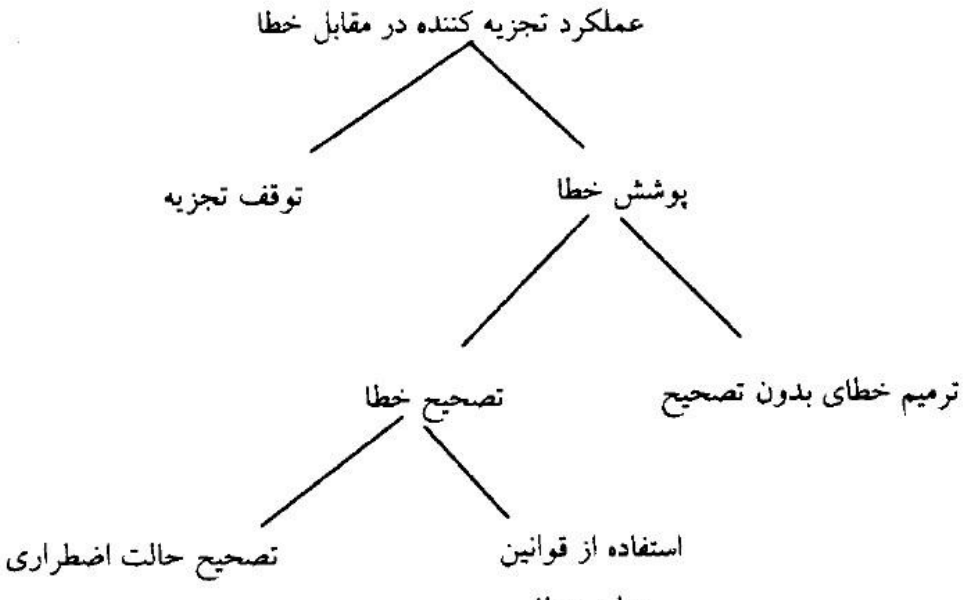
مدیریت خطا در تجزیه کننده



گرامر ذیل نشان می‌دهد که بعد از هر دستور سمی کالن قرار دارد.

stmt→stmt ; stmt_list | stmt ;

با توجه به آمار، یکی از خطاهایی که معمولا در برنامه رخ می‌دهد عدم درج سمی کالن است در نتیجه قاعده تولیدی به صورت ذیل نیز به گرامر اضافه می‌گردد.

stmt_error→stmt stmt_list | stmt

در نتیجه اگر برنامه نویس سمی کالن را درج نکند دستورات به جای stmt با stmt_error کاهش می‌یابد بنابراین تجزیه کننده متوقف نمی‌گردد بلکه پیغام مناسب را صادر و تجزیه ادامه می‌یابد.

پوشش خطا در تجزیه کننده پیشگو غیر بازگشتی

۱- اگر پایانه‌ای مانند a بالای پشته باشد که با نماد جاری یکسان نباشد، خطا رخ می‌دهد، به منظور پوشش خطا، نماد a را به ورودی اضافه می‌کنیم، بنابراین نماد بالای پشته با نماد جاری یکسان می‌شود بنابراین انتقال تطبیق انجام می‌گردد و نماد بالای پشته و نماد ورودی اضافه شده حذف می‌گردد و امید می‌رود تجزیه با موفقیت ادامه یابد، به عبارت ساده نماد بالای پشته حذف می‌گردد. در این روش پیغامی مبنی بر درج a به برنامه‌نویس داده خواهد شد، تا برنامه‌نویس از خطا مطلع شود.

$S \rightarrow bcA$

$A \rightarrow aA|c$

جدول تجزیه پیشگوی غیر بازگشتی

	a	b	c	$
S		S→bcA		
A	A→aA		A→c	

رشته bdcc را در مراحل ذیل تجزیه می‌کنیم.

مراحل تجزیه رشته bdcc

پشته	ورودی	توضیحات
$S	bdcc$	
$Acb	bdcc$	
$Ac	dcc$	نماد بالا پشته c و نماد ورودی d است، در نتیجه خطا رخ می‌دهد. بنابراین به منظور پوشش خطاc را به ورودی اضافه می‌کنیم.
$Ac	cdcc$	نماد بالای پشته و نماد ورودی یکسان است و در نتیجه نماد بالای پشته و نماد ورودی حذف می‌شوند و تجزیه ادامه می‌یابد.

