اصول طراحی کامپایلر

درس‌نامه‌ی ۹

کاظم فولادی
http://kazim.fouladi.ir

ویراست اول: ۱۳۸۸
ویراست دوم: ۱۳۹۳

فهرست مطالب

۹ تجزیه‌ی پایین به بالا: روش LR ۱

۹-۱ مقدمه: روش‌های LR ۱

۹-۲ روش LR ساده (SLR) ۱
۹-۲-۱ آیتم LR(0) ۱
۹-۲-۲ آتوماتون LR(0) ۲
عملیات بستار (Closure) ۲
عملیات برو به (Goto) ۳
ساخت DFAی LR(0) ۴
۹-۲-۳ مدل تجزیه‌گر LR در حالت کلی ۶
۹-۲-۴ الگوریتم تجزیه‌ی LR در حالت کلی ۶
۹-۲-۵ ساخت جدول تجزیه ۷
۹-۲-۶ گرامرهای غیر LR ساده (غیر SLR) ۱۳

۹-۳ روش LR قانون‌مند (CLR) ۱۴
۹-۳-۱ آیتم LR(1) ۱۴
۹-۳-۲ آتوماتون LR(1) ۱۵
عملیات بستار (Closure) ۱۵
عملیات برو به (Goto) ۱۶

ساخت DFAی LR(1) ۱۶
۹-۳-۳ ساخت جدول تجزیه‌ی LR قانون‌مند (CLR) ۱۷

۹-۴ روش LR با نماد پیش‌بینی (LALR) **۱۸**
۹-۴-۱ بروز تداخل کاهش ـ کاهش در اثر ادغام حالت‌ها ۲۰
۹-۴-۲ مقایسه‌ی جدول تجزیه‌ی LALR(1) با جدول تجزیه‌ی SLR(1) ۲۰
۹-۴-۳ رفتار تجزیه‌گر LALR در مقابل تجزیه‌گر CLR: مقایسه ۲۱

۹-۵ تداخل‌ها **۲۱**

۹-۶ مطالب تکمیلی **۲۲**
۹-۶-۱ استفاده از گرامرهای مبهم ۲۲
۹-۶-۲ نسبت خانواده گرامرهای بررسی شده ۲۶
نسبت خانواده گرامرها و زبان‌های LR با یکدیگر ۲۶

❊ تمرین **۲۷**

۹ تجزیه‌ی پایین به بالا: روش LR

BOTTOM-UP PARSING: LR METHODS

۹-۱ مقدمه: روش‌های LR

گرامرهای LR عمومی‌ترین خانواده از گرامرها هستند که بدون عقبگرد قابل تجزیه (parse) می‌باشند.
گرامر G، $\mathrm{LR}(k)$ است اگر در هر مرحله از پویش رشته از سمت چپ به راست، بتوان تنها با نگاه کردن به k توکن ورودی جاری، قاعده‌ی به کار رفته در اشتقاق راست‌ترین در آن گام را تعیین نمود.

$\mathbf{L}$	$\mathbf{R}$	(k)
↙	↓	↘
ورودی از چپ به راست دریافت می‌شود Left-to-right scanning	اشتقاق‌های راست‌ترین ساخته می‌شوند Right-most derivation	k توکن در هر مرحله

گرامرهای LL زیرمجموعه‌ی محضی از گرامرهای LR هستند.
روش‌های تجزیه‌ی LR، به ترتیب قدرت به سه روش SLR، LALR و CLR تقسیم می‌شوند.

۹-۲ روش LR ساده (SLR)

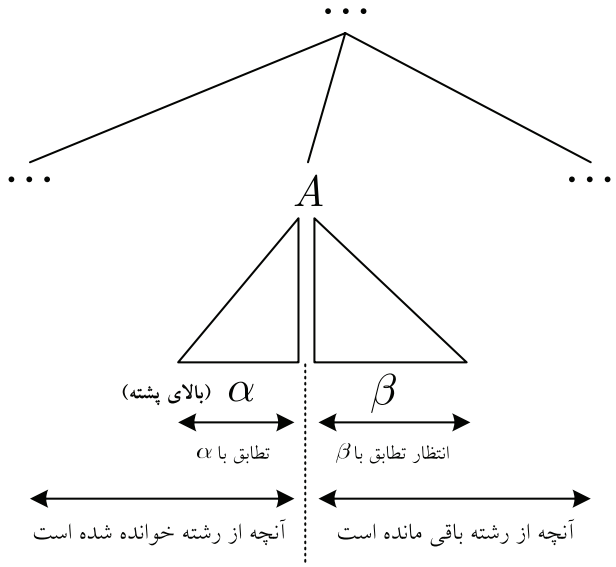
ابتدایی‌ترین روش تجزیه‌ی LR، روش SLR (LR ساده: simple LR) می‌باشد. برای بیان این روش، مقدمات و تعریف‌های لازم را پی می‌گیریم.

۹-۲-۱ آیتم LR(0)

یک **آیتم LR(0)** از گرامر G، یک قاعده‌ی تولید از G است که در یک مکان از سمت راست آن یک **نقطه** قرار گرفته است.

$$A \to \alpha \bullet \beta, \qquad \eta = \alpha\beta,\ A \to \eta \in P(G)$$

منظور از آیتم $A \to \alpha \bullet \beta$ این است که در فرم جمله‌ای جاری، آنچه تاکنون از رشته خوانده شده است با بخش قبل از نقطه تطابق یافته است و انتظار می‌رود ادامه‌ی ورودی با بخش بعد از نقطه تطابق یابد.



شکل ۹-۱: تعبیر شماتیک از آیتم LR(0) به صورت $A \to \alpha \bullet \beta$

مثال

قاعده‌ی تولید $A \to XYZ$ چهار آیتم LR(0) را تولید می‌کند.

$$A \to \bullet XYZ$$
$$A \to X \bullet YZ$$
$$A \to XY \bullet Z$$
$$A \to XYZ\bullet$$

نکته: قاعده‌ی تولید $A \to \epsilon$ تنها یک آیتم LR(0) ($A \to \bullet$) را تولید می‌کند.

۹-۲-۲ آتوماتون LR(0)

در روش‌های LR، دستگیره در هر مرحله به کمک یک آتوماتون (DFA) شناسایی می‌شود. از آیتم‌ها برای ساخت این DFA استفاده می‌گردد.

- آیتم‌هایی که وضعیت یکسانی را نمایش می‌دهند در یک گروه دسته‌بندی می‌شود.
- هر کدام از این مجموعه‌ها یک حالت از DFA را نشان می‌دهد.

عملیات بستار (Closure)

اگر I یک مجموعه از آیتم‌های LR(0) گرامر G باشد، CLOSURE(I) نیز یک مجموعه‌ی از آیتم‌هاست که به صورت زیر محاسبه می‌شود:

۱) در ابتدا I به CLOSURE(I) اضافه می‌شود.

۲) **اگر** $A \to \alpha \bullet B\beta \in \text{CLOSURE}(I)$ (نقطه قبل از یک ناپایانه‌ی B) و
قاعده‌ی $B \to \gamma$ در G موجود بود،
آن‌گاه آیتم $B \to \bullet\gamma$ به $\text{CLOSURE}(I)$ اضافه می‌شود.

۳) قاعده‌ی فوق را آن‌قدر تکرار می‌کنیم تا دیگر چیزی به $\text{CLOSURE}(I)$ اضافه نشود.

اگر $A \to \alpha \bullet B\beta \in \text{CLOSURE}(I)$، انتظار داریم که ادامه‌ی ورودی از $B\beta$ قابل اشتقاق باشد.
حال اگر $B \to \gamma$ در گرامر موجود باشد، بدیهی است که می‌توان انتظار داشت که ادامه‌ی ورودی از γ قابل اشتقاق باشد.

اگر در یکی از آیتم‌های مجموعه‌ی I نقطه قبل از یک ناپایانه قرار داشت، کلیه‌ی آیتم‌های حاصل از قواعد آن ناپایانه با قرارگرفتن نقطه در ابتدای سمت راست آنها به $\text{CLOSURE}(I)$ اضافه می‌شود.

مثال

برای گرامر زیر، نمونه‌ای از محاسبه‌ی تابع CLOSURE را مشاهده می‌کنیم:

$$
\begin{array}{lcl}
E' & \to & E\$ \\
E & \to & E + T \\
E & \to & T \\
T & \to & T * F \\
T & \to & F \\
F & \to & (E) \\
F & \to & \mathbf{id}
\end{array}
\qquad
\begin{array}{rcl}
\text{CLOSURE}(\{E' \to \bullet E\$\}) & = & \{E' \to \bullet E\$, \\
 & & E \to \bullet E + T, \\
 & & E \to \bullet T, \\
 & & T \to \bullet T * F, \\
 & & T \to \bullet F, \\
 & & F \to \bullet(E), \\
 & & F \to \bullet\mathbf{id}\}
\end{array}
$$

عملیات برو به (Goto)

اگر I مجموعه‌ای از آیتم‌ها و X یک نماد گرامر باشد،
در این صورت $\text{GOTO}(I, X)$ بستار مجموعه‌ی همه‌ی آیتم‌ها به صورت $A \to \alpha X \bullet \beta$ است که در آن $A \to \alpha \bullet X\beta$ در I قرار دارد.

$$A \to \alpha \bullet X\beta \in I \quad \Rightarrow \quad \text{CLOSURE}(\{A \to \alpha X \bullet \beta\}) \subseteq \text{GOTO}(I, X)$$

اگر I مجموعه‌ی آیتم‌های مجاز برای پیشوند γ از یک فرم جمله‌ای راست باشد،
در این صورت $\text{GOTO}(I, X)$ مجموعه‌ی آیتم‌های مجاز برای پیشوند γX است.

