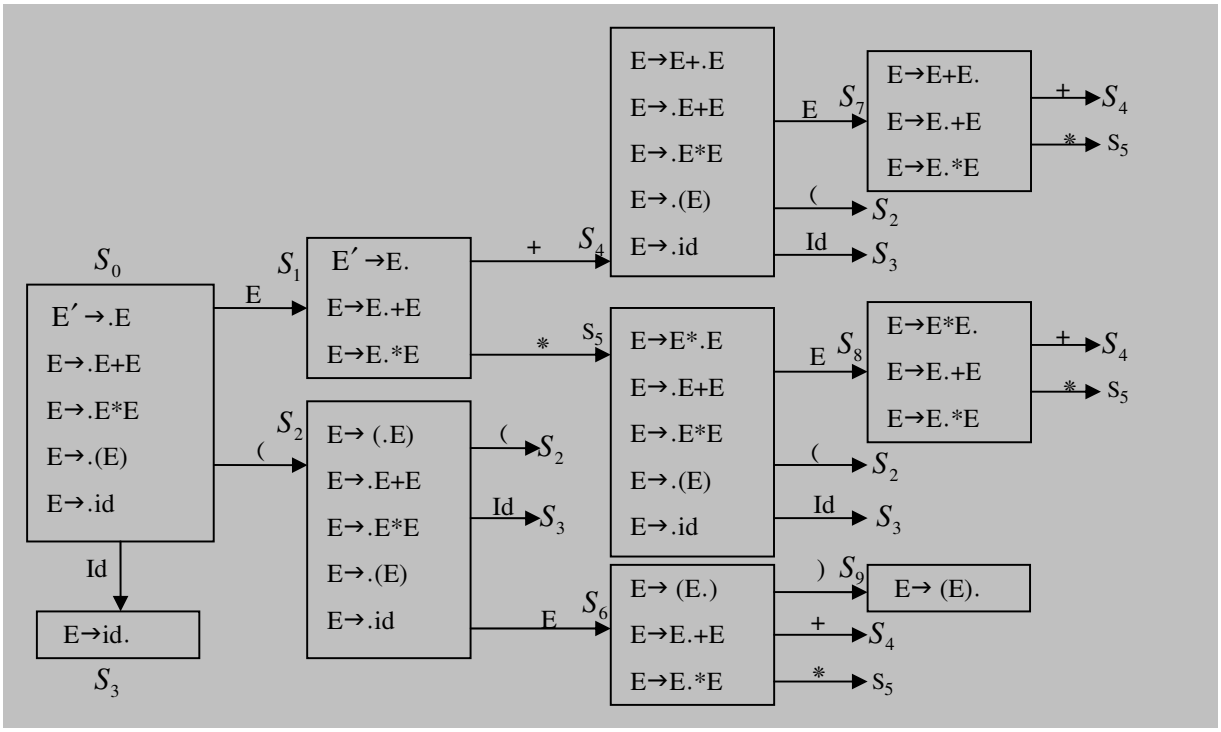
مزیت استفاده ازگرامر های مبهم در تجزیه LR :

با استفاده از ابهام می توان جداول کوچکتری برای تجزیه پایین به بالا ایجاد کرد(مزیت). استفاده از گرامر های مبهم در بسیاری از موارد موجب ایجاد تداخل هائی در جدول تجزیه حواهد شد برای این که عمل تجزیه پایین به بالا امکان پذیر باشد می بایست این تداخل ها را از بین برد . بنابراین با مبهم کردن گرامر گرچه تعداد قوانین و در نتیجه حالات جدول کاهش می یابد اما مشکل تداخل ایجاد می شود.

برای روشن شدن موضوع گرامر $E \to E+T \mid T$, $T \to T*F \mid F$, $F \to id \mid (E)$ را در نظر بگیرید، این گرامر (اگر دیاگرام آن رسم) به یازده حالت(State) نیاز دارد حال می خواهیم دیاگرام مبهم شده این گرامر را که در زیر آمده است رسم کنیم و همچنین جدول تجزیه را ایجاد کرده و به بررسی تداخل ها و تعداد حالت(که کاهش یافته) بپردازیم.

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

رسم دیاگرام:



حال با توجه به دیاگرام رسم شده در صفحه بعد جدول تجزیه را تشکیل می دهیم.

جدول تجزیه دیاگرام صفحه قبل.

$Follow(E) = \{\$,+,*,)\}$

state	action					goto	
	id	+	*	(	)	$	E
0	S_3			S_2			1
1	S_4	S_5				acc	
2	S_3			S_2			6
3		R_4	R_4		R_4	R_4	
4	S_3			S_2			7
5	S_3			S_2			8
6		S_4	S_5		S_9		
7		S_4, R_1	S_5, R_1		R_1	R_1	
8		S_4, R_2	S_5, R_2		R_2	R_2	
9		R_3	R_3		R_3	R_3	

ملاحظه می شود که در چهار خانه تداخل shift/Reduce رخ داده است. جهت از دست ندادن سرعت از گرامر مبهم استفاده می شود و جهت حل تداخل از الویت عملگر ها استفاده می شود.