۲- نمادهای first(A) را به عنوان مجموعه هماهنگ کننده A در نظر می‌گیریم . بنابراین اگر یکی از نمادهای first(A) در ورودی ظاهر شود تجزیه بر اساس A ادامه می‌یابد.

جدول تجزیه این گرامر در جدول ذیل نشان داده شده است.

$S \rightarrow bA$

$A \rightarrow aA|c$

جدول تجزیه پیشگوی غیر بازگشتی

	a	b	c	$
S		S→bA		
A	A→aA		A→c	

مراحل تجزیه رشته bdec

پشته	ورودی	توضیحات
$S	bdec$	
$Ab	bdec$	
$A	dec$	M[A,d] خالی است در نتیجه خطا رخ داده است. بنابراین از نمادهای ورودی تا رسیدن به نمادی از first(A) صرفنظر می‌شود. در نتیجه از d صرفنظر می‌شود.
$A	ec$	M[A,e] خالی است در نتیجه خطا رخ داده است. بنابراین از نمادهای ورودی تا رسیدن به نمادی از first(A) صرفنظر می‌شود. بنابراین از e صرفنظر می‌شود.
$A	c$	
$c	c$	
$	$	

۳- نمادهای follow(A) را به عنوان نمادهای هماهنگ کننده A در نظر می‌گیریم. follow(A) نمادهایی هستند که بعد از A قرار می‌گیرند، بنابراین از نمادهای ورودی تا رویت یکی از نمادهای follow(A) صرفنظر می‌کنیم و پس از رسیدن به یکی از نمادهای follow(A) ،غیر پایانه A را از بالای پشته حذف می‌کنیم.

$S \rightarrow bAe$
$A \rightarrow aA|c$

جدول تجزیه پیشگوی غیر بازگشتی

	a	b	c	e	$
S		S→bAe			
A	A→aA		A→c		

مراحل تجزیه رشته bde

پشته	ورودی	توضیحات
$S	bde$	
$eAb	bde$	
$eA	de$	M[A,d] خالی است در نتیجه خطا رخ داده است. بنابراین از نمادهای ورودی تا رسیدن به نمادی از follow(A) صرفنظر می‌شود. بنابراین از d صرفنظر می‌شود.
$eA	e$	M[A,e] خالی است در نتیجه خطا رخ داده است. بنابراین از نمادهای ورودی تا رسیدن به نمادی از follow(A) صرفنظر می‌شود. چون e در follow(A) است در نتیجه A از پشته حذف می‌شود.
$e	e$	
$	$	

این روش معایبی نیز دارد به عنوان مثال به قطعه برنامه ذیل دقت کنید:

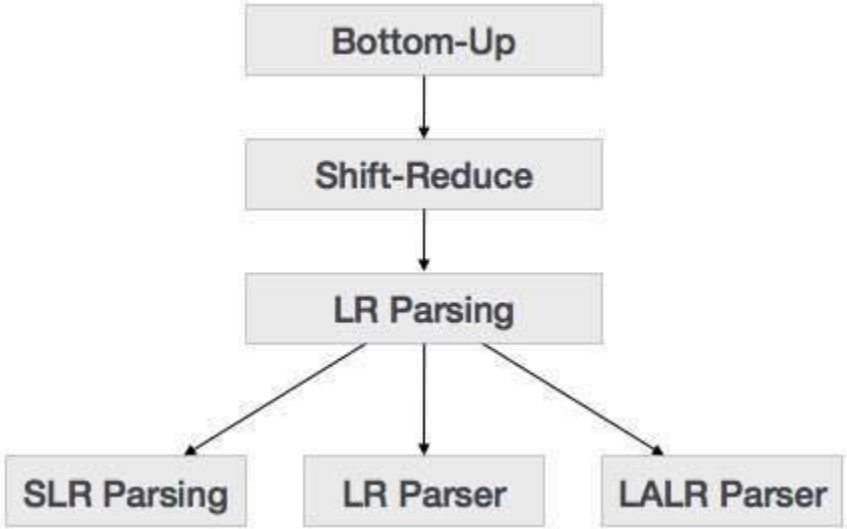
```
a:=b+c
if(a=b)
   d=d+;
```

دستور اول علامت ; از قلم افتاده است بنابراین یک خطای نحوی رخ داده است. در نتیجه تجزیه کننده تا رسیدن به علامت ; که علامت هماهنگ کننده است ورودی را حذف می‌کند ، بنابراین کل عبارت d=d+1 (a=b) if حذف می‌گردد.

