مثال. آیا گرامر زیر LL(1) هست؟

1. $S \rightarrow iEtS\ S'$
2. $S \rightarrow a$
3. $S' \rightarrow eS$
4. $S' \rightarrow \varepsilon$
5. $E \rightarrow b$

حل. ابتد First() و Follow() ها را پیدا می کنیم.

$First(S) = \{i, a\}$
$First(S') = \{e, \varepsilon\}$
$First(E) = \{b\}$
$Follow(S) = \{e, \$\}$
$Follow(S') = \{e, \$\}$
$Follow(E) = \{t\}$

جدول تجزیه

	a	b	e	i	t	$
S	2			1		
S'			3,4			4
E		5				

با توجه به جدول تجزیه گرامر فوق LL(1) نیست.
اگر گرامری در ورودی یا در مدخلی از جدول تجزیه بیش از یک مقدار داشته باشد آن گرامر LL(1) نیست .

بعضی گرامر ها را با
- رفع ابهام
- حذف بازگشتی چپ
- فاکتورگیری از چپ

می توان به LL(1) تبدیل کرد

تشخیص LL(1) بودن بدون استفاده از جدول

گرامری LL(1) است که برای هر قاعده آن که به فرم $A \rightarrow \alpha \mid \beta$ باشد شرایط زیر برقرار باشد.

۱- $First(\beta) \cap First(\alpha) = \phi$
۲- حداکثر یکی از α و β رشته ε را تولید کند.
۳- اگر $\alpha \xrightarrow{*} \varepsilon$ (به عبارتی $\varepsilon \in First(\alpha)$) باشد $First(\beta) \cap Follow(A) = \phi$
نکته: اگر $A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$ آنگاه بایستی $First(\alpha_1) \cap First(\alpha_2) \cap First(\alpha_3) \cap \ldots \cap First(\alpha_n) = \phi$

مثال. آیا گرامر زیر LL(1) هست.

1,2. $S \rightarrow A \mid B$
3,4. $A \rightarrow Ab \mid f$
5. $B \rightarrow cdA \mid ceA$

خیر زیرا $First(cdA) \cap First(ceA) = c \neq \phi$

از رشته ورودی شروع کرده و با جایگزینی قواعد سعی در تولید Start Symbol را دارند.

مثال تجزیه پایین به بالا– انتقال کاهش

دنباله ورودی abbcde

اشتقاق راست $S \rightarrow aABe \rightarrow aAde \rightarrow aAbcde \rightarrow abbcde$

S	→ aABe
A	→ Abc \ b
B	→ d

abbcde	انتقال
aAbcd	کاهش
aAde	انتقال
aABe	کاهش
S	

پارسرهای پایین بالا به جهت عکس اشتقاق راست عمل می کنند.

عبارت(Phars) : بخشی از یک فرم جمله ای است که از یک غیر پایانه بوجود آمده باشد به عنوان نمونه در مثال زیر، β یک عبارت محسوب می شود. $S \Rightarrow^* \alpha A \gamma \Rightarrow^+ \alpha\beta\gamma$

عبارت ساده(Simple Phars) : عبارتی است که در یک قدم بوجود آمده باشد در بسط زیر β یک عبارت ساده است

$S \Rightarrow^* \alpha A \gamma \Rightarrow \alpha\beta\gamma$

دستگیره(Handle): عبارت ساده ای است که در جهت عکس یک اشتقاق راست در نظر گرفته می شود سمت راست Handle هیچ غیر ترمینالی وجود ندارد.

تعریف دیگر: زیر رشته ای منطبق بر سمت راست یک قانون می باشد که ایجاد کننده یک کاهش به غیر پایانه سمت چپ آن قانون می باشد.

تعریف دقیق تر: اگر ترتیب کاهش رشته ورودی به نماد شروع معکوس سمت راست ترین اشتقاق(اشتقاق راست) باشد دنباله ای را که در هر مرحله کاهش می یابد، **دستگیره** می نامیم.

مثال. دستگیره ها را در کاهش bccdef در با توجه به گرامر زیر مشخص کنید.

