به نام خدا

طراحی کامپایلرها

تحلیل گر نحوی
قسمت دوم

۴

فاطمه پورغلامعلی

تحلیل نحوی بالا به پایین

- روش **Recursive Descent**
- **$A \rightarrow \alpha | \beta$**
- درصورتیکه توکن خوانده شده در **First(α)** باشد، قاعده **$A \rightarrow \alpha$** برای گسترش درخت پارس انتخاب میشود
- درصورتیکه توکن خوانده شده در **First(β)** باشد، قاعده **$A \rightarrow \beta$** برای گسترش درخت پارس انتخاب میشود
- اگر **ε** از **α** مشتق میشود و توکن خوانده شده در **Follow(A)** باشد، قاعده **$A \Rightarrow^* \varepsilon$** انتخاب میشود

مشکلات تحلیل نحوی بالا به پایین

- تعریف مساله: با داشتن یک گرامر مستقل از متن (غیر مبهم) میخواهیم یک پارسر پیشگو بالا به پایین طراحی کنیم (نیاز به برگشت به عقب نداشته باشد)
- در روش **RD** برای هر غیرپایانه یک روال داریم
  - درصورتیکه توکن خوانده شده در **First** یک قاعده باشد روال آن قاعده فراخوانی میشود
- قاعده چپ‌گرد (بازگشتی چپ) در گرامر مستقل از متن، می‌تواند تحلیلگر نحوی را در حلقه‌ی نامتناهی بیاندازد
  - مثلاً برای قاعده‌ی **A → A α** رویه‌ی زیر را داریم:

```
procedure A
begin
   if Lookahead is in First(Aα) then
      call procedure A
end
```

- راه حل: حذف چپ‌گردی
  - البته بدون این که زبان متناظر گرامر تغییر کند

مواجهه با چپ‌گردی

- الگوریتمی برای حذف چپ‌گردی
  - ایده‌ی اولیه:

$$A \rightarrow A\alpha \mid \beta \quad \Rightarrow \quad \begin{aligned} A &\rightarrow \beta R \\ R &\rightarrow \alpha R \mid \varepsilon \end{aligned}$$

  - مثال:

$$\begin{aligned} expr &\rightarrow expr + term \mid expr - term \mid term \\ term &\rightarrow \mathbf{id} \end{aligned}$$

$$\Downarrow$$

$$\begin{aligned} expr &\rightarrow term\ rest \\ rest &\rightarrow + term\ rest \mid - term\ rest \mid \varepsilon \\ term &\rightarrow \mathbf{id} \end{aligned}$$

حل مشکلات: چپ‌گردی (۱)

- گرامر دارای چپ‌گردی، قاعده‌ای دارد که به کمک آن اشتقاقی مثل $A \rightarrow^* A\,\alpha$ برای بعضی α ها قابل انجام است
- تحلیل نحوی بالا به پایین نمی‌تواند با چنین گرامرهایی کنار بیاید، زیرا باید تصمیم ثابتی بگیرد که نتیجتاً باعث خاتمه‌نیافتن آن خواهد شد

$A \rightarrow A\,\alpha \mid \beta$

$A \rightarrow A\,\alpha \rightarrow A\,\alpha\,\alpha \rightarrow A\,\alpha\,\alpha\,\alpha \rightarrow \ldots$

- به صورت کلی:

$A \rightarrow A\,\alpha \mid \beta$

گرامر دارای چپ‌گردی

$A \rightarrow \beta\,A'$

$A' \rightarrow \alpha\,R \mid \varepsilon$

گرامر بدون چپ‌گردی

حل مشکلات: چپ‌گردی (۲)

- به شکل غیررسمی:
  - همه‌ی قواعد مربوط به **A** را در نظر بگیرید و به این شکل مرتب کنید (به طوری که هیچ β_i با **A** آغاز نشود):

$$A \rightarrow A\,\alpha_1 \mid A\,\alpha_2 \mid \ldots \mid A\,\alpha_m \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$$

  - حالا روشی را که پیش‌تر گفته شد روی آن‌ها اعمال کنید:

$$A \rightarrow \beta_1\,A' \mid \beta_2\,A' \mid \ldots \mid \beta_n\,A'$$
$$A' \rightarrow \alpha_1\,A' \mid \alpha_2\,A' \mid \ldots \mid \alpha_m\,A' \mid \varepsilon$$

  - با این حساب، در مثال ما:

$$E \rightarrow E + T' \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (\,E\,) \mid \mathbf{id}$$

$$E \rightarrow T\,E'$$
$$E' \rightarrow +\,T\,E' \mid \varepsilon$$

$$T \rightarrow F\,T'$$
$$T' \rightarrow *\,F\,T' \mid \varepsilon$$

حل مشکلات: چپ‌گردی (۳)

