جلسه چهارم

تشخیص خطا

Scanner خطا های زیادی را نمی تواند تشخیص دهد، زیرا دیدگاه محلی نسبت به برنامه دارد به عبارتی برنامه مبدا را به صورت کاراکتر به کاراکتر می خواند به عنوان مثال در عبارت زیری **Scanner** نمی تواند تشخیص دهد که **if** به صورت غلط نوشته شده است(**fi**) و آن را به عنوان یک شناسه معتبر به **parser** تحویل می دهد و کنترل این قبیل خطا ها را به **parser** و مولفه های بعد آن محول می سازد. $fi(x == f(x))$

به عنوان شناسه معتبر مي شناسد

☐ اگر رشته ای از کاراکتر ها با الگوی هیچ کدام از توکن ها مطابقت نداشته باشد در این صورت خطای لغوی رخ می دهد و **Scanner** می بایست این قابلیت را داشته باشد که از این خطا و خطا های دیگر گذر کرده و عمل **Scan** (تحلیل لغوی) را تا پایان فایل ادامه دهد.

☐ در روش **panic mode** اسکنر از رشته ورودی حذف می کند تا جائی که ادامه کاراکتر ها با الگوی یکی از **token** ها تطابقت داشته باشد.

مکانیزم های گذر از خطا

- **panic mode**
- اضافه کردن یک کاراکتر جدید مثل :=) که دو نقطه را اضافه کردیم
- تعویض یا جایگزینی دو کاراکتر مجاور مثل >< → <>
- تغییر یک کاراکتر

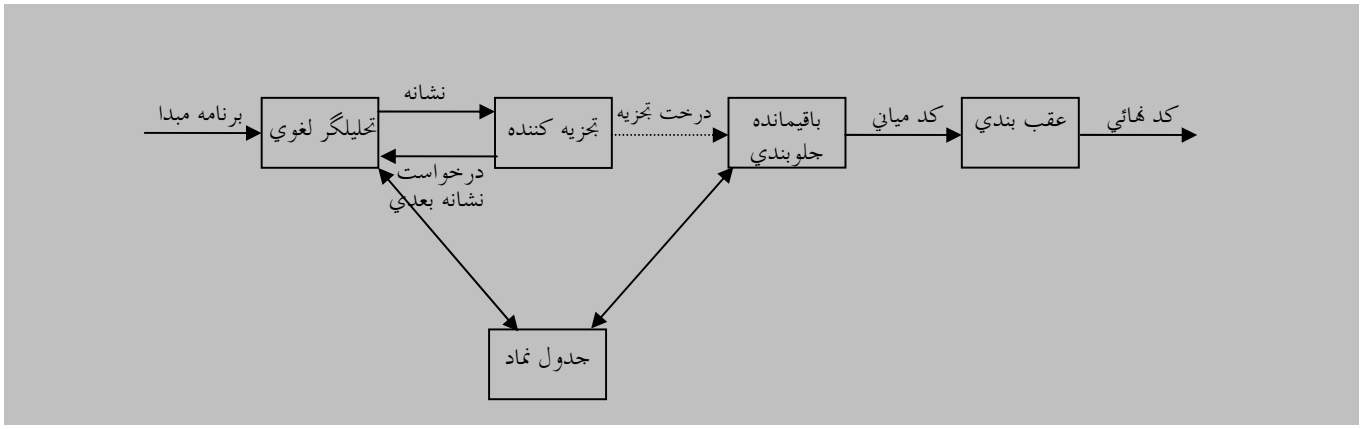
تحلیل نحوی

ساختار هر زبان با قواعد آن مشخص می شوند ساختار زبان های برنامه سازی با گرامر های مستقل از متن پیاده سازی میشود.

$G(S \ , \ NT \ , \ T \ , \ P)$

$S : start\ \ symbol$
$NT : None\ \ Terminal\ \ Set$
$T : Ter\min al\ \ Set$
$P : Production\ \ Rule$

شکل زیر موقعیت تجزیه کننده در مدل کامپایلر را نشان می دهد.



