اصول طراحی کامپایلر

درس‌نامه‌ی ۶

کاظم فولادی
http://kazim.fouladi.ir

ویراست اول: ۱۳۸۸
ویراست دوم: ۱۳۹۳

فهرست مطالب

۶ تجزیه‌ی بالا به پایین: روش LL(1)

TOP-DOWN PARSING:LL(1) METHOD

۶-۱ مقدمه

همان طورکه در فصل قبل اشاره کردیم، در تجزیه‌ی بالا به پایین، درخت تجزیه از بالای درخت (ریشه) به سمت پایین (برگ‌ها) ساخته می‌شود. روش تجزیه‌ی نزولی بازگشتی، با فراخوانی روال‌های مناسب به صورت تو در تو و بازگشتی، درخت تجزیه را ایجاد می‌کرد و رشته‌ی مورد نظر را «پارس» می‌نمود.

مثال

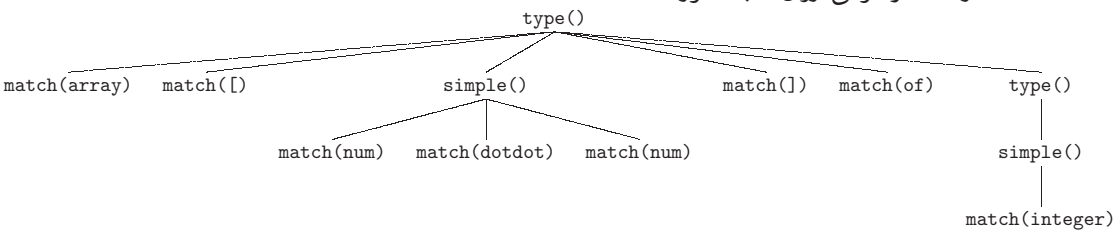
در تجزیه‌ی بالا به پایین پیشگوی بازگشتی برای گرامر

$$
\begin{aligned}
\mathit{type} &\rightarrow \mathit{simple} \\
&\mid \textbf{array}\ [\mathit{simple}]\ \textbf{of}\ \mathit{type} \\
\mathit{simple} &\rightarrow \textbf{integer} \mid \textbf{char} \mid \textbf{num dotdot num}
\end{aligned}
$$

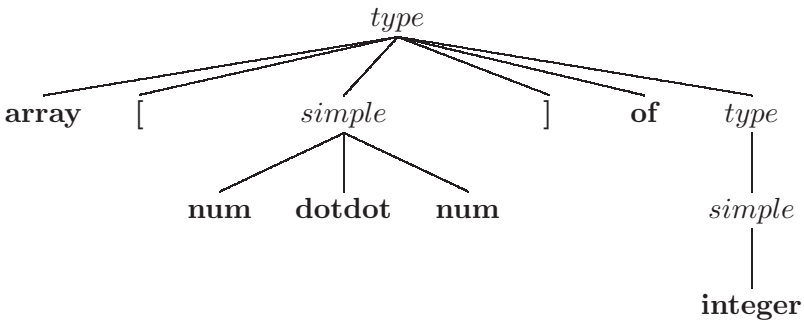
و رشته‌ی

```
array [ num dotdot num ] of integer
```

درخت فراخوانی روال‌ها به صورت



و درخت تجزیه به صورت زیر به دست می‌آید:



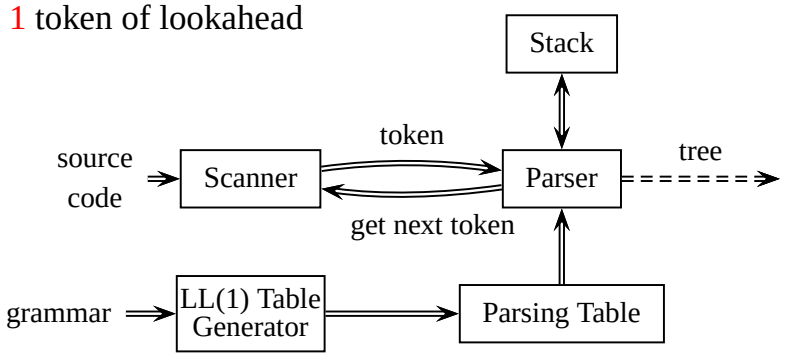
در روش موسوم به LL(1)، تجزیه‌ی بالا به پایین، به جای استفاده از روال‌های بازگشتی، به کمک یک جدول به نام جدول تجزیه (Parse Table) انجام می‌شود که در آن اطلاعات موجود در روال‌های بازگشتی کد می‌شود و در نتیجه ما را از فراخوانی‌های بازگشتی بی‌نیاز می‌کند.