- مشکل: اگر چپ‌گردی دو یا چند مرحله عمق داشته باشد، روش گفته‌شده کافی نیست

S → A a | b
A → A c | S d | ε

الگوریتم حذف چپگردی در گرامر

ورودی: گرامر G که غیرپایانه‌های آن به صورت $A_1, A_2, \ldots, A_n$ مرتب شده‌اند
خروجی: گرامر معادلی که چپ‌گردی ندارد

۱ـ غیرپایانه‌ها را به شکل $A_1, A_2, \ldots, A_n$ مرتب کنید (که A_1 نماد شروع است)
۲ـ برای i از 1 تا n شروع
برای j از 1 تا i - 1 شروع
هر قاعده به شکل $A_i \rightarrow A_j \gamma$ را با قاعده‌ای به شکل
$A_j \rightarrow \delta_1 \mid \delta_2 \mid \ldots \mid \delta_k$ به طوری که $A_i \rightarrow \delta_1 \gamma \mid \delta_2 \gamma \mid \ldots \mid \delta_k \gamma$
قاعده‌های فعلی مربوط به A_j هستند، جای‌گزین کنید
پایان
چپ‌گردی مستقیم را از همه‌ی قاعده‌های مربوط به A_i حذف کنید
پایان

حل مشکلات: چپ‌گردی (۴)

- استفاده از الگوریتم در مثال:

$A_1 \rightarrow A_2\ \mathbf{a} \mid \mathbf{b} \mid \varepsilon$

$A_2 \rightarrow A_2\ \mathbf{c} \mid A_1\ \mathbf{d}$

i = 1

در مورد A_1 هیچ چپ‌گردی‌ای وجود ندارد

i = 2

برای **j** از **1** تا **1**

قاعده‌های به شکل $A_2 \rightarrow A_1\ \gamma$ را با قاعده‌هایی به شکل

$A_2 \rightarrow \delta_1\ \gamma \mid \delta_2\ \gamma \mid \ldots \mid \delta_k\ \gamma$ به طوری که $A_1 \rightarrow \delta_1 \mid \delta_2 \mid \ldots \mid \delta_k$

قاعده‌های فعلی مربوط به A_1 هستند، جای‌گزین می‌کنیم

بنابراین در مثال، $A_2 \rightarrow A_1\ \mathbf{d}$ به $A_2 \rightarrow A_2\ \mathbf{a\ d} \mid \mathbf{b\ d} \mid \mathbf{d}$ تبدیل می‌شود

آن‌چه باقی می‌ماند:

$A_1 \rightarrow A_2\ \mathbf{a} \mid \mathbf{b} \mid \varepsilon$

$A_2 \rightarrow A_2\ \mathbf{c} \mid A_2\ \mathbf{a\ d} \mid \mathbf{b\ d} \mid \mathbf{d}$

حل مشکلات: چپ‌گردی (۵)

- کار تمام نشده. هنوز باید چپ‌گردی را حذف کنیم:

$A_1 \rightarrow A_2\ \mathbf{a} \mid \mathbf{b} \mid \varepsilon$

$A_2 \rightarrow A_2\ \mathbf{c} \mid A_2\ \mathbf{a}\ \mathbf{d} \mid \mathbf{b}\ \mathbf{d} \mid \mathbf{d}$

- باید این قاعده را که پیش‌تر گفته شد، اعمال کنیم:

$A \rightarrow A\,\alpha_1 \mid A\,\alpha_2 \mid \ldots \mid A\,\alpha_m \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$