کلیه‌ی آیتم‌های موجود در مجموعه‌ی I که در آنها نقطه قبل از X قرارگرفته است، با انتقال نقطه به بعد از X به $I' = \text{GOTO}(I, X)$ اضافه می‌شوند و بستار مجموعه‌ی جدید I' محاسبه می‌شود.

مثال

نمونه‌ای از محاسبه‌ی تابع برو به برای گرامر زیر را مشاهده می‌کنید:

$$
\begin{array}{lcl}
E' & \rightarrow & E\$ \\
E & \rightarrow & E + T \\
E & \rightarrow & T \\
T & \rightarrow & T * F \\
T & \rightarrow & F \\
F & \rightarrow & (E) \\
F & \rightarrow & \mathbf{id}
\end{array}
$$

$$
\begin{aligned}
&\textsc{goto}(\{E' \rightarrow E \bullet \$, E \rightarrow E \bullet + T\}, +) = \\
&\textsc{closure}(\{E' \rightarrow E + \bullet T\}) = \{ \\
&E \rightarrow E + \bullet T, \\
&T \rightarrow \bullet T * F, \\
&T \rightarrow \bullet F, \\
&F \rightarrow \bullet (E), \\
&F \rightarrow \bullet \mathbf{id}\}
\end{aligned}
$$

ساخت DFAی LR(0)

هر حالت DFA حاوی یک مجموعه آیتم است و به صورت زیر ساخته می‌شود:

- حالت شروع $\textsc{closure}(\{S' \rightarrow \bullet S\$\})$ است.
- به ازای هر نماد X که در یکی از آیتم‌های مجموعه‌ی مربوط به حالت I به صورت

$$A \rightarrow \alpha \bullet X\beta$$

قرار دارد، بین I و $\textsc{goto}(I, X)$ یک گذر با برچسب X ایجاد می‌شود.

- حالت نهایی DFA، حالتی است که در آن آیتم $S' \rightarrow S \bullet \$$ قرار دارد (نقطه به انتها رسیده است).

آتوماتون LR(0) پیشوندهای نموپذیر یک گرامر را نشان می‌دهد.
پیشوند نموپذیر (viable prefix): پیشوند یک فرم جمله‌ای راست که می‌تواند در بالای پشته‌ی تجزیه‌گر ظاهر شود.

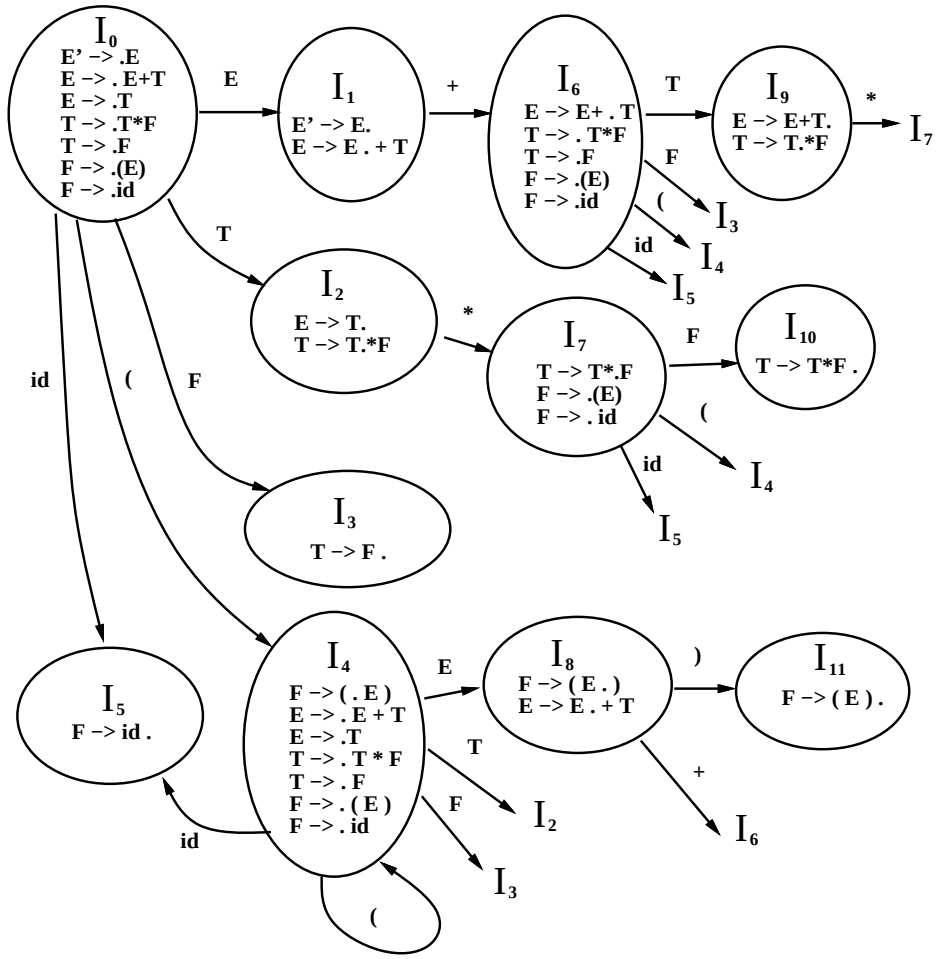
به عبارت دیگر می‌توان گفت، چنانچه برای هر حالت DFA حاصل، عبارت منظم متناظر را بنویسیم (یعنی مجموعه‌ی رشته‌هایی که با شروع از حالت آغازین تا آن حالت قابل مشاهده هستند)، در این صورت این عبارت منظم نشان می‌دهد که هرگاه در بالای پشته شماره‌ی آن حالت قرار داشت، چه پیشوندهایی از فرم جمله‌ای راست می‌تواند در زیر پشته وجود داشته باشد. در واقع، شماره‌ی حالت، کلیه‌ی اطلاعات لازم در مورد زیر پشته را در خود کد می‌کند.

مثال

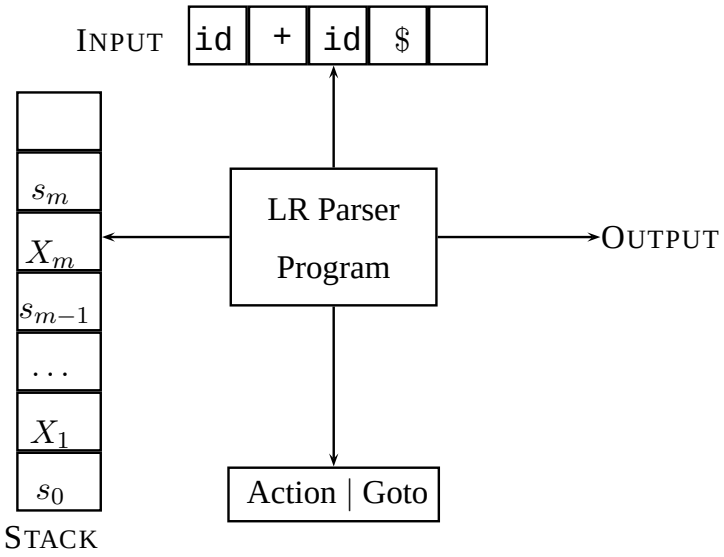
برای گرامر:

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

آتوماتون LR(0) به صورت زیر به دست می‌آید:



۹-۲-۳ مدل تجزیه‌گر LR در حالت کلی



شکل ۹-۲: مدل تجزیه‌گر LR

اجزای تجزیه‌گر LR عبارتند از:

- **بافر ورودی (Input Buffer)** که محتوای آن به $ ختم می‌شود.
- **پشته‌ی تجزیه‌گر (Parser Stack)** که حاوی ترتیبی از پایانه‌ها، ناپایانه‌ها و حالت‌هاست.
- **جدول تجزیه (Parsing Table)** که متشکل از دو بخش ACTION و GOTO است.
- **برنامه‌ی تجزیه (Parsing Program)** که تجزیه‌گر توسط آن کنترل می‌شود.

۹-۲-۴ الگوریتم تجزیه‌ی LR در حالت کلی

محتوای پشته رشته‌ای به صورت $s_0 X_1 s_1 \dots s_{m-1} X_m s_m$ است که در آن:

- X_i یک نماد گرامر است؛
- s_i یک حالت (state) است که اطلاعات پشته در زیر خود را خلاصه می‌کند.

ترکیب **حالت بالای پشته** و **توکن جاری** به عنوان اندیس جدول تجزیه استفاده می‌شود.
پیکربندی (configuration) یک تجزیه‌گر LR، یک زوج است که از ترکیب محتوای پشته و باقیمانده‌ی ورودی ساخته می‌شود:

$$(s_0 X_1 s_1 \dots s_{m-1} X_m s_m, a_i a_{i+1} \dots a_n \$)$$

هر گام از عملیات تجزیه‌گر، بر اساس زوج (s_m, a_i) و مطابق جدول تجزیه مشخص می‌شود:

- اگر

$$\text{ACTION}[s_m, a_i] = \text{shift}(s_j)$$

آن‌گاه عمل شیفت انجام می‌شود: شیفت a_i به بالای پشته و قرار دادن s_j در بالای آن:

$$(s_0 X_1 s_1 \dots s_{m-1} X_m s_m, a_i a_{i+1} \dots a_n\$) \mapsto (s_0 X_1 s_1 \dots s_{m-1} X_m s_m a_i s_j, a_{i+1} \dots a_n\$)$$

- اگر

$$\text{ACTION}[s_m, a_i] = \text{reduce}(A \to \beta)$$

آن‌گاه عمل کاهش انجام می‌شود: اگر $s = \text{GOTO}(s_{m-r}, A)$ و $r = |\beta|$، تجزیه‌گر $2r$ نماد از بالای پشته حذف می‌کند (r حالت و r نماد گرامر) و A و سپس s را در بالای پشته قرار می‌دهد:

$$(s_0 X_1 s_1 \dots s_{m-1} X_m s_m, a_i a_{i+1} \dots a_n\$) \mapsto (s_0 X_1 s_1 \dots s_{m-1} X_{m-r} s_{m-r} A s, a_i a_{i+1} \dots a_n\$)$$

- اگر

$$\text{ACTION}[s_m, a_i] = \text{error}$$

تجزیه‌گر با یک خطا مواجه می‌شود و باید با خطا برخورد کند.