٤- برای رفع مشکل روش ۳ کلمات کلیدی شروع کننده دستورات را به مجموعه هماهنگ کننده اضافه می‌کنیم. به عنوان مثال کلمات repeat،while،if را به مجموعه هماهنگ کننده غیر پایانه تولید کننده عبارات اضافه می‌کنیم.

تجزیه پایین به بالا

فرایند تجزیه پایین به بالا از گره‌های برگ یک درخت آغاز می‌شود و به سمت بالا تا رسیدن به گره ریشه ادامه می‌یابد. در این روش ما از یک جمله آغاز می‌کنیم و سپس قواعد ترکیب را به روش معکوس برای رسیدن به نماد آغازین ادامه می‌دهیم.



expr→ expr + term | expr - term | term
term→ 1|2|3|4

رشته 4+1-2 توسط تجزیه کننده پایین به بالا به صورت ذیل تجزیه می‌گردد.

4+1-2
term +1-2
expr + 1 -2
expr + term -2
expr - 2
expr - term
expr

رشته 4+1-2 با موفقیت به نماد شروع کاهش می‌یابد، در نتیجه این رشته پذیرفته می‌شود. اما رشته 1+2-+3 با توجه به مراحل ذیل قابل کاهش به نماد شروع نیست.

1+2-+3
term+ 2-+3
expr + 2 -+ 3
expr + term -+3
expr -+ 3
expr -+ term
expr -+ expr

تجزیه کننده قادر به کاهش رشته 1+2-+3 به نماد شروع نیست، در نتیجه رشته 4+2-+3 رد می‌شود. رشته 4+1-2 را به روشهای مختلفی می‌توان به نماد شروع کاهش داد.

انواع روشهای کاهش رشته 4+1-2 به نماد شروع

روش اول	روش دوم
4+1-2 4+ term -2 4+ term -term term + term -term expr + term -term expr - term expr	4+1-2 term +1-2 expr + 1 -2 expr + term -2 expr - 2 expr - term expr
روش سوم	روش چهارم
4+1-2 4+ 1 - term 4+ term -term term + expr -term term + expr (عدم موفقیت در تجزیه)	4+1-2 4+ 1 - term 4+ term -term term + term -term expr + term -term expr - term expr

مقایسه تجزیه پایین به بالا با سمت‌راست ترین اشتقاق

تجزیه پایین به بالا	سمت راست ترین اشتقاق
4+1-2	expr
term +1-2	expr - term
expr + 1 -2	expr - 2
expr + term -2	expr + term - 2
expr - 2	expr + 1 - 2
expr - term	term + 1 - 2
expr	4 + 1 - 2

دستگیره: زیر رشته ای منطبق بر سمت راست یک قانون و ایجاد کننده یک کاهش به غیر پایانه سمت چپ آن قانون

ویژگی دستگیره: زیر رشته ای که با عمل کاهش با توانایی هدایت تجزیه کننده به عنصر شروع گرامر

دستگیره‌ها در تجزیه پایین به بالا رشته 4+1-2

دستگیره	کاهش‌ها	قاعده تولید مورد استفاده
4	4+1-2	term→4
term	term +1-2	expr→term
1	expr + 1 -2	term→1
expr + term	expr + term -2	expr→expr+term
2	expr - 2	term→2
expr - term	expr - term	expr→expr+term
	expr	

S→bBCf
B→Bcd|c
C→e

S
bBCf
bBef
bBcdef
bccdef

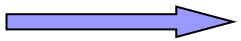
دستگیره‌های کاهش رشته bccdef

کاهش	دستگیره
bccdef	c
bBcdef	Bcd
bBef	e
bBCf	bBCf
S	

مثال دستگیره

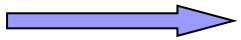
دنباله ورودی abbcde

$S \rightarrow aABe$
$A \rightarrow Abc \mid b$
$B \rightarrow d$



a**b**bcde — انتقال
a**Abc**de — کاهش
aA**d**e — انتقال
aABe — کاهش
S

دستگیره ها



$A \rightarrow b$
$A \rightarrow Abc$
$B \rightarrow d$

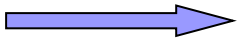
مثال

دنباله ورودی id1 + id2 * id3

(1) $E \rightarrow E + E$
(2) $E \rightarrow E * E$
(3) $E \rightarrow (E)$
(4) $E \rightarrow id$