$A \rightarrow \beta_1\ A' \mid \beta_2\ A' \mid \ldots \mid \beta_n\ A'$

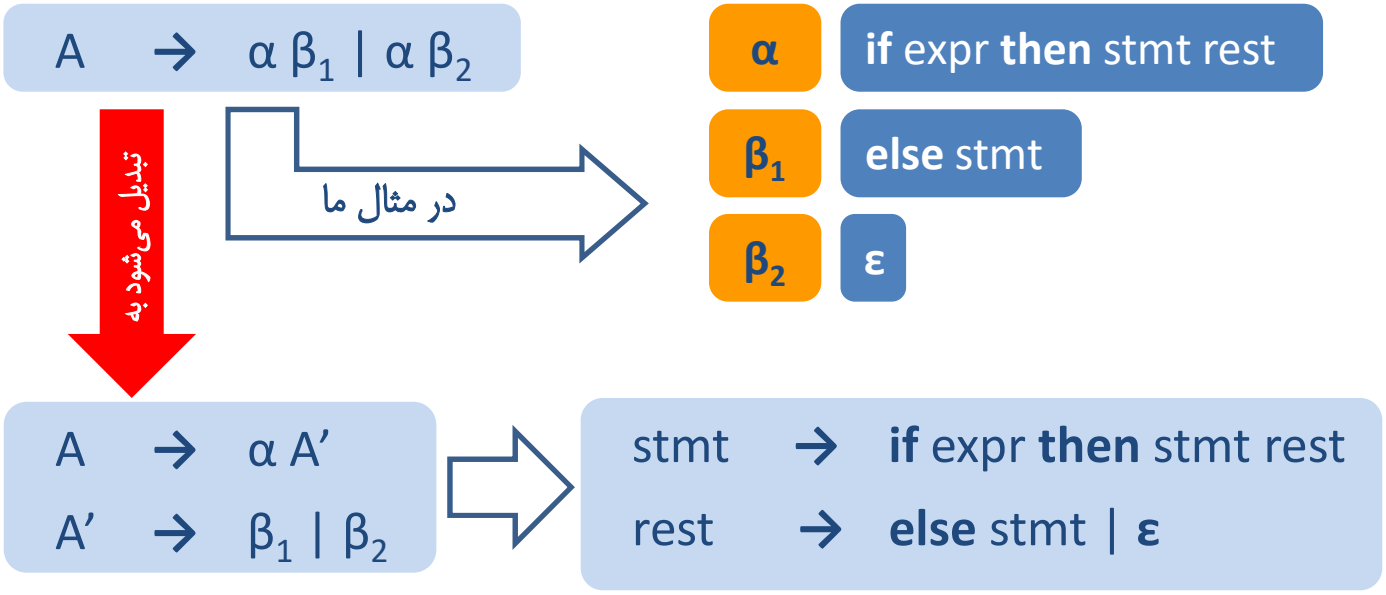
$A' \rightarrow \alpha_1\ A' \mid \alpha_2\ A' \mid \ldots \mid \alpha_m\ A' \mid \varepsilon$

مشکل دیگر: فاکتورگیری چپ

- مشکل: کدام‌یک از دو قاعده‌ی زیر باید انتخاب شود؟

stmt → **if** expr **then** stmt **else** stmt
| **if** expr **then** stmt

- حالت کلی چنین مشکلی:

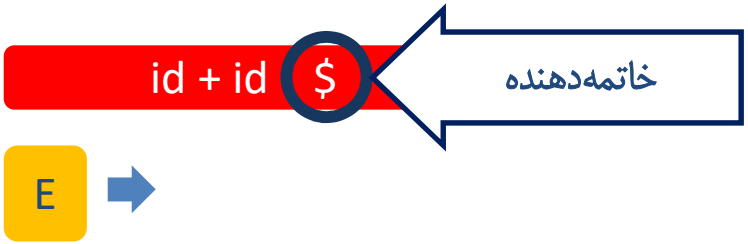


تحلیل نحوی مبتنی بر جدول

- (۱) پویش چپ به راست
- (۲) یافتن اشتقاق از چپ
- گرامر:

$$E \rightarrow T\ E'$$
$$E' \rightarrow +\ T\ E' \mid \varepsilon$$
$$T \rightarrow id$$

- ورودی:
- اشتقاق:
- انباره (پشته) پردازش



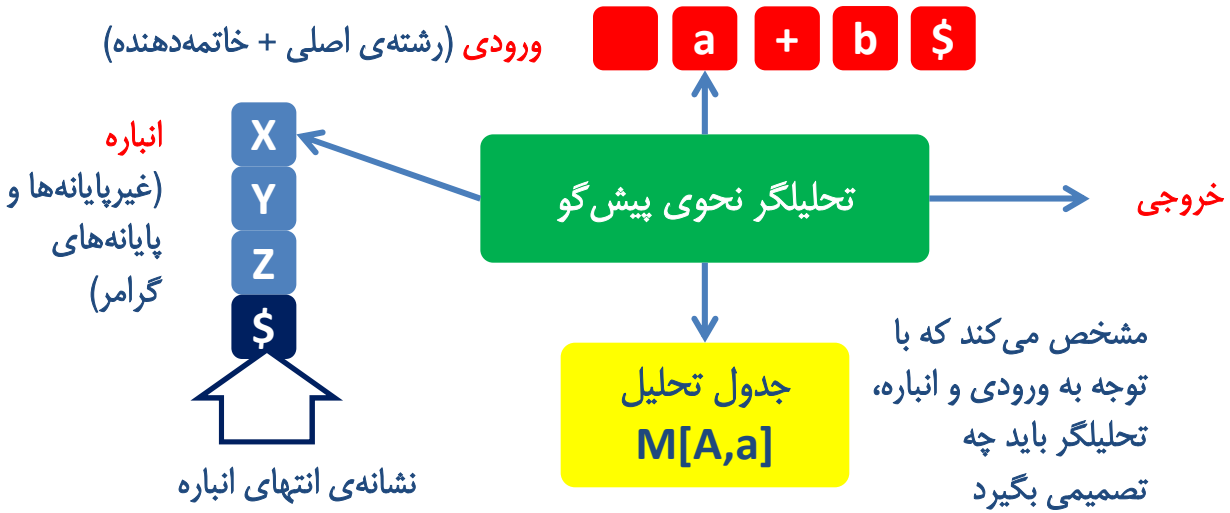
گرامرهای LL(1)