- وظیفه اسکنر تشخیص توکن ها می باشد.
- وظیفه اصلی **parser** تشخیص این که رشته توکن های تولید شده توسط **scanner** ، آیا توسط قواعد زبان قابل تولید هست یا نه

انواع پارسر ها

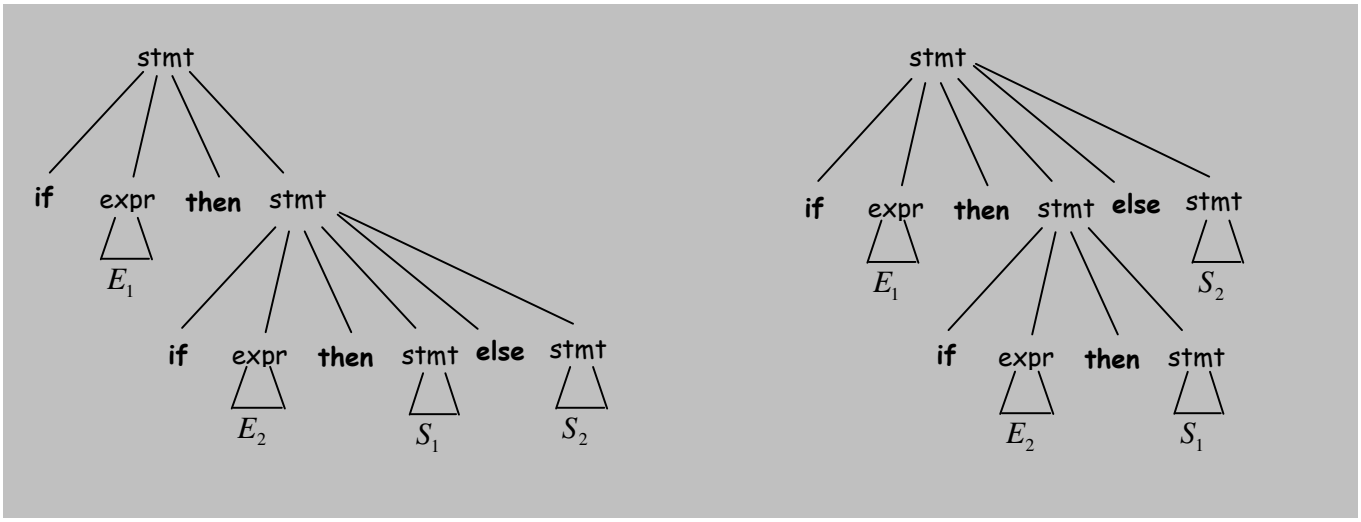
- پارسرهای غیر پیشگو کننده، باید قابلیت **back tracking** داشته باشد.
- پارسرهای پیشگو کننده ۱- پارسر های بالا به پایین ۲- پارسر های پایین به بالا

- در غیر پیشگو کننده با گرفتن رشته ای از توکن ها با سعی و خطا سعی می کنند که تشخیص دهند آیا این رشته قابل تولید توسط گرامر زبان هست یا نه که این پارسر ها باید خاصیت **back tracking** داشته باشند
- در پارسرهای بالا به پایین از جمله **start symbol** شروع کرده و با اشتقاق سعی در تولید جمله مورد نظر را دارند به عبارتی جمله مورد نظر برگ های درخت اشتقاق می باشد.
- در پارسر های پایین به بالا از جمله ورودی(رشته توکن ها) شروع کرده و سعی در رسیدن به **start symbol** را دارند به عبارتی به جای عبارت سمت چپ، سمت راست را می گذارند.
- در پیشگو کننده گرامر ها بایستی فاقد ابهام باشند.