$S \rightarrow bBCf$
$B \rightarrow Bcd \mid c$
$C \rightarrow e$

حل. ابتدا bccdef را بوسیله سمت راست ترین اشتقاق تولید می کنیم

S
bBCf
bBef
bBcdef
bccdef

دستگیره	کاهش
c	bccdef
Bcd	bBcdef
e	bBef
bBCf	bBCf
	S

پیاده سازی پارسرهای پایین به بالا با استفاده از یک پشته:

در این روش از یک پشته و یک بافر ورودی جهت نگهداری رشته ورودی استفاده می شود. در شروع عمل پارسینگ روی پشته و انتهای رشته ورودی $ را قرا می دهیم.

اعمال مورد استفاده در این روش:

انتقال(shift) : در انتقال، سمبول ها(Token های) ورودی بالای پشته قرار داده می شوند تا زمانی که یک handel در بالای پشته ظاهر شود. به عبارتی عمل انتقال تا وقتی که علایم روی پشته با سمت راست یکی از قوانین گرامر منطبق شود ادامه می یابد.

کاهش(Reduce) : مجموعه عناصر روی پشته که منطبق با سمت راست یکی از قوانین است را حذف و به جای آن یک غیر ترمینال که در سمت چپ قانون مذکور است را قرار می دهد.

قبول ورودی(Accept) : پارسر پایان موفقیت امیز تجزیه را اعلام می کند.

تشخیص خطا(Error) : پارسر یک خطای نحوی تشخیص داده و رویه خطا پرداز را فرا می خواند.

در زیر مراحل پارسینگ رشته id+id*id بر اساس گرامر زیر دیده می شود.

(1) E → E + E
(2) E → E * E
(3) E → (E)
(4) E → id

پشته	ورودی	عمل
$	id+id*id$	انتقال id
$id	+id*id$	کاهش $E \to id$
$E	+id*id$	انتقال +
$E+	id*id$	انتقال id
$E+id	*id$	کاهش $E \to id$
$E+E	*id$	انتقال *
$E+E*	id$	انتقال id
$E+E* id	$	کاهش $E \to id$
$E+E*E	$	کاهش $E \to E*E$
$E+E	$	کاهش $E \to E+E$
$E	$	Accept

در روش پارسینگ انتقال-کاهش دو نوع مشکل وجود دارد.
۱- تصمیم گیری در این مورد که کدام مجموعه از عناصر روی پشته با سمت راست قوانین منطبق است(مشکل در انتخاب دستگیره)
۲- انتخاب قانونی که کاهش تحت آن انجام شود. این مشکل وقتی پیش می اید، که سمت راست بیش از یک قانون با مجموعه ای از عناصر روی پشته منطبق باشد. چنین حالتی را تداخل کاهش-کاهش گویند

انواع تداخل در پارسینگ انتقال- کاهش

۱- **تداخل انتقال-کاهش(Shift-Reduce conflict):** زمانی روی می دهد که پارسر نتواند تصمیم بگیرد که عمل انتقال را انجام بدهد یا عمل کاهش را.
۲- **تداخل کاهش-کاهش(Reduce-Reduce conflict):** اگر بیش از یک قاعده برای کاهش در یک لحظه قابل استفاده باشد این نوع مشکل رخ داده است.

روش پارسینگ LR

روش پارسینگ LR جزء دسته الگوریتم های پارسینگ پایین به بالا می باشد، الگوریتم های LR خود به سه دسته کوچکتر تقسیم می شوند. که تفاوتشان در ساخت جدول تجزیه شان می باشد. در حالی که روش پارسینگ برای آنها مشابه است