- **L**: پویش ورودی از چپ به راست
- **L**: ساخت اشتقاق از چپ
- **1**: برای تصمیم‌گیری در تحلیل نحوی، علاوه بر انباره، تنها به «یک» نشانه‌ی پیش رو از ورودی نگاه می‌کند
- در جدول تحلیل گرامرهای **LL(1)**، در هیچ خانه‌ای ۲ قاعده به چشم نمی‌خورد
- ویژگی‌های گرامرهای **LL(1)**:
  - مبهم نیستند و چپ‌گردی ندارند
  - برای قاعده‌هایی به شکل $A \rightarrow \alpha \mid \beta$:
    1. هیچ دو رشته از رشته‌هایی که از α و β مشتق می‌شوند، نباید با نماد یکسان (مثلاً a) شروع شوند. به عبارت دیگر اشتراک **First(α)** و **First(β)** باید تهی باشد.
    2. ε حداکثر از یکی از α یا β ممکن است مشتق شود
    3. اگر ε از α مشتق می شود، آنگاه اشتراک **First(β)** و **Follow(A)** باید تهی باشد.
- همه‌ی گرامرها را نمی‌توان به شکل **LL(1)** تبدیل کرد

گرامرهای LL(1)

- بررسی **LL(1)** بودن یک گرامر
  - ابتدا فهرست تمام قواعد را در می آوریم
  - بعد قانون ۱ و ۲ را بررسی میکنیم
  - حالا فالو غیرپایانه ها را محاسبه میکنیم
  - حالا قانون ۳ را بررسی میکنیم

اجزاء یک پارسر LL(1)



- رفتار کلی تحلیلگر با توجه به: a (ورودی فعلی)، و X (بالای انباره)

1. داخل انباره قرار میدهیم: $ S و انتهای رشته هم یک $ قرار میدهیم
2. وقتی X = a = $، کار پایان می‌یابد، و رشته‌ی ورودی پذیرفته می‌شود
3. وقتی X = a ≠ $، X را از روی انباره برمی‌داریم، نشانه‌ی بعدی را از ورودی می‌گیریم و به قدم (1) برمی‌گردیم
4. وقتی X یک غیرپایانه باشد، خانه‌ی M[X,a] در جدول بررسی می‌شود:
   - اگر خطا بود، رویه‌ی برخورد با خطا فراخوانی می‌شود
   - اگر قاعده‌ای به شکل X → U V W بود، X را از روی انباره برمی‌داریم و به ترتیب W، V و U را روی انباره قرار می‌دهیم

تحلیل نحوی غیربازگشتی (۱)

- مثال:

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +\ T\ E' \mid \varepsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \varepsilon \\
F &\rightarrow (\ E\ ) \mid id
\end{aligned}
$$

جدول M

غیرپایانه‌ها	نشانه‌ی ورودی					
	id	+	*	(	)	$
E	E → T E'			E → T E'		
E'		E' → + T E'			E' → ε	E' → ε
T	T → F T'			T → F T'		
T'		T' → ε	T' → * F T'		T' → ε	T' → ε
F	F → id			F → (E)		

تحلیل نحوی غیربازگشتی (۲)

- مثال روند تحلیل نحوی:

انباره	ورودی	خروجی
$ E	id + id * id $	E → T E’
$ E’ T	id + id * id $	T → F T’
$ E’ T’ F	id + id * id $	F → id
$ E’ T’ id	id + id * id $	
$ E’ T’	+ id * id $	T’ → ε
$ E’	+ id * id $	E’ → + T E’
$ E’ T +	+ id * id $	
$ E’ T	id * id $	T → F T’
$ E’ T’ F	id * id $	F → id
$ E’ T’ id	id * id $	
$ E’ T’	* id $	T’ → * F T’
$ E’ T’ F *	* id $	
$ E’ T’ F	id $	F → id
$ E’ T’ id	id $	
$ E’ T’	$	T’ → ε
$ E’	$	E’ → ε
$	$	

گسترش ورودی

تحلیل نحوی غیربازگشتی (۳)

- اشتقاق از چپ مثال قبلی:

E ➡ T E’ ➡ F T’ E’ ➡ **id** T’ E’ ➡ **id** E’ ➡ **id + T** E’

➡ **id + F** T’ E’ ➡ **id + id** T’ E’ ➡ **id + id * F** T’ E’

➡ **id + id * id** T’ E’ ➡ **id + id * id** E’ ➡ **id + id * id**

- پارسر **LL(1)** همواره سمت راست قواعد را طوری در انباره قرار میدهد که سمت چپ ترین علامت بالا باشد
- اگر این علامت غیرپایانه باشد زودتر از بقیه غیرپایانه ها گسترش می یابد
- این معادل اشتقاق از چپ است.

