذاری توسعه یافتهBNF

عبارات نشانه گذاری زیر قدرت گرامر BNF را تغییر نمی دهد ،اما توصیف زبان ها را ساده تر می کند.

مثال :

<singed integer> ::= [+|-] <digit> {<digit>}*

<identifier> ::= <letter>{<letter>|<digit>}*

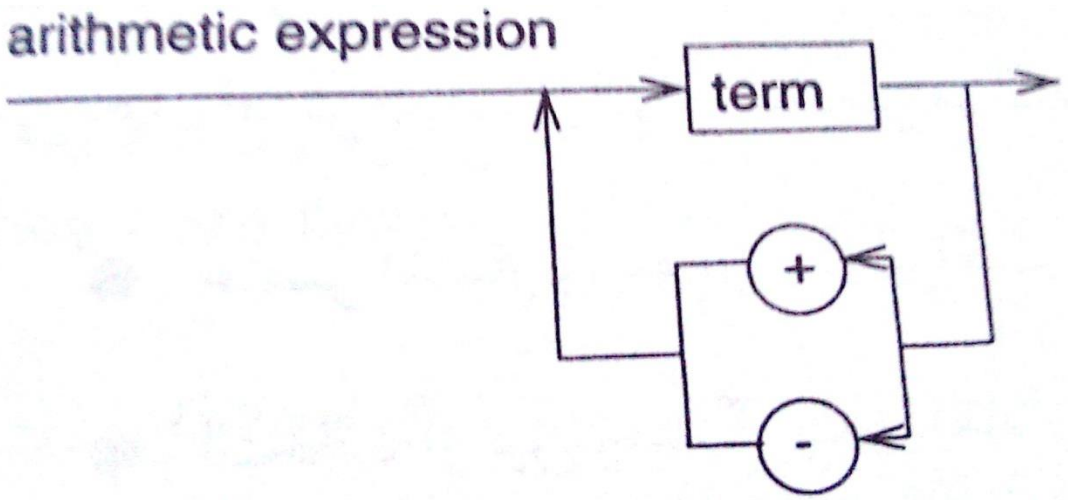
بسط نشانه های BNF....

نمودار نحوی

نمودار نحوی ابزاری گرافیکی برای بیان قواعد BNF توسعه یافته است

مثال:

<arithmetic expression> ::= <term> { [+|-] <term> }*



ماشین خودکار متناهی(FSA)

مدل ساده ایست که نشانه ها را تشخیص می دهد.

- یک FSA دارای یک حالت شروع، یک یا چند حالت نهایی و مجموعه ای از انتقالها، از یک حالت به حالت دیگر است.
- تا زمانی که می دانیم در چه حالتی هستیم، می توانیم تعیین کنیم که آیا ورودی که دیدیم، بخشی از نشانه ایست که فعلاً با آن سر و کار داریم یا خیر.
- هر رشته ای که ماشین را از طریق مجموعه ای از انتقالها، از حالت شروع به حالت نهایی ببرد، پذیرفته می شود.

(NFA)ماشین خودکار متناهی غیر قطعی

یک ماشین خودکار متناهی غیر قطعی است، وقتی از یک حالت چندین انتقال با ورودی یکسان، خارج شود.

دراین ماشین وقتی رشته ای پذیرفته می شود که مسیری از گره شروع به یکی از گره های نهایی وجود داشته باشد، حتی اگر مسیرهای دیگری وجود داشته باشند که به حالت نهایی نرسند.

گرامرهای منظم

گرامرهایی که در آن هر قاعده متشکل از یک ترمینال است که ممکن است قبل یا بعد از آن یک غیر ترمینال باشد.

مانند:

$$A \rightarrow 0A \mid 1A \mid 0$$

دوبخش اول برای تولید هر رشته دودویی است و بخش سوم مشخص می کند آخرین کاراکتر باید صفر باشد.

- ✓ هر گرامر منظم را می توان با یک FSA نمایش داد.

عبارات منظم

با استفاده از عبارات منظم می توانیم هر زبانی را که توسط گرامر منظم یا FSA تعریف شد، نمایش دهیم، گرچه تبدیل هر FSA به عبارت منظم همیشه واضح نیست.

1. نمادهای پایانی منفرد، عبارت منظم اند.
2. اگر a و b عبارات منظم باشند، آنگاه $a \vee b, ab, (a), a^*$ نیز عبارت منظم اند.
3. برای شناسه ها، عبارت منظم $\leftarrow$ letter(letter $\vee$ digit)*

قدرت محاسباتی FSA

FSA ها حاوی مقدار متناهی از state ها ست،بنابراین،مجموعه ای از رشته هایی را که می توانند تشخیص دهند،محدود است.

به عنوان مثال مجموعه $a^n b^n$ نمی تواند توسط FSA ها تشخیص داده شود.

FSA ها فقط گرامرهای منظم را تشخیص می دهند و برای تشخیص گرامرهای مستقل از متن از PDA ها استفاده می کنیم.

ماشین خودکار پشته ای(PDA)

با استفاده از گرامرBNF رشته هایی را در زبان ایجاد می کنیم با این ماشین می توان تشخیص داد که رشته مورد نظر در زبان وجود دارد یا نه.

شرط پذیرش رشته درPDA : stack خالی شود،رشته به انتها برسد و در state نهایی باشیم.

مثال : $a^n b^n$