- **(Simple LR)SLR**
- **(Look Ahead LR)LALR**
- **(Cannonical)CLR**

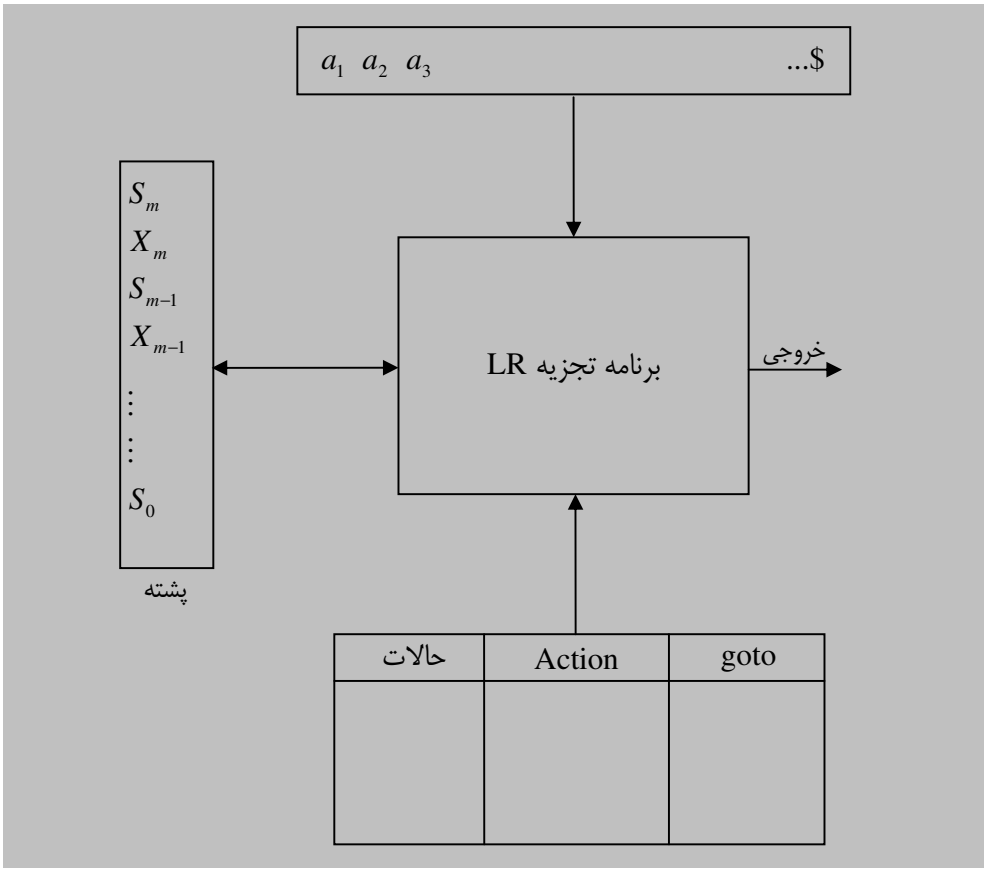
برای اکثریت قریب به اتفاق زبان های برنامه نویسی می توان از این روش پارسینگ استفاده نمود پارسر های LR دارای 4 جزء اصلی هستند

- **بافر ورودی**، رشته ورودی در این قسمت قرار می گیرد و به انتهای رشته ورودی $ اضافه می شود LR با دیدن $ پایان رشته ورودی را تشخیص می دهد
- **پشته:** محتوای پشته به صورت $S_0X_1S_1\cdots X_mS_m$ نشان داده می شود. S_i نشان دهنده حالات و X_i پایانه ها و غیر پایانه های گرامر است، نمادهای ورودی به پشته منتقل می شوند تا یک دستگیره در پشته یافت شود، پس از کشف دستگیره کاهش انجام میگیرد
- **برنامه پارسینگ:** این برنامه قسمت اصلی تجزیه کننده است برنامه تجزیه کننده بر اساس نماد جاری رشته ورودی و حالات بالای پشته و محتوای جدول تجزیه مرحله بعدی را تعیین می کند
- **جدول پارسینگ**: این بخش شامل دو قسمت action و goto است. قسمت action عملی را که باید انجام شود و بخش goto حالت بعدی را مشخص می کند

مزیت: سرعت اعلان خطا های نحوی از این پارسرها بالاست

عیب: جهت ساخت جدول تجزیه این پارسر ها حجم محاسبات دستی خیلی زیاد است.

در شکل زیر اجراء مختلف پارسر LR و رابطه انها با هم نشان داده شده است



یک قلم(1-item) برای LR(0) (قلم به طور خلاصه) از گرامر G مولدی از G با یک نقطه(یا علامت خاص دیگر) در مکانی در سمت راست آن است بنابراین مولد $A \to XYZ$ چهار قلم را ایجاد می کند.

A→ . XYZ
A→ X.YZ
A→ XY. Z
A→ XYZ.