چه چیزی کم داریم؟

- جدول تحلیل **M** هنوز ساخته نشده است!
  - یک ـ مجموعه‌های **First** و **Follow** را برای گرامر به دست آورید.
  - دو ـ الگوریتم زیر را جهت ساخت جدول تحلیل اعمال کنید:

۱ـ قدم‌های (۲) و (۳) را به ازای هر قاعده به شکل $A \rightarrow \alpha$ تکرار می‌کنیم:

۲ـ اگر پایانه‌ی **a** در **First(α)** باشد، $A \rightarrow \alpha$ را در خانه‌ی **M[A, a]** می‌گذاریم

۳ـ اگر ε عضو **First(α)** باشد، $A \rightarrow \alpha$ را در تمام خانه‌های **M[A, b]** می‌گذاریم که **b** عضو **Follow(A)** است

۴ـ همه‌ی خانه‌های باقی‌مانده‌ی جدول که خالی‌اند، معرّف خطای نحوی هستند.

ساخت جدول تحلیل (۲)

- مثال ۱:

S → i E t S S’ | a
S’ → e S | ε
E → b

First(S) = { i, a }
First(S’) = { e, ε }
First(E) = { b }

Follow(S) = { e, $ }
Follow(S’) = { e, $ }
Follow(E) = { t }

S → i E t S S’

First(i E t S S’) = { i }

S → a

First(a) = { a }

E → b

First(b) = { b }

S’ → e S

First(e S) = { e }

S’ → ε

First(ε) = { ε }

Follow(S’) = { e, $ }

غیرپایانه‌ها	نشانه‌ی ورودی					
	a	b	e	i	t	$
S	S → a			S → i E t S S’		
S’			S → ε S’ → e S			S → ε
E		E → b				

ساخت جدول تحلیل (۳)

- مثال ۲:

E	→	T E’
E’	→	+ T E’ | ε
T	→	F T’
T’	→	* F T’ | ε
F	→	(E) | id

First(**E, F, T**) = { **(, id** }
First(**E’**) = { **+, ε** }
First(**T’**) = { ***, ε** }

Follow(**E, E’**) = { **), $** }
Follow(**F**) = { ***, +,), $** }
Follow(**T, T’**) = { **+,), $** }

E → T E’
First(T E’) = First(**T**) = { **(, id** }

E → + T E’
First(**+** T E’) = { **+** }

E → **b**
First(**b**) = { **b** }

E’ → ε
First(**ε**) = { **ε** }

T’ → ε
First(**ε**) = { **ε** }

با توجه به قدم (۲) در الگوریتم
با توجه به قدم (۳) در الگوریتم

غیرپایانه‌ها	نشانه‌ی ورودی					
	id	**+**	*****	**(**	**)**	**$**
E	E → T E’			E → T E’		
E’		E’ → + T E’			E’ → ε	E’ → ε
T	T → F T’			T → F T’		
T’		T’ → ε	T’ → * F T’		T’ → ε	T’ → ε
F	F → id			F → (E)		

تمرین

بررسی کنید این گرامر **LL(1)** هست یا خیر

(جدول پارس را رسم کنید)

E → T E’

E’ → + T E’ | ε

T → F T’

T’ → * F T’ | ε

F → (E) | id| ε

گرامرهای LL(K)

در گرامرهای **LL(2)** ورودی از راست به چپ با دو توکن در هر مرحله خوانده میشود

مثال **S→ ab|ac**

این گرامر **LL(1)** نیست اما **LL(2)** هست.

اگر یک توکن **a** داشته باشیم نمیتوان قانون درست را نمیتوانیم تشخیص دهیم

اما اگر دو توکن بخوانیم میتوان قانون درست را تشخیص داد

بررسی گرامرهای LL(2)

توابع First’ و Follow’ شبیه First و Follow گرامرهای LL(1)

با این تفاوت که اشاره به دو پایانه مجاور دارند (به جای یک پایانه)

(نکته Follow’(S)=$$)

شرایط LL(2) بودن گرامر: برای قوانین به شکل A→α|β

1. اشتراک First’(α) و First’(β) تهی باشد
2. اگر α تک حرفی است، اشتراک First(α).Follow(A) و First’(β) تهی باشد
3. اگر ε از α مشتق می شود، آنگاه اشتراک First’(β) و Follow’(A) باید تهی باشد.
4. اگر β تک حرفی است و ε از α مشتق می شود، آنگاه اشتراک First(β).Follow(A) و Follow’(A) باید تهی باشد.