به عبارت دیگر روش LL(1) همان روش نزولی بازگشتی است که به صورت غیر بازگشتی (به کمک یک حلقه‌ی تکرار) پیاده‌سازی می‌شود. بنابراین شرایط استفاده از این روش، همان شرایط روش نزولی بازگشتی است. به عبارت دیگر، گرامر مورد استفاده باید LL(1) باشد.

۶-۱-۱ مدل تجزیه‌گر

تجزیه‌گر (Parser)، ورودی خود را از خروجی پویشگر (Scanner) که همان توکن‌هاست، دریافت می‌کند. تجزیه‌گر در هر مرحله از طریق تابع get next token توکن بعدی را از پویشگر درخواست می‌کند. جدول تجزیه اطلاعات لازم در مورد گرامر زبان و چگونگی عمل تجزیه را در اختیار تجزیه‌گر قرار می‌دهد (شکل ۶-۱).

- Input parsed from left to right
- Leftmost derivation
- 1 token of lookahead



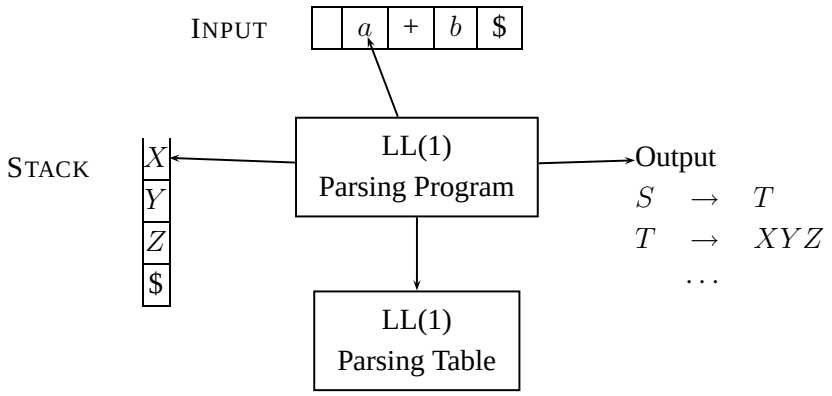
شکل ۶-۱: نمودار عمومی تجزیه‌گر

مدل تجزیه‌گر، در اصل همان مدل آتوماتون پشته‌ای (pushdown automaton: PDA) است که به عنوان پذیرنده‌ی زبان‌های مستقل از متن عمل می‌کند.

اجزای تجزیه‌گر LL(1)

همان طورکه در شکل ۶-۲ ملاحظه می‌شود، مطابق مدل، تجزیه‌گر LL(1) دارای چهار جزء است.

- **بافر ورودی (Input Buffer)** که محتوای آن به $ ختم می‌شود.
- **پشته‌ی تجزیه‌گر (Parser Stack)** که حاوی ترتیبی از پایانه‌ها و ناپایانه‌هاست.
- **جدول تجزیه (Parsing Table)** که یک آرایه‌ی دوبعدی است: $M[X, a]$.
- **برنامه‌ی تجزیه (Parsing Program)** که تجزیه‌گر توسط آن کنترل می‌شود.



شکل ۶-۲: مدل تجزیه‌گر LL(1)

۶-۲ روال تجزیه

برنامه‌ی تجزیه‌گر در قالب روال تجزیه کد می‌شود. این روال، الگوریتمی است که در هر گام با ملاحظه‌ی عنصر بالای پشته و توکن جاری، کنش (action) مناسب را مشخص می‌کند.

LL(1) Parsing Procedure

$push(\$S)$
repeat
  $a \leftarrow lookahead$
  $X \leftarrow top(Stack)$
  if $X = \$$ **and** $a = \$$ **then** $accept()$
  if $X = a$ **and** $a \neq \$$ **then** $pop();\ a \leftarrow nextToken()$
  if $X = A$ **and** $a \neq \$$ **then** $pop();push(reverse(M[X, a]))$
  else $error(M[X, a])$

۶-۳ روش ساخت جدول تجزیه

اطلاعات لازم برای روال تجزیه در مورد یک گرامر خاص، در قالب جدول تجزیه کد می‌شود. جدول تجزیه‌ی LL(1) یک ماتریس دو بعدی است که سطرهای آن را ناپایانه‌ها و ستون‌های آن را پایانه‌ها به علاوه‌ی نماد انتهای رشته تشکیل می‌دهند.

$$M : N \times (T \cup \{\$\}) \rightarrow P \cup \{\text{error}\}$$

روش ساخت این جدول با چهار قاعده‌ی زیر مشخص می‌شود:

$M[A, a] = \alpha$ **if** $a \in First(\alpha)$, $A \rightarrow \alpha \in P$, $\alpha \not\Rightarrow^* \epsilon$
$M[A, a] = \epsilon$ **if** $a \in Follow(A)$, $A \rightarrow \epsilon \in P$
$M[A, a] = \alpha$ **if** $a \in First(\alpha) \cup Follow(A)$, $A \rightarrow \alpha \in P$, $\alpha \Rightarrow^+ \epsilon$
$M[A, a] = \text{error}$ **if** otherwise

مثال

گرامر زیر را در نظر می‌گیریم:

$$\begin{aligned} E &\rightarrow TE' \\ E' &\rightarrow +TE' \mid \epsilon \\ T &\rightarrow FT' \\ T' &\rightarrow *FT' \mid \epsilon \\ F &\rightarrow \mathbf{id} \mid (E) \end{aligned}$$

برای ایجاد جدول تجزیه‌ی LL(1)، ابتدا محاسبه‌ی تابع $First$ را انجام می‌دهیم:

$$\begin{aligned} First(E) &= First(TE') = \{\mathbf{id}, (\} \\ First(E') &= First(+TE') \cup \{\epsilon\} = \{+, \epsilon\} \\ First(+TE') &= \{+\} \\ First(T) &= First(FT') = \{\mathbf{id}, (\} \\ First(T') &= First(*FT') \cup \{\epsilon\} = \{*, \epsilon\} \\ First(*FT') &= \{*\} \\ First(\mathbf{id}) &= \{\mathbf{id}\} \\ First((E)) &= \{(\} \\ First(F) &= First(\mathbf{id}) \cup First((E)) = \{\mathbf{id}, (\} \end{aligned}$$

سپس محاسبه‌ی تابع $Follow$:

$$\begin{aligned} Follow(E) &= \{), \$\} \\ Follow(E') &= \{), \$\} \\ Follow(T) &= \{+,), \$\} \\ Follow(T') &= \{+,), \$\} \\ Follow(F) &= \{+, *,), \$\} \end{aligned}$$

با توجه به قواعد موجود برای ساخت جدول، جدول تجزیه به صورت زیر خواهد بود:

	id	+	*	(	)	$
E	TE'			TE'		
E'		$+TE'$			ϵ	ϵ
T	FT'			FT'		
T'		ϵ	$*FT'$		ϵ	ϵ
F	**id**			(E)		

در جدول حاصل، خانه‌های خالی، نشان‌دهنده‌ی محل‌های وقوع خطاست.

مثال

این مثال، تجزیه رشته‌ی

$$\mathbf{id} + \mathbf{id} * \mathbf{id}$$

را با جدول تجزیه‌ی مثال قبلی با استفاده از روال تجزیه‌ی LL(1) نشان می‌دهد:

Stack	Input	Output
$\$E$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	$E \to TE'$
$\$E'T$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	$T \to FT'$
$\$E'T'F$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	$F \to \mathbf{id}$
$\$E'T'\mathbf{id}$	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	
$\$E'T'$	$+\mathbf{id} * \mathbf{id}\$$	$T' \to \epsilon$
$\$E'$	$+\mathbf{id} * \mathbf{id}\$$	$E' \to +TE'$
$\$E'T+$	$+\mathbf{id} * \mathbf{id}\$$	
$\$E'T$	$\mathbf{id} * \mathbf{id}\$$	$T \to FT'$
$\$E'T'F$	$\mathbf{id} * \mathbf{id}\$$	$F \to \mathbf{id}$
$\$E'T'\mathbf{id}$	$\mathbf{id} * \mathbf{id}\$$	
$\$E'T'$	$*\mathbf{id}\$$	$T' \to *FT'$
$\$E'T'F*$	$*\mathbf{id}\$$	
$\$E'T'F$	$\mathbf{id}\$$	$F \to \mathbf{id}$
$\$E'T'\mathbf{id}$	$\mathbf{id}\$$	
$\$E'T'$	$\$$	$T \to \epsilon$
$\$E'$	$\$$	$E \to \epsilon$
$\$$	$\$$	accept

در هر مرحله، روال تجزیه با توجه به نماد بالای پشته و توکن جاری، کنش مناسب را انجام می‌دهد. خروجی ترتیب استفاده از قواعد گرامر برای ایجاد رشته‌ی ورودی را نشان می‌دهد.