- اگر

$$\text{ACTION}[s_m, \$] = \text{accept}$$

تجزیه‌گر به طور موفق توقف می‌کند.

۹-۲-۵ ساخت جدول تجزیه

جدول تجزیه‌ی SLR، عملکرد DFA را نمایش می‌دهد که با استفاده از قواعد زیر ساخته می‌شود:

- اگر $\text{GOTO}(I_i, a) = I_j$ در این صورت $\text{ACTION}[i, a] = \text{shift}(j)$.
- اگر $A \to \alpha\bullet \in I_i$، برای هر $a \in Follow(A)$ داریم $\text{ACTION}[i, a] = \text{reduce}(A \to \alpha)$.
- اگر $S' \to S \bullet \$ \in I_i$، داریم $\text{ACTION}[i, \$] = \text{accept}$.
- اگر $\text{GOTO}(I_i, A) = I_j$، داریم $\text{GOTO}[i, A] = j$.
- اگر خانه‌ای از جدول با قواعد فوق پر نشد، آن خانه حاوی error است.
- حالت آغازین، حالتی است که آیتم $S' \to \bullet S\$$ را در خود دارد.

> اگر جدول تجزیه برای گرامری حاوی تداخل باشد، آن گرامر SLR نیست.

مثال

برای گرامر

$$\begin{aligned}
E' &\to E\$ && (0)\\
E &\to E + T && (1)\\
E &\to T && (2)\\
T &\to T * F && (3)\\
T &\to F && (4)\\
F &\to (E) && (5)\\
F &\to \mathbf{id} && (6)
\end{aligned}$$

جدول تجزیه‌ی SLR به صورت زیر به دست می‌آید:

	Action						Goto		
State	**id**	+	*	(	)	$	*E*	*T*	*F*
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

در بازنمایی جدول فوق، قراردادهای زیر به کار رفته است:

$$sn \equiv \text{shift}(n) \qquad rm \equiv \text{reduce}(m),\ m = \text{Rule-Number}(A \to \beta)$$

مثال

تجزیه‌ی LR رشته‌ی $\mathbf{id} * \mathbf{id} + \mathbf{id}$ با استفاده از جدول و گرامر مثال قبل، به صورت زیر اجرا می‌شود:

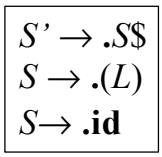
Stack	Input	Action
0	$\mathbf{id} * \mathbf{id} + \mathbf{id}\$$	shift
0 **id** 5	$*\mathbf{id} + \mathbf{id}\$$	reduce($F \to \mathbf{id}$)
0 F 3	$*\mathbf{id} + \mathbf{id}\$$	reduce($T \to F$)
0 T 2	$*\mathbf{id} + \mathbf{id}\$$	shift
0 T 2 $*$ 7	$\mathbf{id} + \mathbf{id}\$$	shift
0 T 2 $*$ 7 **id** 5	$+\mathbf{id}\$$	reduce($F \to \mathbf{id}$)
0 T 2 $*$ 7 F 10	$+\mathbf{id}\$$	reduce($T \to T * F$)
0 T 2	$+\mathbf{id}\$$	reduce($E \to T$)
0 E 1	$+\mathbf{id}\$$	shift
0 E 1 + 6	$\mathbf{id}\$$	shift
0 E 1 + 6 **id** 5	$\$$	reduce($F \to \mathbf{id}$)
0 E 1 + 6 F 3	$\$$	reduce($T \to F$)
0 E 1 + 6 T 9	$\$$	reduce($E \to E + T$)
0 E 1	$\$$	accept

مثال

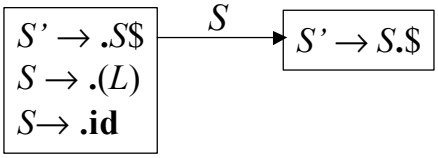
ساخت آتوماتون LR(0) و جدول تجزیه‌ی SLR(1) برای یک گرامر نمونه، به صورت گام به گام نشان داده می‌شود.

$$\begin{array}{lcl} S' & \rightarrow & S\$ \\ S & \rightarrow & (L) \\ S & \rightarrow & \mathbf{id} \\ L & \rightarrow & S \\ L & \rightarrow & L, S \end{array}$$

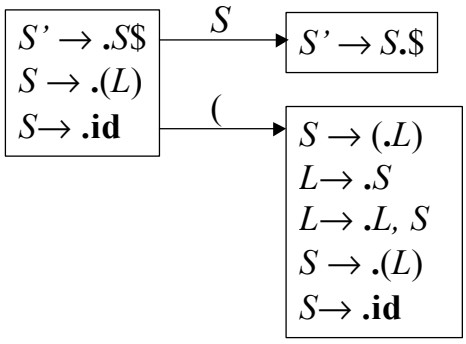
نخستین حالت آتوماتون، از بستار آیتم $S' \rightarrow \bullet S\$$ به دست می‌آید:



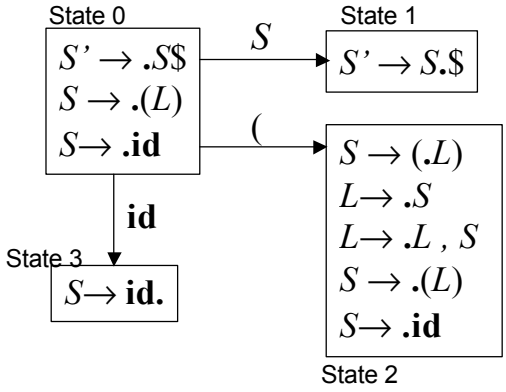
حال باید گذرها را از نخستین حالت به حالت‌های بعدی ایجاد کرد. این کار برای کلیه‌ی نمادهای گرامر که در حالت نخست پیش از آنها نقطه قرارگرفته است، انجام می‌شود. ابتدا برای S:



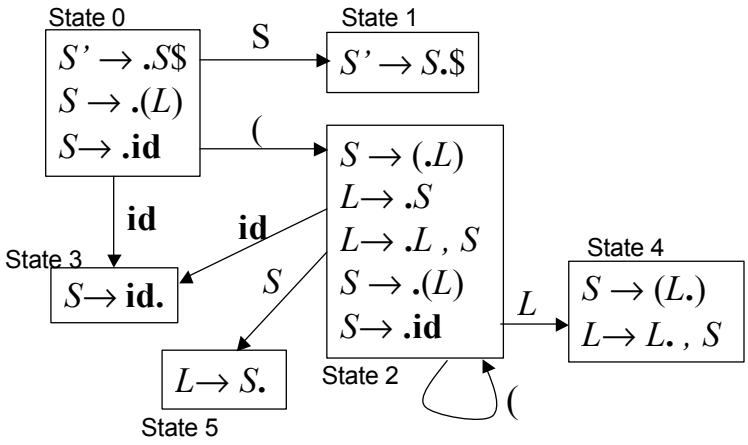
و سپس):



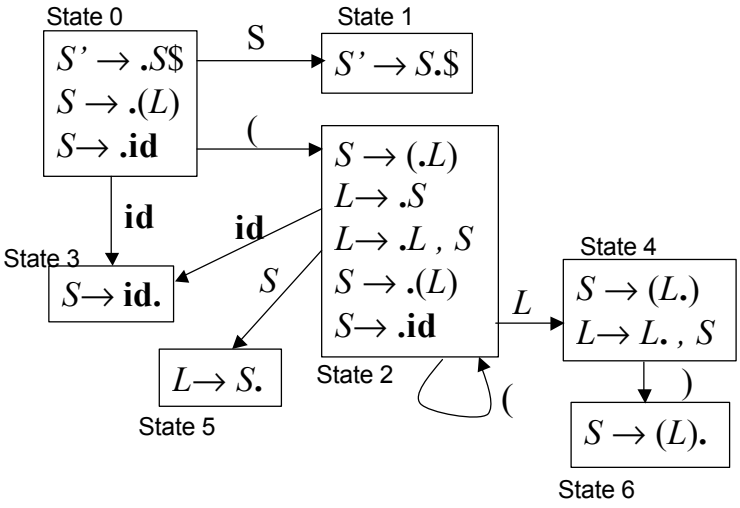
و همچنین **id**:

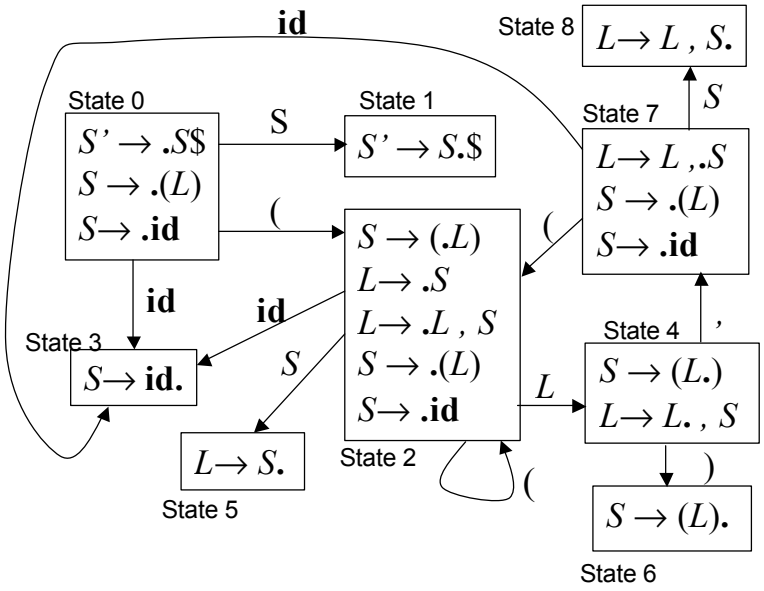


این فرایند برای سایر حالت‌ها نیز تکرار می‌شود. برای State 2 داریم:

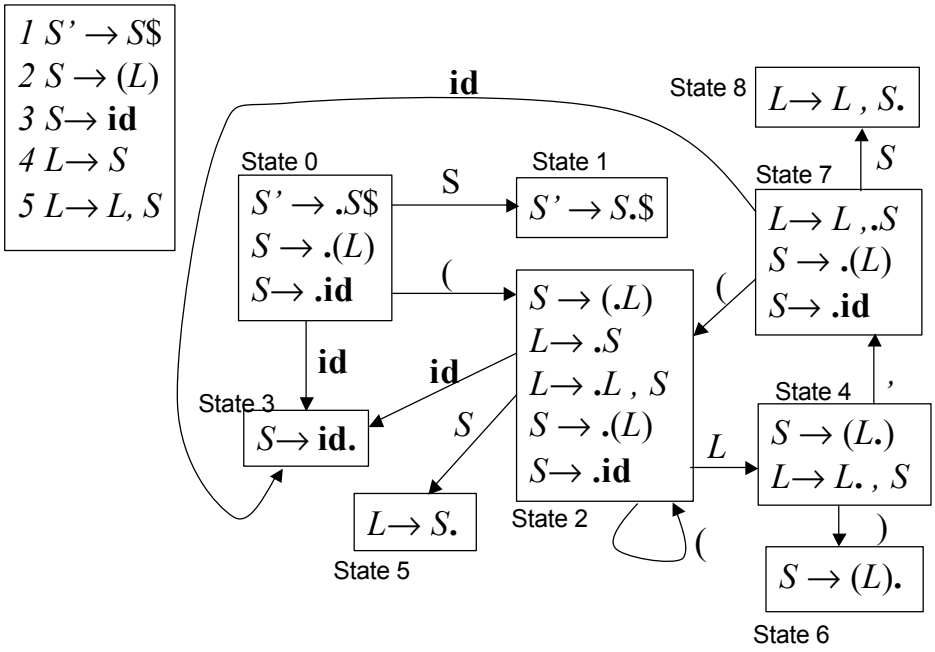


و سپس برای حالت State 4:





در نهایت به آتوماتون زیر می‌رسیم:



با شماره‌گذاری قواعد گرامر به صورت زیر،

$$
\begin{array}{lcll}
S' & \rightarrow & S\$ & (1) \\
S & \rightarrow & (L) & (2) \\
S & \rightarrow & \mathbf{id} & (3) \\
L & \rightarrow & S & (4) \\
L & \rightarrow & L, S & (5)
\end{array}
$$

جدول تجزیه‌ی SLR زیر را به دست می‌آوریم:

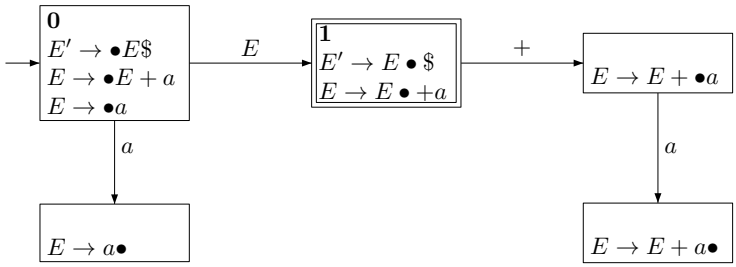
	Action					Goto	
State	(	)	**id**	,	$	S	L
0	s2		s3			1	
1					acc		
2	s2		s3			5	4
3		r3		r3	r3		
4		s6		s7			
5		r4		r4			
6		r2		r2	r2		
7	s2		s3			8	
8		r5		r5			

مثال

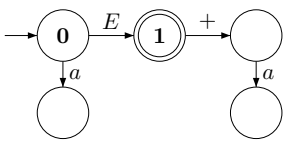
گرامر زیر را در نظر می‌گیریم:

$$E' \to E\$, \qquad E \to E + a, \qquad E \to a$$

آتوماتون LR(0) برای این گرامر به صورت زیر به دست می‌آید:



که شکل ساده شده‌ی آن به صورت زیر است:



عبارت منظم متناظر با هر یک از حالت‌های این آتوماتون مطابق جدول زیر می‌باشد:

(0) ϵ
(1) E
(2) a
(3) $E+$
(4) $E + a$

که کلیه‌ی پیشوندهای قابل رشد این گرامر را نشان می‌دهد. به عبارت دیگر، وقتی در بالای پشته، شماره‌ی یکی از این حالت‌ها قرارگیرد، نشان می‌دهد که در زیر پشته، چه پیشوندی از فرم جمله‌ای می‌تواند ظاهر شود. یعنی با دانستن شماره‌ی حالت جاری، وضعیت فعلی زیر پشته را می‌دانیم.

۹-۲-۶ گرامرهای غیر LR ساده (غیر SLR)

همان طور که گفته شد، اگر جدول تجزیه برای گرامری حاوی تداخل باشد، آن گرامر SLR نیست.

مثال

گرامر و آتوماتون LR(0) زیر را در نظر بگیرید:

Augmented Grammar

$$S' \rightarrow S$$
$$S \rightarrow L = R$$
$$S \rightarrow R$$
$$L \rightarrow *R$$
$$L \rightarrow \text{id}$$
$$R \rightarrow L$$

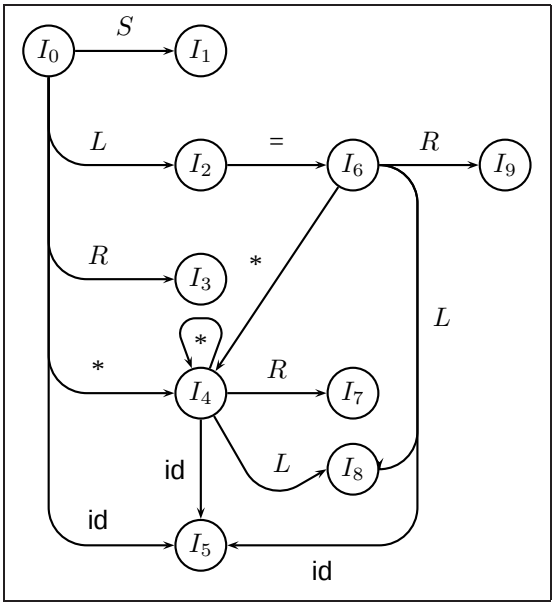
$$I_0: \begin{array}{lcl} S' & \rightarrow & .S \\ S & \rightarrow & .L = R \\ S & \rightarrow & .R \\ L & \rightarrow & .*R \\ L & \rightarrow & .\text{id} \\ R & \rightarrow & .L \end{array} \qquad I_1: S' \rightarrow S.$$

$$I_2: \begin{array}{lcl} S & \rightarrow & L.= R \\ R & \rightarrow & L. \end{array} \qquad I_3: S \rightarrow R.$$

$$I_4: \begin{array}{lcl} L & \rightarrow & *.R \\ R & \rightarrow & .L \\ L & \rightarrow & .*R \\ L & \rightarrow & .\text{id} \end{array} \qquad I_5: L \rightarrow \text{id}.$$

$$I_6: \begin{array}{lcl} S & \rightarrow & L = .R \\ R & \rightarrow & .L \\ L & \rightarrow & .*R \\ L & \rightarrow & .\text{id} \end{array} \qquad I_7: L \rightarrow *R.$$

$$I_8: R \rightarrow L. \qquad I_9: S \rightarrow L = R.$$



جدول تجزیه‌ی حاصل، دارای تداخل شیفت ـ کاهش است:

$$I_2 \quad :$$
$$\text{ACTION}[2, =] = \text{shift}(6)$$
$$\text{ACTION}[2, =] = \text{reduce}(R \rightarrow L), \quad \text{since } = \in Follow(R)$$

بنابراین، این گرامر SLR(1) نیست.

۹-۳ روش LR قانون‌مند (CLR)

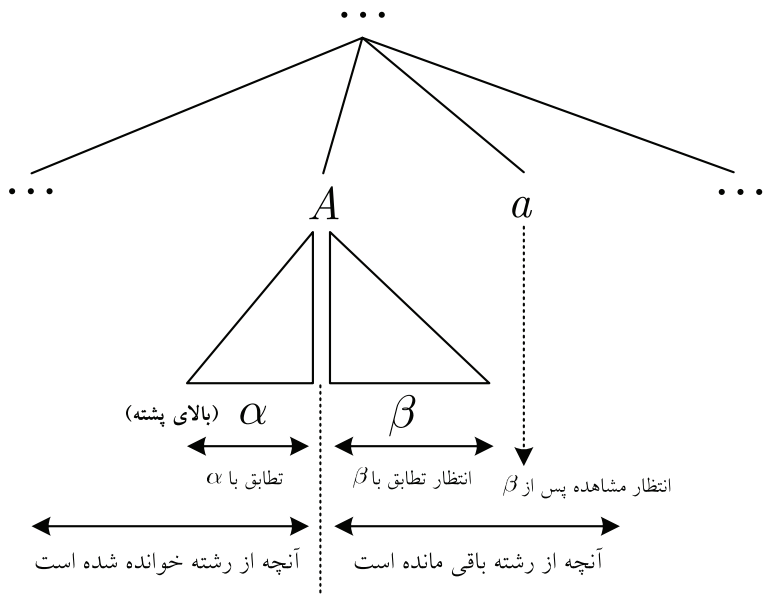
روش LR قانون‌مند (Canonical LR :CLR) قدرت‌مندترین روش تجزیه‌ی LR است. هر حالت یک تجزیه‌گر LR (CLR) نشان می‌دهد که در صورت وجود یک امکان برای کاهش، چه نمادی می‌تواند به دنبال دستگیره ظاهر شود.