ابهام:

گرامری دارای ابهام است اگر بتوان برای حداقل یک جمله از زبان تولید شده توسط آن گرامر دو درخت اشتقاق چپ یا دو درخت اشتقاق راست مختلف پیدا کرد به عنوان مثال. گرامر زیر را در نظر بگیرید ملاحظه می شود که برای تولید جمله **If** E_1 **then if** E_2 **then** S_1 **else** S_2 دو درخت اشتقاق چپ وجود دارد و گرامر مبهم است. و می بایست رفع ابهام شود.

```
stmt → if expr then stmt
     | if expr then stmt else stmt
     | other
```



- اگر زبانی داشته باشیم که نتوانیم گرامر مبهمی برا ی آن پیدا کنیم زبان ذاتا غیر مبهم است.

رفع ابهام گرامر صفحه قبل به شکل زیر خواهد بود

$$stmt \rightarrow m\_stmt \mid unm\_stmt$$
$$m\_stmt \rightarrow if \;\; expr \;\; then \;\; m\_stmt \mid other$$
$$unm\_stmt \rightarrow if \;\; expr \;\; then \;\; m\_stmt \;\; else \;\; unm\_stmt$$

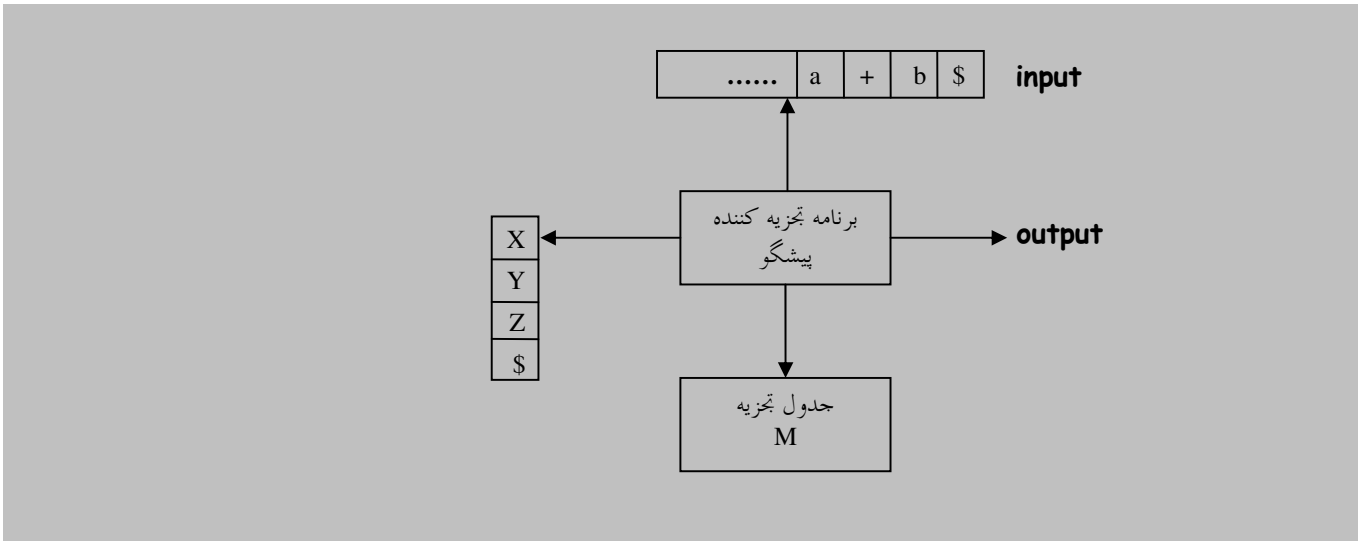
- در اسکنر ورودی کاراکتر است در پارسر ورودی یک توکن است

تجزیه گر پیشگوئی پذیر غیر بازگشتی(بالا به پایین)

$LL(1)$ در این روش "L" اول به این معنی است که رشته ورودی از سمت چپ خوانده میشود و "L" دوم به این معنی است که پارسر از سمت چپ ترین اشتقاقات استفاده می کند.