۴-۶ تداخل در جدول تجزیه

اگر گرامری LL(1) نباشد، جدول تجزیه‌ی آن دارای تداخل خواهد بود و برعکس. گاهی با وجود LL(1) نبودن یک گرامر، می‌توان با اصلاح جدول تجزیه، امکان تجزیه‌ی آن را با روش LL(1) فراهم کرد.

مثال

گرامر زیر برای ساختار شرطی LL(1) نیست، پس در جدول تجزیه‌ی آن تداخل وجود خواهد داشت:

$$\begin{aligned} S &\to \mathbf{if}\ E\ \mathbf{then}\ S\ S' \mid \mathbf{other} \\ S' &\to \mathbf{else}\ S \mid \epsilon \\ E &\to \mathbf{expr} \end{aligned}$$

	if	then	other	else	expr	$
S	**if** E **then** S S'		**other**			
S'				**else** S ϵ		ϵ
E					**expr**	

ملاحظه می‌شود که خانه‌ی $M[S', \textbf{else}]$ دارای تداخل است.

تکنیک موردی: برای **تطبیق هر else با نزدیک‌ترین then**، گزینه‌ی S **else** را انتخاب می‌کنیم.

۶-۵ روال ساخت تجزیه‌گر پیشگوی LL(1)

در یک جمع‌بندی، برای ایجاد تجزیه‌گر پیشگوی LL(1) بایستی پنج مرحله‌ی زیر را دنبال کنیم:

۱) بررسی LL(1) بودن گرامر

۲) تبدیل به گرامر معادل LL(1) در صورت LL(1) نبودن

۳) محاسبه‌ی تابع $First$ برای ناپایانه‌ها و سمت راست‌های قواعد

۴) محاسبه‌ی تابع $Follow$ برای ناپایانه‌ها

۵)
- ساخت جدول تجزیه‌ی LL(1) (برای تجزیه‌گر غیربازگشتی)
- کد کردن روال‌های لازم: برای هر ناپایانه یک روال + روال match() (برای تجزیه‌گر بازگشتی)

◄ تمرین

۱. گرامر زیر را در نظر بگیرید:

$$\begin{aligned} S &\rightarrow (L) \mid a \\ L &\rightarrow L, S \mid S \end{aligned}$$

آ) یک گرامر LL(1) معادل با گرامر فوق به دست آورید.

ب) برای گرامر حاصل یک تجزیه‌گر LL(1) بسازید.

پ) رفتار تجزیه‌گر فوق را بر روی رشته‌های زیر نشان دهید:

(a, a) $\qquad$ $(a, ((a, a), (a, a)))$

۲. یک تجزیه‌گر LL(1) برای گرامر زیر بسازید:

$$\begin{aligned} bexpr &\rightarrow bexpr \textbf{ or } bterm \mid bterm \\ bterm &\rightarrow bterm \textbf{ and } bfactor \mid bfactor \\ bfactor &\rightarrow \textbf{not } bfactor \mid (bexpr) \mid \textbf{true} \mid \textbf{false} \end{aligned}$$

۳. گرامر زیر عبارت‌های منظم بر روی الفبای $\{a, b\}$ را تولید می‌کند:

$$R \rightarrow R + R \mid RR \mid R^* \mid (R) \mid a \mid b$$

یک تجزیه‌گر LL(1) برای گرامر غیر مبهم معادل با گرامر فوق بسازید.

۴. نشان دهید گرامری که آلترناتیوهای حداقل یکی از قواعد آن دارای پیشوند مشترک باشد، LL(1) نیست.

۵. نشان دهید گرامری که LL(1) باشد، نمی‌تواند مبهم باشد.

۶. نشان دهید گرامری که بازگشتی از چپ باشد، نمی‌تواند LL(1) باشد.

۷. نشان دهید هر گرامر بدون قواعد تهی (ϵ) که در آن هر آلترناتیو یک ناپایانه با یک پایانه‌ی متمایز آغاز شود، همواره LL(1) است.

۸. نشان دهید که هر گرامر منظم، یک گرامر LL(1) معادل دارد.

۹. برای ساختار switch-case زیر:

```
switch(id)
{
case num1:
 stmts
case num2:
 stmts
...
default:
 stmts
}
```

آ) یک گرامر LL(1) ارائه دهید. توضیح اینکه هر switch می‌تواند صفر یا بیشتر case داشته باشد و حداکثر دارای یک default در آخر است. برای راحتی فرض کنید که stmts و id پایانه هستند.

ب) رشته‌ی زیر را به روش LL(1) و به کمک جدول تجزیه‌ای که به دست می‌آورید، تجزیه (parse) کنید:

```
switch(id){ case 1: stmts default: stmts }
```