۹-۳-۱ آیتم LR(1)

یک **آیتم LR(1)** از گرامر G، یک زوج مرتب به صورت زیر است که در آن L مجموعه‌ی پیش‌بینی (lookahead) نام دارد:

$$[A \to \alpha \bullet \beta, L], \qquad \eta = \alpha\beta,\ A \to \eta \in P(G),\ L \subseteq Follow(A)$$

- آیتم $[A \to \alpha \bullet \beta, \{a\}]$ وضعیت تجزیه‌گر را توصیف می‌کند:
  - سعی می‌کنیم یک A (ناپایانه) بیابیم که به دنبال آن یک a (پایانه) بیاید.
  - هم اکنون α را بر بالای پشته داریم.
  - بنابراین نیاز داریم که یک پیشوند مشتق شده از βa را مشاهده کنیم.
- نماد پیش‌بینی a تاثیری بر آیتم‌های به شکل $[A \to \alpha \bullet \beta, \{a\}]$ با $\beta \not\Rightarrow \epsilon$ ندارد.
- اگر آیتم به شکل $[A \to \alpha\bullet, \{a\}]$ را داشته باشیم، تجزیه‌گر $\text{reduce}(A \to \alpha)$ را اجرا می‌کند اگر توکن ورودی جاری a باشد.



شکل ۹-۳: تعبیر شماتیک از آیتم LR(1) به صورت $[A \to \alpha \bullet \beta, \{a\}]$: سعی می‌کنیم یک ناپایانه‌ی A بیابیم که به دنبال آن، پایانه‌ی a بیاید. هم‌اکنون α را بر بالای پشته داریم، پس باید یک پیشوند مشتق شده از βa را بیابیم.

۹-۳-۲ آتوماتون LR(1)

در روش‌های LR، دستگیره در هر مرحله به کمک یک آتوماتون (DFA) شناسایی می‌شود. از آیتم‌ها برای ساخت این DFA استفاده می‌گردد.

- آیتم‌هایی که وضعیت یکسانی را نمایش می‌دهند در یک گروه دسته‌بندی می‌شود.
- هر کدام از این مجموعه‌ها یک حالت از DFA را نشان می‌دهد.

عملیات بستار (Closure)

اگر I یک مجموعه از آیتم‌های LR(1) گرامر G باشد، $\text{CLOSURE}(I)$ نیز یک مجموعه‌ی از آیتم‌هاست که به صورت زیر محاسبه می‌شود:

۱) در ابتدا I به $\text{CLOSURE}(I)$ اضافه می‌شود.

۲) **اگر** $[A \to \alpha \bullet B\beta, \{a\}] \in \text{CLOSURE}(I)$ (نقطه قبل از یک ناپایانه‌ی B) و قاعده‌ی $B \to \gamma$ در G موجود بود،
آن‌گاه آیتم $[B \to \bullet\gamma, First(\beta a)]$ به $\text{CLOSURE}(I)$ اضافه می‌شود.

۳) قاعده‌ی فوق را آن‌قدر تکرار می‌کنیم تا دیگر چیزی به $\text{CLOSURE}(I)$ اضافه نشود.

> اگر $[A \to \alpha \bullet B\beta, \{a\}] \in \text{CLOSURE}(I)$، انتظار داریم که ادامه‌ی ورودی از $B\beta$ قابل اشتقاق باشد در حالی که توکن جاری a باشد.
> حال اگر $B \to \gamma$ در گرامر موجود باشد، بدیهی است که می‌توان انتظار داشت که ادامه‌ی ورودی از γ قابل اشتقاق باشد در حالی که توکن جاری در $First(\beta a)$ قرار داشته باشد.

مثال

گرامر زیر را در نظر می‌گیریم:

$$\begin{aligned} S' &\to S\$ \\ S &\to E + S \\ S &\to E \\ E &\to \mathbf{num} \end{aligned}$$

بستار یک آیتم نمونه به صورت زیر محاسبه می‌شود:

$$\begin{aligned} \text{CLOSURE}(\{[S' \to \bullet S\$, \{\$\}]\}) = \{&[S' \to \bullet S\$, \{\$\}], \\ &[S \to \bullet E + S, \{\$\}], \\ &[S \to \bullet E, \{\$\}], \\ &[E \to \bullet \mathbf{num}, \{+, \$\}]\} \end{aligned}$$

عملیات برو به (Goto)

اگر I مجموعه‌ای از آیتم‌ها و X یک نماد گرامر باشد،
در این صورت $\text{GOTO}(I, X)$ بستار مجموعه‌ی همه‌ی آیتم‌ها به صورت $[A \to \alpha X \bullet \beta, \{a\}]$ است که در آن $[A \to \alpha \bullet X\beta, \{a\}]$ در I قرار دارد.

$$[A \to \alpha \bullet X\beta, \{a\}] \in I \quad \Rightarrow \quad \text{CLOSURE}(\{[A \to \alpha X \bullet \beta, \{a\}]\}) \subseteq \text{GOTO}(I, X)$$

کلیه‌ی آیتم‌های موجود در مجموعه‌ی I که در آنها نقطه قبل از X قرارگرفته است، با انتقال نقطه به بعد از X به $I' = \text{GOTO}(I, X)$ اضافه می‌شوند و بستار مجموعه‌ی جدید I' محاسبه می‌شود.

مثال

گرامر زیر را در نظر می‌گیریم:

$$\begin{aligned} S' &\to S\$ \\ S &\to E + S \\ S &\to E \\ E &\to \textbf{num} \end{aligned}$$

یک نمونه از محاسبه‌ی تابع Goto به صورت زیر می‌باشد:

$$\begin{aligned} &\text{GOTO}(\{[S \to E \bullet + S, \{\$\}], [S \to E \bullet, \{\$\}]\}, +) = \\ &\text{CLOSURE}(\{[S \to E + \bullet S, \{\$\}]\}) = \{ \\ &[S \to E + \bullet S, \{\$\}], \\ &[S \to \bullet E + S, \{\$\}], \\ &[S \to \bullet E, \{\$\}], \\ &[E \to \bullet \textbf{num}, \{+, \$\}]\} \end{aligned}$$

ساخت DFAی LR(1)

هر حالت DFA حاوی یک مجموعه آیتم است و به صورت زیر ساخته می‌شود:

- حالت شروع $(\text{CLOSURE}(\{[S' \to \bullet S\$, \{\$\}]\})$ است.
- به ازای هر نماد X که در یکی از آیتم‌های مجموعه‌ی مربوط به حالت I به صورت

$$[A \to \alpha \bullet X\beta, \{a\}]$$

قرار دارد، بین I و $\text{GOTO}(I, X)$ یک گذر با برچسب X ایجاد می‌شود.

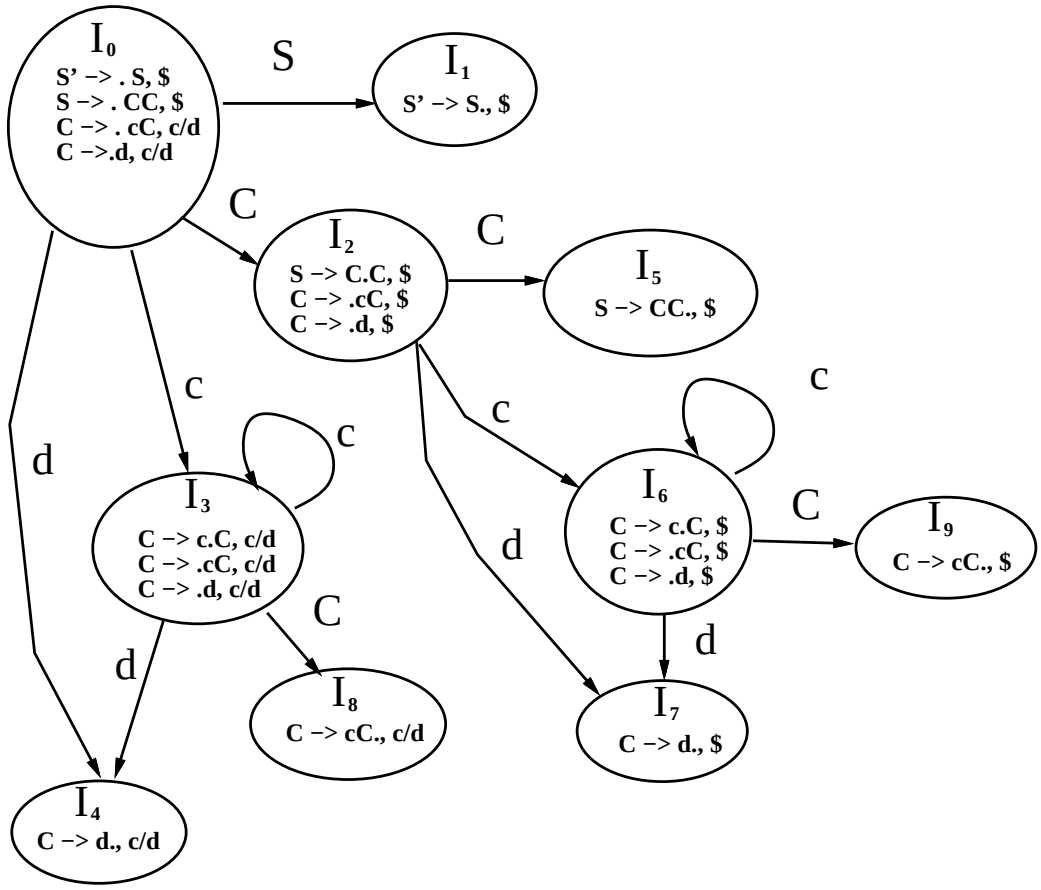
- حالت نهایی DFA، حالتی است که در آن آیتم $[S' \to S \bullet \$, \{\$\}]$ قرار دارد (نقطه به انتها رسیده است).