ورودی رشته ای از توکن هاست که پایان آن را با $ نمایش می دهیم

دارای یک پشته است که در ابتدای پشته $ و **startsymbol** قرار می گیرد، همچنین دارای یک جدول تجزیه می باشد، ساختار کلی این نوع پارسر به فرم زیر است.



مجموعه آغازین $first(\alpha)$

برای سمبل α مجموعه آغازین یا $first$ آن مجموعه ترمینال هائی است که شکل های جمله ای مشتق شده از α با آن شروع می شوند

قواعد ساخت مجموعه آغازین x

- اگر α پایانه باشد $first(\alpha)=\{\alpha\}$
- اگر α بتواند ε را تولید کند آنگاه ε را به مجموعه $first(x)$ اضافه کنید.
- اگر X یک غیر پایانه و $X \rightarrow Y_1Y_2\dots Y_k$ یک مولد باشد. و مجموعه های $first(Y_1), first(Y_2), first(Y_3) \dots first(Y_n)$ شامل ε باشند یعنی همه Y_i ها بتوانند تهی را تولید کنند در نتیجه X نیز می تواند ε را تولید کند که در این صورت ε به مجموعه $first(X)$ اضافه می شود.
- اگر X یک غیر پایانه و $X \rightarrow Y_1Y_2\dots Y_k$ یک مولد باشد. مجموعه $first(Y_1)$ (به جز ε) به مجموعه $first(X)$ اضافه میشود.
- اگر X یک غیر پایانه و $X \rightarrow Y_1Y_2\dots Y_k$ یک مولد باشد. و ε در مجموعه $first(Y_1)$ باشد(Y_1 میتواند ε را تولید کند) در این صورت علاوه بر $first(Y_1)$ (به جز ε) مجموعه $first(Y_2)$ (به جز ε)ا نیز به $first(X)$ اضافه میشود.
- با توجه به قاعده تولید $X \rightarrow Y_1Y_2\dots Y_i\dots Y_k$ اگر ε در $first(Y_1), first(Y_2), \dots first(Y_{i-1})$ باشد مجموعه $first(Y_i)$ (به جز ε) به مجموعه $first(X)$ اضافه میشود.

مثال: گرامر زیر را در نظر بگیرید.

1) $E \to TE'$
2) $E' \to +TE' \mid \varepsilon$
3) $T \to FT'$
4) $T' \to *FT' \mid \varepsilon$
5) $F \to (E) \mid id$

پاسخ

$first(T') = \{ * , \varepsilon \}$
$first(F) = \{ (, id \}$
$first(E') = \{ + , \varepsilon \}$
$first(T) = \{ (, id \}$
$first(E) = \{ (, id \}$

مثال. رشته a را توسط گرامر زیر تولید کنید، آیا a عضو این زبان هست یا نه

$S \to aB \mid b$
$B \to S \mid \varepsilon$

پاسخ. $S \to aB \to a$ و $first(B) = \{\varepsilon, a, b\}$, $first(s) = \{b, a\}$

مثال.در گرامر زیر first() ها را مشخص کنید.

$S \to BAa \mid b$
$B \to aF \mid \varepsilon$
$A \to a \mid b \mid aF$
$F \to a \mid c$

پاسخ

$first(B) = \{a, \varepsilon\}$
$first(A) = \{a, b\}$
$first(f) = \{a, c\}$
$first(s) = \{a, b\}$

مثال. در گرامر زیر first() ها را مشخص کنید.

$S \to BAa \mid b \mid Ba$
$B \to aF \mid \varepsilon$
$A \to a \mid b \mid aF \mid \varepsilon$
$F \to a \mid c$

پاسخ

$first(B) = \{a, \varepsilon\}$
$first(A) = \{a, b, \varepsilon\}$
$first(f) = \{a, c\}$
$first(s) = \{a, b\}$