(نکته اگر گرامر مبهم باشد LL(K) نیست)

پردازش خطا

- شناسایی خطاها
- پیداکردن جایی که خطا در آن رخ داده است (مثلاً خطی از کد)
- اطلاع‌رسانی دقیق و روشن
- مواجهه با (یا عبور از) خطا برای ادامه‌ی کار و یافتن خطاهای احتمالی آینده
- در کامپایل برنامه‌های صحیح نباید تغییری ایجاد شود

راه‌بردهای مواجهه با خطا (۱)

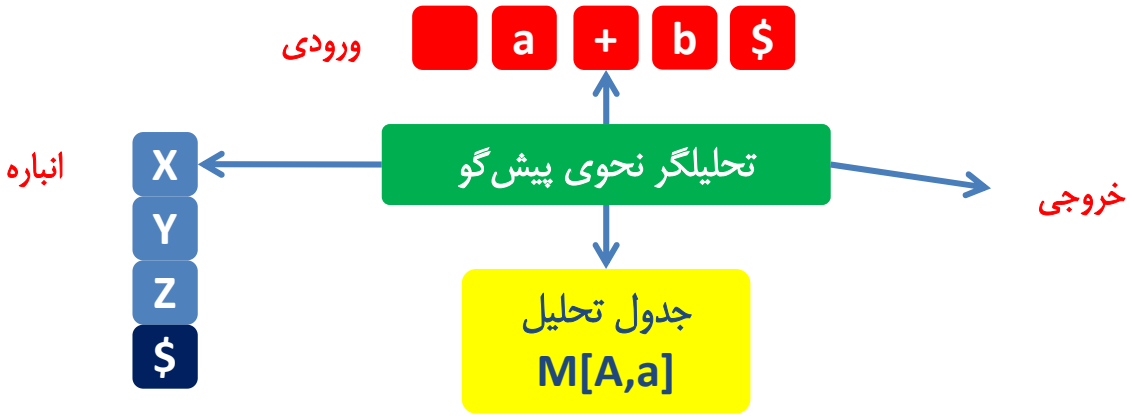
- حالت ترس (**Panic Mode**): دور انداختن نشانه‌ها تا جایی که به یک نشانه‌ی «همگام‌سازی» (**Synchronizing**) بربخوریم
  - نشانه‌های همگام‌سازی مثل: «**end**» و «**;**» و «**}**» در زبان‌های برنامه‌سازی
  - بسته به تصمیم طراح کامپایلر
  - کاستی‌ها:
    - دور انداختن ورودی باعث عدم تعریف صحیح (مثلاً تعریف متغیرها) و به این ترتیب ایجاد خطاهای بیش‌تر می‌شود
    - خطاهای احتمالی در بخشی که دور انداخته‌ایم شناسایی نمی‌شوند
  - مزایا:
    - سادگی
- سطح عبارت (**Phrase Level**): تصحیح محلی ورودی
  - مثلاً در برخورد با «**,**» به جای «**;**»، «**,**» حذف و «**;**» اضافه می‌شود
  - بسته به تصمیم طراح کامپایلر
  - برای همه‌ی خطاها مناسب نیست
  - می تواند به همراه حالت ترس استفاده می‌شود تا ورودی کم‌تری دور انداخته شود
  - هر گونه تغییر (مخصوصادر انباره) باید با احتیاط انجام شود

راه‌بردهای مواجهه با خطا (۲)

- قواعد خطا (Error Productions)
  - برای خطاهای متداول افزودن قواعدی به گرامر
  - ساخت یا تولید تحلیلگر نحوی با گرامر تقویت شده
  - مثلاً قاعده‌ای برای علامت := (انتساب در پاسکال) به گرامر زبان C می‌افزاییم
  - خطا گزارش داده می‌شود اما فرآیند کامپایل بدون وقفه ادامه می‌یابد (مزیت این روش: عدم وقفه کامپایلر)
- تصحیح سراسری (Global Correction)
  - افزودن، حذف کردن، یا جای‌گزینی نشانه‌ها نتایجی متفاوت دارد و ممکن است هر کدام به تغییرات زیادی منجر شود
  - الگوریتم‌هایی وجود دارند که می‌کوشند تا اقدامی در مقابل خطا انجام شود که این تغییرات را در سطح برنامه به حداقل برسانند

اصلاح خطا

- خطا در چه صورتی رخ می‌دهد؟ یادآوری تحلیلگر نحوی:



- (۱) در صورتی که **X** پایانه باشد و با ورودی انطباق نداشته باشد
- (۲) در صورتی که **M[X, ورودی]** خالی باشد