مثال

برای گرامر زیر:

$$
\begin{aligned}
S' &\rightarrow S\$ \\
S &\rightarrow CC \\
C &\rightarrow cC \mid d
\end{aligned}
$$

آتوماتون LR(1) به صورت زیر ساخته می‌شود:



۹-۳-۳ ساخت جدول تجزیه‌ی LR قانون‌مند (CLR)

جدول تجزیه‌ی LR، عملکرد DFA را نمایش می‌دهد:

- اگر $\text{GOTO}(I_i, a) = I_j$ در این صورت $\text{ACTION}[i, a] = \text{shift}(j)$.
- اگر $[A \rightarrow \alpha\bullet, \{a\}] \in I_i$، داریم $\text{ACTION}[i, a] = \text{reduce}(A \rightarrow \alpha)$.
- اگر $[S' \rightarrow S \bullet \$, \{\$\}] \in I_i$، داریم $\text{ACTION}[i, \$] = \text{accept}$.
- اگر $\text{GOTO}(I_i, A) = I_j$، داریم $\text{GOTO}[i, A] = j$.
- اگر خانه‌ای از جدول با قواعد فوق پر نشد، آن خانه حاوی error است.
- حالت آغازین، حالتی است که آیتم $[S' \rightarrow \bullet S\$, \{\$\}]$ را در خود دارد.

اگر جدول تجزیه برای گرامری حاوی تداخل باشد، آن گرامر LR(1) نیست.

جدول تجزیه‌ی ایجاد شده توسط الگوریتم فوق، جدول تجزیه‌ی LR(1) قانونمند (Canonoic) یا CLR(1) نام دارد.

مثال

برای گرامر زیر،

$$
\begin{array}{lcll}
S' & \rightarrow & S\$ & (0) \\
S & \rightarrow & CC & (1) \\
C & \rightarrow & cC \mid d & (2,3)
\end{array}
$$

جدول تجزیه CLR(1) متناظر با آتوماتون LR(1) مثال قبل، به صورت زیر می‌باشد:

	Action			Goto	
State	c	d	\$	S	C
0	s3	s4		1	2
1			acc		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

۴-۹ روش LR با نماد پیش‌بینی (LALR)

LALR (Lookahead-LR)

- در عمل به جاری جدول تجزیه‌ی CLR از جدول تجزیه‌ی LALR استفاده می‌شود. زیرا تعداد حالت‌های آن بسیار کمتر است.
- تعداد حالت‌های جدول تجزیه‌ی LALR مساوی با تعداد حالت‌های جدول تجزیه‌ی SLR است.
- برای زبانی مانند پاسکال، تعداد حالت‌های جدول تجزیه‌ی CLR هزاران حالت است در حالی که تعداد حالت‌های جدول تجزیه‌ی LALR صدها حالت است.

روش ایجاد: حالت‌هایی (مجموعه آیتم‌هایی) که تنها مجموعه‌ی Lookahead آنها متفاوت است با هم ادغام می‌شوند.

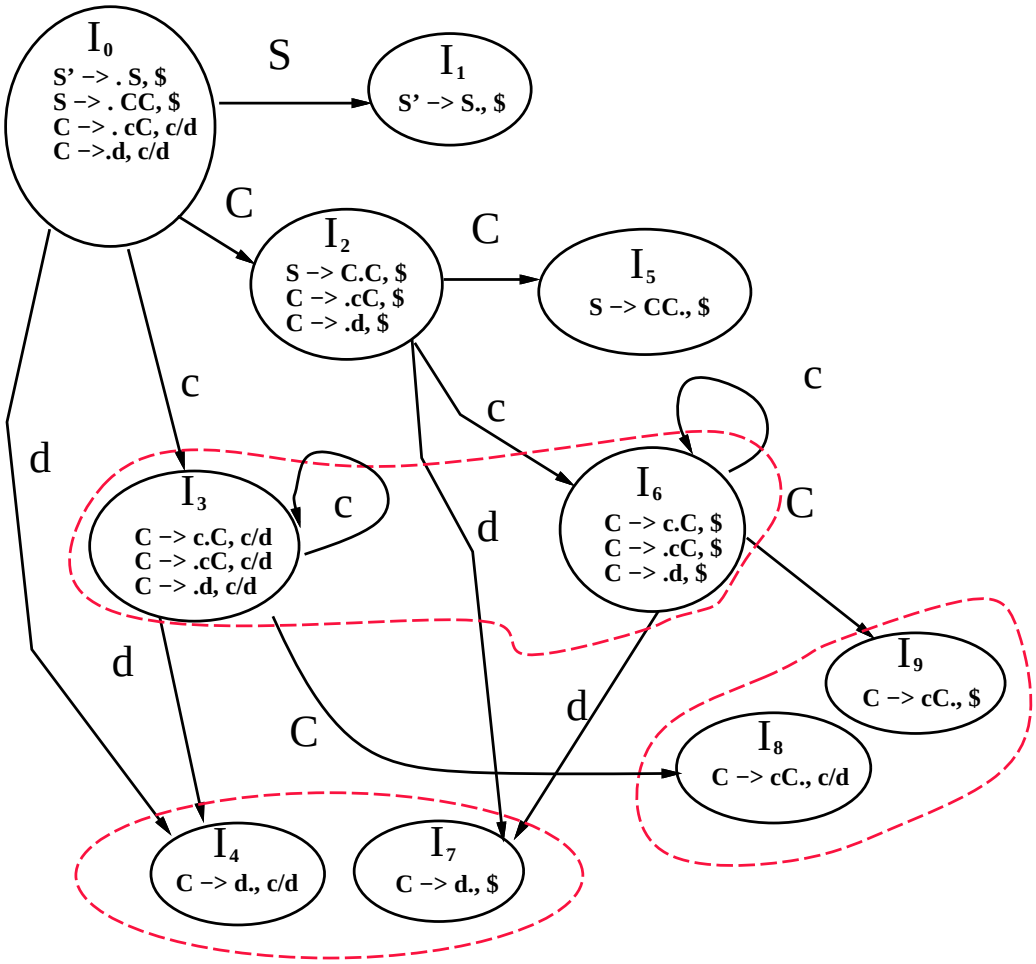
به عبارت دیگر، حالت‌هایی که دارای **core** یکسان هستند، با هم ادغام می‌شوند.

مثال

می‌خواهیم جدول تجزیه‌ی LALR را برای گرامر زیر به دست آوریم:

$$
\begin{array}{lcll}
S' & \rightarrow & S\$ & (0) \\
S & \rightarrow & CC & (1) \\
C & \rightarrow & cC \mid d & (2,3)
\end{array}
$$

از آتوماتون LR(1) حاصل در قسمت قبل استفاده می‌کنیم:



و حالت‌های دارای core یکسان را با هم ادغام می‌کنیم: آیتم‌های (I_3 و I_6)، (I_4 و I_7) و (I_8 و I_9) دارای core یکسان هستند.

$$
\begin{array}{llcll}
I_{36}: & C & \rightarrow & \mathsf{c}.C, & \mathsf{c/d/\$} \\
 & C & \rightarrow & .\mathsf{c}C, & \mathsf{c/d/\$} \\
 & C & \rightarrow & .\mathsf{d}, & \mathsf{c/d/\$}
\end{array}
\qquad
\begin{array}{llcll}
I_{47}: & C & \rightarrow & \mathsf{d}., & \mathsf{c/d/\$} \\
 & & & & \\
I_{89}: & C & \rightarrow & \mathsf{c}C., & \mathsf{c/d/\$}
\end{array}
$$

با استفاده از DFA حاصل، جدول تجزیه LALR(1) گرامر فوق، به صورت زیر به دست می‌آید:

	Action			Goto	
State	c	d	\$	S	C
0	s36	s47		1	2
1			acc		
2	s36	s47			5
36	s36	s47			89
47	r3	r3	r3		
5			r1		
89	r2	r2	r2		

۹-۴-۱ بروز تداخل کاهش ـ کاهش در اثر ادغام حالت‌ها

در صورت ادغام حالت‌هایی که دارای core یکسان هستند، تداخل شیفت ـ کاهش هرگز بروز نمی‌کند.
در صورت ادغام حالت‌هایی که دارای core یکسان هستند، ممکن است تداخل کاهش ـ کاهش بروزکند.
در صورت بروز تداخل، گرامر مربوطه LALR(1) نیست.