سمت راست ترین اشتقاق

$E \rightarrow E + E$
$\rightarrow E + E * E$
$\rightarrow E + E * id3$
$\rightarrow E + id2 * id3$
$\rightarrow id1 + id2 * id3$



جمله	دستگیره
id1 + id2 * id3	id1
E + id2 * id3	id2
E + E * id3	id3
E + E * E	E * E

روش تجزیه انتقال – کاهش shift-reduce

روش تجزیه انتقال – کاهش از دو مرحله منحصر به فرد برای تجزیه پایین به بالا استفاده می کند. این مراحل به عنوان shift-step و reduce-stepشناخته می شوند.

مرحله Shiftیا انتقال: (یافتن دستگیره)

مرحله انتقال به پیشروی اشاره گر ورودی به نماد ورودی بعدی اشاره دارد که به آن نماد انتقالی گفته می شود. این نماد روی پشته وارد می شود. نماد انتقالی به صورت یک گره واحد یا منفرد از درخت تجزیه است.

مرحله Reduceیا کاهش : (کاهش دستگیره)

وقتی تجزیه کننده یک قانون گرامری کامل RHSرا پیدا کرد و آن را به LHSجایگزین کرد ، به عنوان reduce-stepیا گام کاهشی شناخته می شود. این فرایند وقتی اتفاق می افتد که عنصر بالایی پشته حاوی یک دستگیره یا handleباشد. برای کاهش، یک عملکرد POPروی پشته اجرا می شود که دستگیره را از پشته خارج می کند و آن را با نماد غیر پایانی LHS جایگزین می کند.

- همه تجزیه کننده‌های پایین به بالا بعد از یافتن دستگیره، آن را کاهش می‌دهند. تفاوت انواع تجزیه کننده‌های پایین به بالا در **نحوه کشف دستگیره‌هاست.**
- مهمترین تجزیه کننده‌های پایین به بالا: ۱-تجزیه کننده عملگر-اولویت

  ۲- تجزیه کننده LR

تجزیه کننده عملگر-اولویت

تجزیه کننده عملگر-اولویت، را می‌توان به سادگی به صورت دستی ایجاد کرد. این تجزیه کننده ضعیف است و بیشتر برای عبارات محاسباتی کاربرد دارد. تجزیه کننده عملگر-اولویت از روابط عملگرها برای تعیین دستگیره‌ها استفاده می کند. تجزیه کننده عملگر-اولویت را برای همه انواع گرامرها نمی‌توان ایجاد کرد. این تجزیه کننده فقط برای گرامرهای موسوم به گرامرهای عملگر قابل تولید است. گرامرهای عملگر، گرامرهایی هستند که دو ویژگی ذیل را داشته باشند.

۱- سمت راست هیچ قاعده تولیدی $\in$ نباشد.

۲- در سمت راست هیچ قاعده تولیدی بیش از یک غیر پایانه مجاور هم وجود نداشته باشد.

نقطه ضعف‌های روش عملگر اولویت

- دشوار بودن اداره نمودن نشانه هایی مانند علامت منها با دو اولویت متفاوت (دودیی یا یگانی)
- عدم اطمینان از نتیجه درست تجزیه به دلیل رابطه نزدیک بین گرامر زبان در حال تجزیه و تجزیه کننده عملگر اولویت
- قابلیت تجزیه بر روی تنها رده کوچکی از گرامرها

این گرامر عملگر نیست، یکی از قواعد سمت راست λ است.

$S \rightarrow ACD$
$A \rightarrow Sc \mid D$
$B \rightarrow Dd \mid \epsilon$

این گرامر عملگر نیست، زیرا دو غیرپایانه مجاور یکدیگردر سمت راستS قرار دارند.

$S \rightarrow AB$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid b$

برای تبدیل به گرامر عملگر می‌توان از جایگذاری استفاده نمود.

$S \rightarrow aAbB \mid aAb \mid abB \mid ab$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid b$

عملگر اولویت – تعیین اولویت‌ها

تعریف ۳ رابطه اولویت مجزای بین هر زوج از پایانه ها ممکن است در یک زبان دو عملگر وجود داشته باشد.

رابطه	مفهوم
a <. b	اولویت a کمتر از b است.
a = b	اولویت a و b یکسان است.
a .> b	اولویت a بیش از b است.