- دو روش اصلاح:
  - حالت ترس
  - سطح عبارت

اصلاح خطا: روش حالت ترس (۱)

- فرض کنید **A** یک غیرپایانه در بالای انباره باشد
- ایده: ورودی‌ها را دور بیاندازیم تا به یکی از نشانه‌های مجموعه‌ی از پیش‌تعریف‌شده‌ی همگام‌سازی برسیم
- انتخاب اعضای مجموعه‌ی همگام‌سازی مهم است؛ مثلاً:
  - مجموعه‌ی همگام‌سازی را **Follow(A)** در نظر بگیریم و ورودی‌ها را دور بیاندازیم تا به عضوی از این مجموعه برسیم. **A** را از بالای انباره برداریم و تحلیل نحوی را ادامه دهیم
  - مجموعه‌ی همگام‌سازی را **First(A)** در نظر بگیریم و ورودی‌ها را دور بیاندازیم تا به عضوی از این مجموعه برسیم. تحلیل نحوی را از همین‌جا ادامه دهیم
  - ممکن است بتوان از قاعده‌های منتهی به **ε** هم استفاده کرد
- اگر پایانه‌ای بالای انباره باشد که با ورودی نخواند، آن را از روی انباره برمی‌داریم و در پیغامی می‌گوییم که آن پایانه را اضافه کرده‌ایم

اصلاح خطا: روش حالت ترس (۲)

- ایده‌ی کلی: خانه‌های خالی جدول تحلیل را تغییر دهیم
  - اگر خانه‌ی **M[A,a]** خالی، و **a** عضو **Follow(A)** بود، **M[A,a]** را برابر **"sync"** (همگام‌سازی) قرار می‌دهیم
- بنابراین اگر **A** عنصر بالای انباره، و **a** ورودی فعلی باشد:
  - اگر **A** غیرپایانه، و **M[A,a]** خالی بود، **a** را از ورودی دور می‌اندازیم
  - اگر **A** غیرپایانه، و **M[A,a]** برابر **"sync"** بود، **A** را از روی انباره برمی‌داریم
  - اگر **A** پایانه، ولی نامساوی **a** بود، **A** را از روی انباره برمی‌داریم (در واقع به این معنی است که آن را در ورودی اضافه کرده‌ایم)

اصلاح خطا: روش حالت ترس (۳)

- مثال جدول تحلیل تغییریافته:

غیرپایانه‌ها	نشانه‌ی ورودی					
	id	+	*	(	)	$
E	E → T E’			E → T E’		
E’		E’ → + T E’			E’ → ε	E’ → ε
T	T → F T’			T → F T’		
T’		T’ → ε	T’ → * F T’		T’ → ε	T’ → ε
F	F → id			F → (E)		

همگام‌سازی (**"sync"**). بر اساس مجموعه‌ی **Follow**. غیرپایانه‌ی بالای انباره را برمی‌داریم

ورودی را دور می‌اندازیم

اصلاح خطا: روش حالت ترس (۴)

- مثال روند تغییریافته‌ی تحلیل نحوی:

انباره	ورودی	ملاحظات
$ E	+ id * + id $	خطا: دور انداختن +
$ E	id * + id $	
$ E’ T	id * + id $	
$ E’ T’ F	id * + id $	
$ E’ T’ id	id * + id $	
$ E’ T’	* + id $	
$ E’ T’ F *	* + id $	
$ E’ T’ F	+ id $	خطا: خانه‌ی M[F,+]=“sync”
$ E’ T’	+ id $	F از روی انباره برداشته می‌شود
$ E’	+ id $	
$ E’ T +	+ id $	
$ E’ T	id $	
$ E’ T’ F	id $	
$ E’ T’ id	id $	
$ E’ T’	$	
$ E’	$	
$	$	

پیغام خطای نمونه:
«+ به اشتباه واردشده است، و دور انداخته می‌شود»

پیغام خطای نمونه:
«جمله (Term) پیدا نشد»

نوشتن پیغام‌های خطا (۱)

- یادآوری: هر غیرپایانه، معرّف یک ساخت انتزاعی زبان است
- مثال‌هایی از پیغام‌های خطا برای گرامر ما:
  - اگر E معرّف «عبارت» باشد:
    - با فرض این که E روی انباره، و + ورودی فعلی است: «خطا در محل i: عبارت نمی‌تواند با ’+‘ آغاز شود» یا «خطا در محل i: عبارت نادرست است»
    - به طور مشابه برای E روی انباره، و * به عنوان ورودی فعلی
  - اگر E’ معرّف «انتهای عبارت» باشد:
    - با فرض این که E’ بالای انباره، و ورودی فعلی * یا id است: «خطا: عبارتی که در محل j شروع شده است، در محل i ظاهری نادرست دارد»
    - نکته: هر بار E را از روی انباره برمی‌داریم، محل فعلی را جایی ذخیره می‌کنیم (محلی که آخرین عبارت تمام شده است و عبارت جدید شروع شده است j)