مثال

در یک آتوماتون LR(1)، آیتم‌های زیر را داریم:

$I_i = \{[A \to c\bullet, \{d\}], [B \to c\bullet, \{e\}]\}$
$I_i = \{[A \to c\bullet, \{e\}], [B \to c\bullet, \{d\}]\}$

آیتم‌های I_i و I_j دارای core یکسان هستند و هر یک به تنهایی ایجاد تداخل نمی‌کنند، اما آیتم I_{ij} که از ادغام دو آیتم فوق به دست می‌آید:

$$I_{ij} = \{[A \to c\bullet, \{d, e\}], [B \to c\bullet, \{d, e\}]\}$$

دارای تداخل کاهش ـ کاهش بر روی ورودی‌های e و d است.

۹-۴-۲ مقایسه‌ی جدول تجزیه‌ی LALR(1) با جدول تجزیه‌ی SLR(1)

تعداد حالت‌های جدول تجزیه‌ی LALR(1) با تعداد حالت‌های جدول تجزیه‌ی SLR(1) مساوی است.
تنها تفاوت این دو جدول در تعداد درایه‌های حاوی عمل کاهش است.

نکته: اگر یک جدول تجزیه‌ی SLR(1) تنها حاوی تداخل شیفت ـ کاهش باشد و به نحوی بدانیم که گرامر آن LALR(1) است می‌توان از میان تداخل‌ها، کاهش را حذف کرد و به این ترتیب به جدول تجزیه‌ی بدون تداخل رسید. در این جدول تعداد کاهش‌ها بزرگتر یا مساوی با تعداد کاهش‌های جدول LALR(1) است، یعنی در صورت وقوع خطا ممکن است تعداد کاهش بیشتری نسبت به تجزیه‌گر LALR(1) انجام شود.

۹-۴-۳ رفتار تجزیه‌گر LALR در مقابل تجزیه‌گر CLR: مقایسه

- در صورت صحیح بودن رشته‌ی ورودی، هر دو تجزیه‌گر دنباله‌ی یکسانی از شیفت ـ کاهش انجام می‌دهند. (صرف نظر از نام حالت‌ها)
- در صورت خطادار بودن رشته‌ی ورودی، تجزیه‌گر LALR تعداد بیشتری کاهش نسبت به تجزیه‌گر CLR انجام می‌دهد.

هر سه‌ی تجزیه‌گرهای LR در صورت رسیدن به توکن خطا دیگر shift اضافی انجام نمی‌دهند. روش CLR حتی reduce اضافی نیز انجام نمی‌دهد.

مثال

برای مقایسه‌ی رفتار تجزیه‌گر LALR در مقابل تجزیه‌گر CLR، گرامر زیر را در نظر می‌گیریم:

$$\begin{array}{lcll} S' & \rightarrow & S\$ & (0) \\ S & \rightarrow & CC & (1) \\ C & \rightarrow & cC \mid d & (2,3) \end{array}$$

رفتار تجزیه‌گر CLR بر روی ورودی نادرست ccd به صورت زیر می‌باشد:

Stack	Input	Action
0	$ccd\$$	shift(3)
0 c 3	$cd\$$	shift(3)
0 c 3 c 3	$d\$$	shift(4)
0 c 3 c d 4	$\$$	error

در صورتی که رفتار تجزیه‌گر LALR بر روی همان ورودی به صورت زیر است:

Stack	Input	Action
0	$ccd\$$	shift(36)
0 c 36	$cd\$$	shift(36)
0 c 36 c 36	$d\$$	shift(47)
0 c 36 c 36 d 47	$\$$	reduce($C \rightarrow d$)
0 c 36 c 36 C 89	$\$$	reduce($C \rightarrow cC$)
0 c 36 C 89	$\$$	reduce($C \rightarrow cC$)
0 C 2	$\$$	error

که مشخصاً تعداد مراحل تجزیه تا رسیدن به خطا در روش LALR بیشتر و شامل چند کاهش اضافی است.

۹-۵ تداخل‌ها

جدول زیر، شرایط وقوع تداخل در جدول تجزیه را برای روش‌های مختلف LR نشان می‌دهد:

	Shift-Reduce $[\ A ::= \alpha \bullet , \delta\ ]$ $[\ B ::= \beta \bullet \gamma, \eta\ ]$	Reduce-Reduce $[\ A ::= \alpha \bullet , \delta\ ]$ $[\ B ::= \beta \bullet, \eta\ ]$
LR(0)	conflict	conflict
SLR(1)	$\text{FOLLOW}(A) \cap \text{FIRST}(\gamma)$	$\text{FOLLOW}(A) \cap \text{FOLLOW}(B)$
LR(1)	$\delta \cap \text{FIRST}(\gamma)$	$\delta \cap \eta$

۹-۶ مطالب تکمیلی

۹-۶-۱ استفاده از گرامرهای مبهم

- گرامرهای مبهم توصیف کوتاه‌تر و طبیعی‌تری را از ساختارهای نحوی فراهم می‌کنند.
- این به آن معنی است که تجزیه‌گر در صورت استفاده از گرامر مبهم تعداد کمتری حالت خواهد داشت.
- اما گرامرهای مبهم، LR نیستند، زیرا:
  - جدول تجزیه شامل خانه‌های چندمقداری است.
  - بنابراین در فرایند تجزیه با تداخل‌های شیفت ـ کاهش و کاهش ـ کاهش مواجه می‌شویم.
- برای استفاده از گرامرهای مبهم، از قواعدی جهت **ابهام‌زدایی در جدول تجزیه** استفاده می‌کنیم: رفع تداخل‌های موجود در جدول تجزیه در صورت امکان

مثال

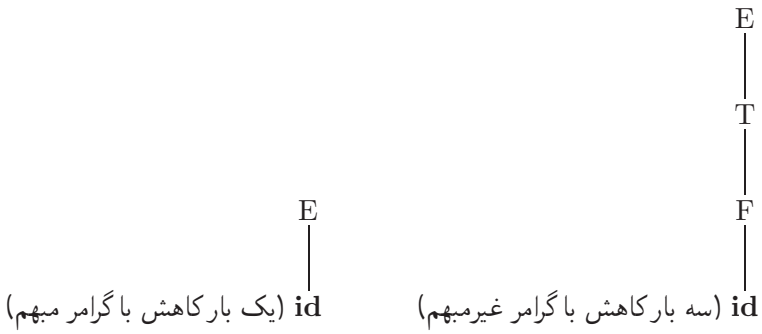
- گرامر زیر یک گرامر مبهم برای عبارت‌های ریاضی است که در آن تقدم و شرکت‌پذیری عملگرها لحاظ نشده است:

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id}$$

- گرامر غیر مبهم معادل با گرامر فوق با لحاظ کردن تقدم بیشتر $*$ نسبت به $+$ و شرکت‌پذیری عملگرها از چپ به صورت زیر است:

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T * F \mid F \\ F &\rightarrow (E) \mid \mathbf{id} \end{aligned}$$

این دو گرامر، رشته‌ی **id** را به دو صورت تولید می‌کنند:



جدول تجزیه برای گرامر مبهم عبارت‌های ریاضی به صورت زیر به دست می‌آید:

$$E \to E + E \mid E * E \mid (E) \mid \mathbf{id} \quad (1, 2, 3, 4)$$

	Action						Goto
State	**id**	+	*	(	)	$	E
0	s3			s2			1
1		s4	s5			acc	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			7
5	s3			s2			8
6		s4	s5		s9		
7		s4/r1	s5/r1		r1	r1	
8		s4/r2	s5/r2		r2	r2	
9		r3	r3		r3	r3	

رفع تداخل‌های جدول تجزیه بر اساس موارد زیر انجام می‌شود:

- **رفع تداخل بر اساس شرکت‌پذیری چپ عمل جمع**

Stack	Input	Action
0	**id** + **id** + **id**$	shift(3)
0 **id** 3	+ **id** + **id**$	reduce($F \to$ **id**)
0 E 1	+ **id** + **id**$	shift(4)
0 E 1 + 4	**id** + **id**$	shift(3)
0 E 1 + 4 **id** 3	+ **id**$	reduce($F \to$ **id**)
0 E 1 + 4 E 7	+ **id**$	shift(4)/reduce($E \to E + E$)

- از آنجا که + شرکت‌پذیر از چپ است، باید عمل reduce انتخاب شود.
- در این صورت ابتدا به محتوای پشته رسیدگی می‌شود و سپس ادامه‌ی ورودی به آن اضافه می‌گردد.

$$\text{Action}[7, +] = \text{reduce}(E \to E + E)$$

- **رفع تداخل بر اساس تقدم بیشتر عمل ضرب نسبت به عمل جمع**

Stack	Input	Action
0	$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	$\text{shift}(3)$
$0\ \mathbf{id}\ 3$	$+\mathbf{id} * \mathbf{id}\$$	$\text{reduce}(F \rightarrow \mathbf{id})$
$0\ E\ 1$	$+\mathbf{id} * \mathbf{id}\$$	$\text{shift}(4)$
$0\ E\ 1\ +\ 4$	$\mathbf{id} * \mathbf{id}\$$	$\text{shift}(3)$
$0\ E\ 1\ +\ 4\ \mathbf{id}\ 3$	$*\mathbf{id}\$$	$\text{reduce}(F \rightarrow \mathbf{id})$
$0\ E\ 1\ +\ 4\ E\ 7$	$*\mathbf{id}\$$	$\text{shift}(5)/\text{reduce}(E \rightarrow E + E)$

- از آنجا که تقدم $*$ بیشتر از $+$ است باید shift انجام شود.
- در این صورت ابتدا ادامه‌ی ورودی به پشته اضافه می‌شود و پس از تکمیل، کاهش صورت می‌گیرد.