عملگر اولویت روشهای تعیین اولویت

۱- استفاده از شرکت پذیری و اولویت موجود بین خود عملگرها در زبان

۲- ایجاد گرامر غیر مبهمی برای زبان و درخت تجزیه آن با قابلیت انعکاس شرکت پذیری و اولویت صحیح بین عناصر پایانه در درخت

استفاده از اولویت ها

۱- قرار دادن روابط اولویت بین پایانه ها در رشته ورودی به تجزیه

۲- قرار دادن علامت $ ابتدا و انتهای رشته ورودی به همراه اولویت آن نسبت به اولین پایانه و آخرین پایانه رشته

۳- حذف غیر پایانه ها از جمله ورودی

۴- پویش از انتهای چپ رشته تا رسیدن به اولین اولویت <.

۵- پویش به عقب (چپ) از همان نقطه با پشت سرگذاردن هر = تا رسیدن به .<

۶- تعیین دستگیره شامل هر چیزی در سمت چپ اولین راست ۵. در مرحله >و سمت <

۱- راست (توان) بالاترین اولویت و شرکت پذیر از ↑

۲- * و / بالاترین اولویت بعدی و شرکت پذیر از چپ

۳- + و – پایین ترین اولویت و شرکت پذیر از چپ

$E \rightarrow E+E \mid E*E \mid id$

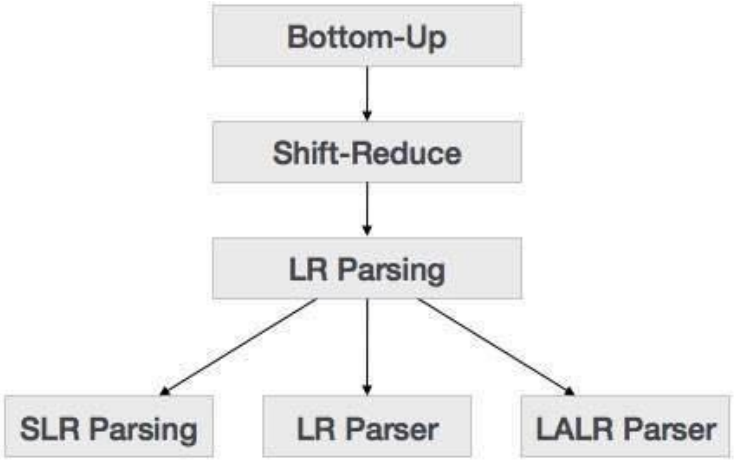
جدول اولویت عملگرها

	id	+	*	$
id		>	>	>
+	<	>	<	>
*	<	>	>	>
$	<	<	<	

مراحل تجزیه رشته id+id*id

دستگیره	عمل	ورودی	رابطه نماد جاری ورودی و نماد بالای پشته	پشته
	انتقال به پشته	$ id + id * id $	<	$
id دستگیره است و کاهش با E→id	کاهش	$ + id * id $	>	$id
	انتقال به پشته	$ id * id $	<	$+
id دستگیره است و کاهش با E→id	کاهش	$ * id $	>	$+id
	انتقال به پشته	$ * id $	<	$+
	انتقال به پشته	$ id $	<	$+*
id دستگیره است و کاهش با E→id	کاهش	$	>	$+*id
* به همراه غیرپایانه‌های طرفین آن که در مراحل قبلی به دست آمده است. دستگیره است و کاهش با E→E*E		$	>	$+*
+ به همراه غیرپایانه‌های طرفین آن که در مراحل قبلی به دست آمده است. دستگیره است و کاهش با E→E+E		$	>	$+
رشته ورودی تمام شده و نماد شروع تولید شده است در نتیجه تجزیه تکمیل می‌گردد.		$		$