مولد $A \to \varepsilon$ تنها یک قلم به صورت $A \to .$ را تولید می نماید، یک قلم می تواند با یک زوج عدد صحیح نشان داده شود، اولین عدد شماره مولد و دومین عدد موقعیت نقطه را مشخص می نماید، در نتیجه قلم مشخص می نماید که چه مقداری از مولد، در یک نقطه از فرایند تجزیه دیده شده است برای مثال اولین قلم فوق نشان میدهد که انتظار دیدن رشته مشتق پذیر از XYZ را در ورودی داریم. دومین قلم نشان می دهد که اخیرا در ورودی رشته مشتق پذیر از X دیده شده است و این که در ادامه انتظار دیدن یک رشته مشتق پذیر از YZ را داریم.

عمل Closure :

اگر I مجموعه ای از اقلام برای گرامر G باشد، آنگاه Closure(I) مجموعه اقلام ایجاد شده از I با استفاده از دو قانون زیر است

۱- در ابتدا هر قلم موجود در I به Closure(I) اضافه می شود.
- اگر $A \to \alpha \,.\, B\beta$ در Closure(I) موجود باشد و $B \to \gamma$ یک مولد باشد انگاه قلم B→. γ اگر قبلا در I نباشد به آن اضافه می گردد. این قانون تا زمانی به کار گرفته می شود که هیچ قلم جدید دیگری نتواند به Closure(I) اضافه شود.

مثال. باتوجه به گرامر زیر اگر I مجموعه ای یک قلم به صورت $\{[E' \to .E]\}$ باشد آنگاه Closure(I) شامل چه اقلام هائی خواهد شد؟

$$E' \to E$$
$$E \to E+T \mid T$$
$$T \to T*F \mid F$$
$$F \to (E) \mid id$$

پاسخ.

$$E' \to .E$$
$$E \to .E+T$$
$$E \to .T$$
$$T \to .T*F$$
$$T \to .F$$
$$F \to .(E)$$
$$F \to .id$$

ساخت جدول تجزیه پارسر SLR

ابتدا می بایست دیاگرام حالت این پارسر رسم شود جهت رسم دیاگرام حالت یک پارسر به صورت زیر عمل می کنیم

- اگر S علامت شروع گرامر باشد ابتدا قاعده ای به فرم $S' \to S$ ($S' \to S\$$)که در آن S' یک غیر پایانه جدید است به گرامر اضافه می کنیم، گرامر حاصل را گرامر افزوده (augmented grammar) گویند
- حالت جدیدی به نام S_0 ایجاد می کنیم و ایتم $S' \to .S_0$ در S_0 قرار داده و سپس Closure این ایتم را به S_0 اضافه می کنیم

- اگر به طور کلی در حالت S_i ایتم هائی به فرم $\begin{cases}\{A_1 \to \alpha_1.x\beta_1 \\ \vdots \\ \{A_n \to \alpha_n.x\beta_n\end{cases}$ داشته باشیم(که در آن x می تواند پایانه و یا غیر پایانه باشد) در

در این صورت حالت جدیدی به نام S_j ایجاد کرده، S_i را توسط لبه ای با بر چسب x به S_j منتقل می کنیم و ایتم فـوق را بـا ایـن تغییر که در همه علامت . به بعد از علامت x منتقل شده است در وضیعت جدید قرار می دهیم و سپس بستار این ایتم ها را محاسبه و در S_j قرار می دهیم. چنانکه در دیاگرام حالتی مانند S_k وجود داشته باشد که دقیقا مطابق S_j باشد در این صورت S_j ایجاد نشده و در عوض S_i توسط لبه ای با برچسب x به S_k متصل می گردد، این قدم را آنقدر تکرار می کنیم تا دیگر حالت جدیدی به دیـاگرام اضافه نشود.

مثال. دیاگرام حالت گرامر زیر را رسم کنید.

$$E \to E+T$$
$$E \to T$$
$$T \to id$$
$$T \to (E)$$

در قدم اول یک Start Symbol مانند $S \to E\$$ را به گرامر اضافه می کنیم و حالت جدید S_0 را ایجاد کرده و مراحل گفته شده را پـی می گیریم

$$S \to E$$
$$E \to E+T$$
$$E \to T$$
$$T \to id$$
$$T \to (E)$$