$$\text{Action}[7, *] = \text{shift}(5)$$

- **رفع تداخل بر اساس شرکت‌پذیری عمل ضرب**

Stack	Input	Action
0	$\mathbf{id} * \mathbf{id} * \mathbf{id}\$$	$\text{shift}(3)$
$0\ \mathbf{id}\ 3$	$*\mathbf{id} * \mathbf{id}\$$	$\text{reduce}(F \rightarrow \mathbf{id})$
$0\ E\ 1$	$*\mathbf{id} * \mathbf{id}\$$	$\text{shift}(5)$
$0\ E\ 1\ *\ 4$	$\mathbf{id} * \mathbf{id}\$$	$\text{shift}(3)$
$0\ E\ 1\ *\ 4\ \mathbf{id}\ 3$	$*\mathbf{id}\$$	$\text{reduce}(F \rightarrow \mathbf{id})$
$0\ E\ 1\ *\ 4\ E\ 8$	$*\mathbf{id}\$$	$\text{shift}(5)/\text{reduce}(E \rightarrow E * E)$

- از آنجا که $*$ شرکت‌پذیر از چپ است، باید عمل reduce انتخاب شود.
- در این صورت ابتدا به محتوای پشته رسیدگی می‌شود و سپس ادامه‌ی ورودی به آن اضافه می‌گردد.

$$\text{Action}[8, *] = \text{reduce}(E \rightarrow E * E)$$

- **رفع تداخل بر اساس تقدم کمتر عمل جمع نسبت به عمل ضرب**

Stack	Input	Action
0	$\mathbf{id} * \mathbf{id} + \mathbf{id}\$$	$\text{shift}(3)$
$0\ \mathbf{id}\ 3$	$*\mathbf{id} + \mathbf{id}\$$	$\text{reduce}(F \rightarrow \mathbf{id})$
$0\ E\ 1$	$*\mathbf{id} + \mathbf{id}\$$	$\text{shift}(4)$
$0\ E\ 1\ *\ 4$	$\mathbf{id} + \mathbf{id}\$$	$\text{shift}(3)$
$0\ E\ 1\ *\ 4\ \mathbf{id}\ 3$	$+\mathbf{id}\$$	$\text{reduce}(F \rightarrow \mathbf{id})$
$0\ E\ 1\ *\ 4\ E\ 8$	$+\mathbf{id}\$$	$\text{shift}(4)/\text{reduce}(E \rightarrow E * E)$

- از آنجا که تقدم $+$ کمتر از $*$ است باید reduce انجام شود.

– در این صورت ابتدا به محتوای پشته رسیدگی می‌شود و سپس ادامه‌ی ورودی به آن اضافه می‌گردد.

$$\textsc{Action}[8, +] = \text{reduce}(E \to E * E)$$

نهایتاً جدول تجزیه بدون تداخل برای گرامر مبهم عبارت‌های ریاضی به صورت زیر به دست می‌آید:

$$E \to E + E \mid E * E \mid (E) \mid \mathbf{id} \quad (1, 2, 3, 4)$$

	Action						Goto
State	**id**	+	*	(	)	$	E
0	s3			s2			1
1		s4	s5			acc	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			7
5	s3			s2			8
6		s4	s5		s9		
7		r1	s5		r1	r1	
8		r2	r2		r2	r2	
9		r3	r3		r3	r3	

مثال

گرامر مبهم دستور شرطی که دارای مساله‌ی else معلق است، به صورت زیر می‌باشد:

$$S \to iSeS \mid iS \mid a$$

آیتم I_4 برای این گرامر به صورت زیر است:

$$I_4 = \{S \to iS \bullet eS, S \to iS\bullet\}$$

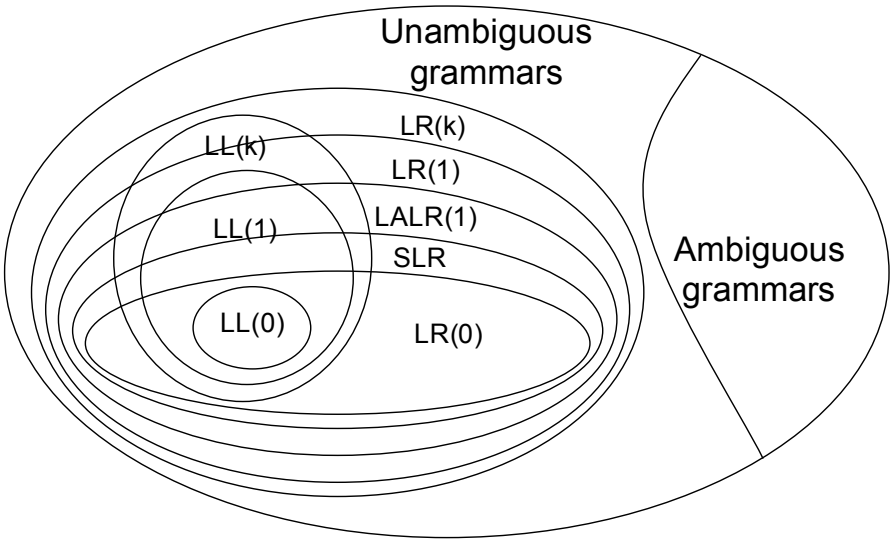
از آنجا که $Follow(S) = \{e, \$\}$ تداخل خواهیم داشت در خانه‌ی $\textsc{Action}[4, e]$:

$$\textsc{Action}[4, e] = \{\text{shift}(e), \text{reduce}(S \to iS)\}$$

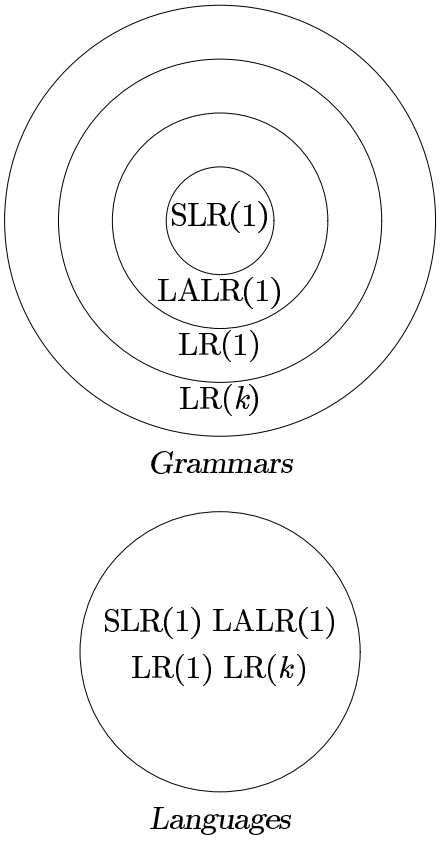
برای تطابق if با نزدیک‌ترین else باید $\textsc{Action}[4, e] = \text{shift}(e)$ انتخاب شود:

Stack	Input	Action
$0\ i\ 2\ i\ 2\ S\ 4$	$ea\$$	$\text{shift}(e)$

۹-۶-۲ نسبت خانواده گرامرهای بررسی شده



نسبت خانواده گرامرها و زبان‌های LR با یکدیگر



گرامر LR(0): گرامری $\mathrm{LR}(0)$ است که با پویش چپ به راست ورودی و با استفاده از عکس اشتقاق راست، در هر مرحله بدون نگاه کردن به ورودی و تنها بر اساس محتوای پشته بتوان قاعده‌ی مورد استفاده در آن گام را شناسایی کرد.

◄ تمرین

۱. گرامر زیر را در نظر بگیرید:

$$\begin{array}{lcll} E & \rightarrow & E+T \mid T & (1,2) \\ T & \rightarrow & TF \mid F & (3,4) \\ F & \rightarrow & F* \mid a \mid b & (5,6,7) \end{array}$$

آ) جدول تجزیه‌ی SLR را برای این گرامر بسازید.

ب) جدول تجزیه‌ی CLR را برای این گرامر بسازید.

پ) جدول تجزیه‌ی LALR را برای این گرامر بسازید.

۲. نشان دهید که گرامر زیر LL(1) است ولی SLR(1) نمی‌باشد:

$$\begin{array}{lcl} S & \rightarrow & AaAb \mid BbBa \\ A & \rightarrow & \epsilon \\ B & \rightarrow & \epsilon \end{array}$$

۳. نشان دهید که یک گرامر LR(1) نمی‌تواند مبهم باشد.

۴. نشان دهید که گرامر زیر LALR(1) است، اما SLR(1) نیست:

$$\begin{array}{lcl} S & \rightarrow & Aa \mid bAc \mid dc \mid bda \\ A & \rightarrow & d \end{array}$$

۵. نشان دهید که گرامر زیر LR(1) است، اما LALR(1) نیست:

$$\begin{array}{lcl} S & \rightarrow & Aa \mid bAc \mid Bc \mid bBa \\ A & \rightarrow & d \\ B & \rightarrow & d \end{array}$$

۶. یک جدول تجزیه برای گرامر مبهم زیر بسازید که عبارت‌های منظم را بر روی الفبای $\{a, b\}$ تولید می‌کند. تداخل‌های جدول تجزیه را به گونه‌ای رفع کنید که رفتار تجزیه‌گر به صورت مورد انتظار شود:

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

۷. یک جدول تجزیه برای گرامر زیر (دارای **else** معلق) بسازید که در آن **expr** یک پایانه است. تداخل‌های جدول تجزیه را به گونه‌ای رفع کنید که رفتار تجزیه‌گر به صورت مورد انتظار شود:

$$\begin{array}{lcl} stmt & \rightarrow & \textbf{if expr then } stmt \\ & \mid & \textbf{if expr then } stmt \textbf{ else } stmt \\ & \mid & \textbf{other} \end{array}$$