تجزیه کننده های LR



دلایل پر طرفداربودن تجزیه کننده های LR

۱- قابلیت تشخیص ساختارهای زبانهای مستقل از متن

۲- عمومی ترین روش تجزیه انتقال کاهش غیر بازگشتی

۳- توانایی تجزیه رده گرامرهای قابل تجزیه پیش گو

۴- سریعترین تشخیص خطای نحوی با پویش چپ به راست

تجزیه LR- نقاط ضعف

۱-کار زیاد در ساخت آن برای گرامر زبان بشکل دستی

۲- نیازمند ابزار مولد تجزیه کننده LR برای ایجاد

مقایسه LL و LR

LL	LR
اشتقاق چپ ترین	اشتقاق راست ترین را بطور معکوس انجام می دهد.
با ریشه غیر پایانی روی پشته شروع می کند.	با ریشه غیر پایانی روی پشته پایان می یابد.
وقتی پشته خالی شود به پایان می رسد.	با یک پشته خالی شروع می شود.
از پشته برای تعیین این که چه چیزی هنوز باقی مانده است استفاده می کند.	از پشته برای تعیین این که چه چیزی قبلا دیده است استفاده می کند.
درخت تجزیه بالا به پایین را می سازد.	درخت تجزیه پایین به بالا را می سازد.
به طور مداوم غیر پایانه را از پشته بر می دارد و سمت راست مربوطه را وارد پشته می کند.	سعی می کند طرف راست را روی پشته تشخیص دهد، آن را بردارد و غیر پایانی متناظر را وارد پشته نماید.
غیر پایانی ها را گسترش می دهد.	غیر پایانی ها را کاهش می دهد.
پایانی ها را می خواند وقتی یک عنصر را از پشته بر می دارد.	پایانی ها را می خواند همزمانی که آن ها را روی پشته وارد می دهد.
درخت تجزیه پیمایش پیش ترتیبی دارد.	درخت تجزیه پیمایش پس ترتیبی دارد.

گرامر LL(K) توانایی تشخیص وقوع سمت راست یک رشته با دیدن تمام آنچه که از آن سمت راست مشتق شده ، با استفاده از K نماد پیش نگر

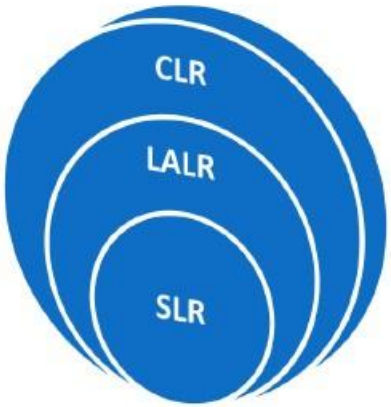
گرامر LR(K) توانایی تشخیص وقوع سمت راست تنها با دیدن اولین K نماد از آنچه توسط سمت راست آن مشتق شده

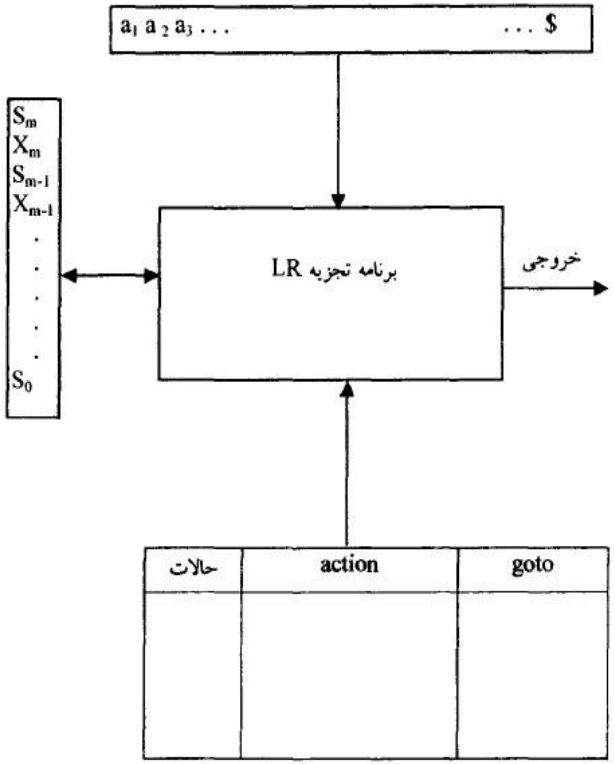
تجزیه‌کننده LR

تجزیه گر LRیک تجزیه کننده غیر بازگشتی، (shift-reduceانتقالی-کاهشی) و پایین به بالا است. تجزیه گر LR از کلاس گسترده ای از گرامر مستقل از متن استفاده می کند که آن را به کار آمدترین روش تجزیه نحوی تبدیل می کند. تجزیه گرهای LRهمچنین به عنوان تجزیه کننده LR (k) شناخته می شود، که Lمخفف اسکن از چپ به راست left-to-right جریان ورودی است. R مخفف ساختار راست ترین اشتقاق به صورت معکوس است و k علامت تعدادی نمادهای رو به جلو برای تصمیم گیری را مشخص می کند.