نوشتن پیغام‌های خطا (۲)

- مثال‌هایی از پیغام خطا برای خانه‌ی **"sync"**
  - فرض کنید **F** بالای انباره، و **+** ورودی فعلی است:
    - «خطا در محل **i**: بر خلاف انتظار، عملوند پیدا نشد»
  - فرض کنید **E** بالای انباره، و **(** ورودی فعلی است:
    - «خطا در محل **i**: بر خلاف انتظار، عبارتی پیدا نشد»

نوشتن پیغام‌های خطا (۳)

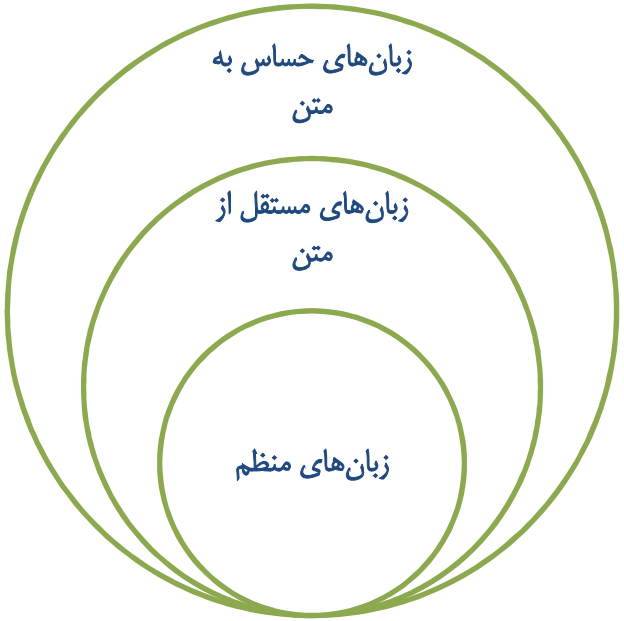
- مثال‌هایی از حالتی که بالای انباره پایانه‌ای است که با ورودی نمی‌خوانَد
  - فرض کنید **id** بالای انباره، و **+** ورودی فعلی است:
    - «خطا در محل **i**: بر خلاف انتظار، شناسه‌ای پیدا نشد»
  - فرض کنید **(** بالای انباره، و ورودی فعلی پایانه‌ای غیر از **(** است:
    - هر وقت به یک **)** برخوردیم، محل آن را در «انباره‌ی ویژه پرانتز باز» ذخیره می‌کنیم
    - وقتی که حالت فوق اتفاق افتاد، به انباره‌ی پرانتز باز نگاه می‌کنیم تا محل آن پرانتز باز را که بسته نشده است ، بیابیم
    - «خطا در محل **i**: برای پرانتز بازی که در محل **m** واقع شده، هیچ پرانتز بسته‌ای پیدا نشد» (مثلاً در صورتی که ورودی **$ (id id) + * id)** باشد)

تجمیع پیغام‌های خطا با جدول تحلیل

- با توجه به آن‌چه گفته شد، خانه‌های خالی جدول تحلیل را می‌توانیم با روش مناسب صدور پیغام خطا پر کنیم

حل مشکلات گرامر

- همه‌ی ویژگی‌های یک زبان برنامه‌سازی را نمی‌توان با گرامرها (زبان‌ها) ی مستقل از متن توصیف کرد، مثلاً:
  - تعریف یک شناسه (مثلاً نام متغیر) پیش از استفاده از آن در ادامه‌ی برنامه
  - رعایت تجانس نوع‌ها در عبارت‌ها (مثلاً جمع عدد صحیح با عدد صحیح)
  - تطابق پارامترها در زمان تعریف تابع، با آرگومان‌ها در زمان فراخوانی آن
- این ویژگی‌ها و نظایر آن‌ها «حساس به متن» اند و دسته‌ی دیگری از زبان‌ها را معرفی می‌کنند: زبان‌های حساس به متن



زبان‌های حساس به متن

- مثال:
  - تعریف شناسه پیش از استفاده

$L_1 = \{ w\,c\,w \mid w \in (a|b)^* \}$

  - تطابق پارامترها ($a^n b^m$) و آرگومان‌ها ($c^n d^m$)

$L_2 = \{a^n\, b^m\, c^n\, d^m \mid n \geq 1, m \geq 1 \}$

زبان‌های مستقل از متن (۱)

- مثال:

$L_3 = \{ w\, c\, w^R \mid w \in (a|b)^* \}$

$L_4 = \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1 \}$

$L_5 = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1 \}$

$L_6 = \{a^n b^n \mid n \geq 1 \}$