سه الگوریتم هستند که به طور گسترده‌ای برای ساخت تجزیه‌کننده LRاستفاده می‌شوند:

- **SLR(1)** تجزیه‌کننده LR ساده: (ساده ترین، کمترین توانایی)
  - بر روی دسته کوچک‌ترین گرامرها کار می‌کند.
  - تعداد حالت‌ها معدود است و از این رو جدول بسیار کوچکی دارد.
  - ساخت ساده و آسانی دارد.
- **LR(1)** تجزیه‌کننده LR (CLR) قدرتمندترین، گران‌ترین
  - بر روی مجموعه کاملی از گرامرهای (1)LRکار می‌کند.
  - جدول بزرگی ایجاد می‌کند و تعداد حالت‌ها زیاد است.
  - ساخت کندی دارد.
- **LALR(1)** تجزیه‌کننده LRرو به جلو (LR پیش نگر) قدرت متوسط، هزینه متوسط
  - بر روی گرامرهای با اندازه متوسط کار می‌کند
  - تعداد حالت‌ها همانند الگوریتم (1)SLRاست.





قسمتهای مختلف تجزیه کننده LR

۱- ورودی: رشته ورودی که تجزیه کننده باید آن را تجزیه کند در این قسمت قرار دارد. به انتهای رشته ورودی $ اضافه می‌شود. تجزیه کننده LR با دیدن $ پایان رشته ورودی را تشخیص می‌دهد.

۲- پشته: محتوای پشته به صورت $S_0X_1S_1...X_mS_m$ نشان داده می‌شود. Si نشان دهنده حالات و X_i پایانه‌ها و غیر پایانه‌های گرامر است. نمادهای ورودی به پشته منتقل[1] می‌شوند تا یک دستگیره در پشته یافت شود، پس از کشف دستگیره، کاهش[2] انجام می‌شود.

۳- جدول تجزیه LR: این بخش شامل دو قسمت action و goto است. قسمت action عملی که باید انجام شود و بخش goto حالت بعدی را مشخص می‌کند.

٤- خروجی: دنباله ای از قواعد تولید گرامر است که در تجزیه استفاده شده اند.

۵- برنامه تجزیه کننده LR: این برنامه قسمت اصلی تجزیه کننده است. برنامه تجزیه کننده بر اساس نماد جاری رشته ورودی و حالت بالای پشته و محتوای جدول تجزیه، مرحله بعدی را تعیین می‌کند. نحوه عملکرد برنامه تجزیه کننده را با یک مثال نشان می‌دهیم.

تجزیه انتقال-کاهش

- در تجزیه انتقال-کاهش،یک پشته نمادهای گرامر را نگهداری می کند،وبخشی از رشته ورودی که هنوز تجزیه نشده است در بافر ورودی قرار می گیرد.
- در روند پویش از چپ به راست در رشته ورودی،تجزیه کننده چندین نماد را به پشته انتقال می دهد.و این عمل تا زمانیکه یک دستگیرهβ در بالای پشته ظاهر شود ادامه می یابد.سپسβبا سمت چپ مولد مناسب کاهش می‌یابد.این عمل‌تا زمانیکه خطایی پیدا شود یا پشته شامل نمادشروع و ورودی تهی شود،ادامه می‌یابد.

مزیت هرس نمودن: توانایی تولید معکوس سمت راست ترین اشتقاق

مشکلات هرس نمودن دستگیره

۱- تعیین زیر رشته مناسب برای کاهش در یک شبه جمله راست

۲- انتخاب قانون مناسب در موارد وجود دو یا بیشتر قانون با زیر رشته یکسان در سمت راست

(1) $E \rightarrow E + E$

(2) $E \rightarrow E * E$

(3) $E \rightarrow (E)$

(4) $E \rightarrow id$

مثال هرس نمودن دستگیره

id1 + id2 * id3

شبه جمله راست	دستگیره	قانون کاهشی
id1 + id2 * id3	id1	$E \rightarrow id$
E + id2 * id3	id2	$E \rightarrow id$
E + E * id3	id3	$E \rightarrow id$
E + E * E	E * E	$E \rightarrow E * E$
E + E	E + E	$E \rightarrow E + E$
E		