تجزیه CLR :

آیتم LR(1) : یک ایتم LR(1) یک زوج مرتب متشکل از یک آیتم LR(0) و یک مجموعه پایانه به نام مجموعه پیش بینی(Lookahead) است، و معمولا به صورت $[A \to \alpha., LA]$ نمایش داده می شود رابطه زیر در مورد مجموعه پیش بینی و مجموعه Follow غیر پایانه سمت راست این آیتم ها وجود دارد. $LA \subseteq Follow(A)$

توضیح این که جهت پر کردن جدول تجزیه CLR ابتدا می بایست دیاگرام حالت آن را رسم کنیم دیاگرام حالت CLR مانند SLR است وآیتم شروع $< S' \to .S, \{\$\} >$ می باشد

الگوریتم محاسبه تابع بستار که در مورد آیتم های LR(0) توضیح داده شده بسیار شبیه الگوریتم یافتن بستار آیتم های LR(1) است با این تفاوت که در این جا باید به صورت زیر برای آیتم های غیر هسته ای جدیدی که به مجموعه بستار اضافه می شوند، مجموعه پیش بینی تعیین نمود.

$[A \to \alpha.B\beta, \{a\}]$

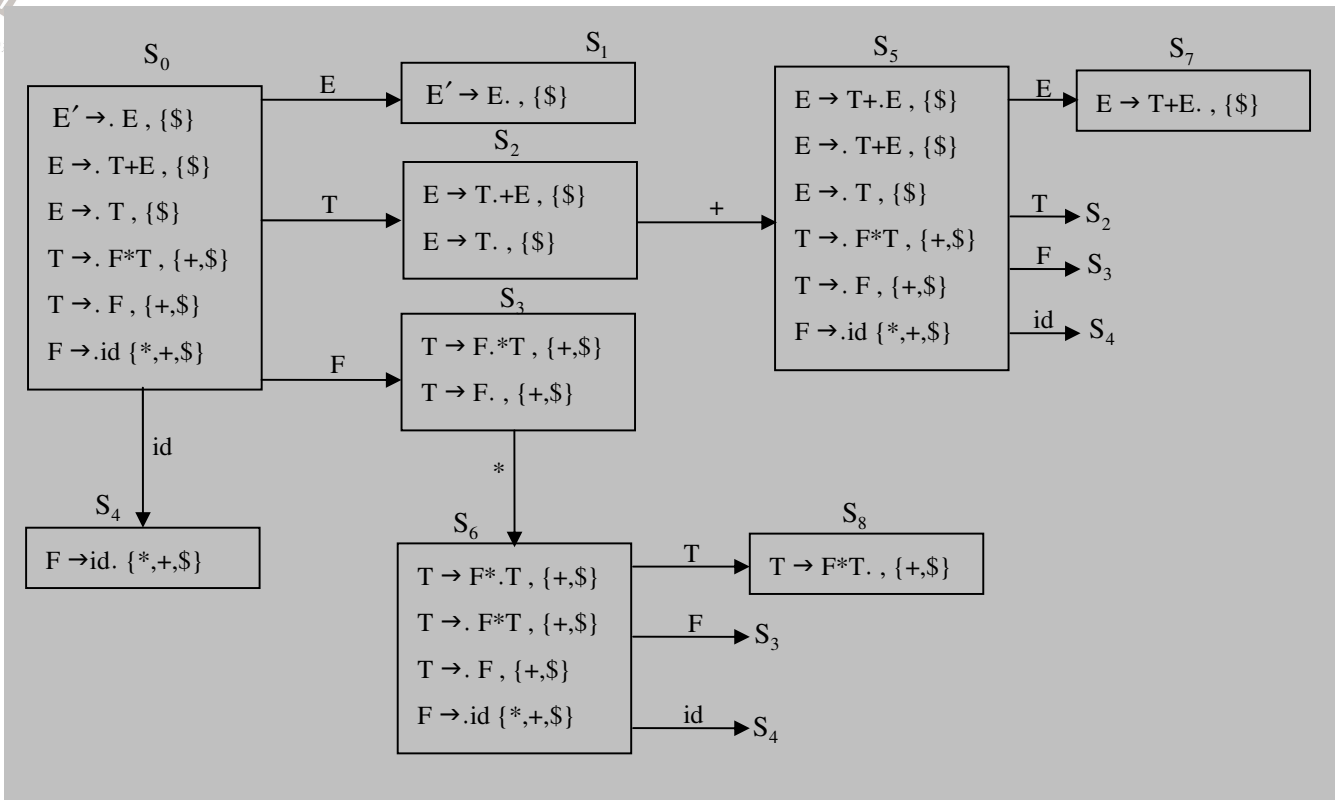
$[B \to .\delta, \{b\}] \mid b \in First(\beta a)$

مثال. دیاگرام حالت CLR گرامر زیر را رسم کنید.

$E' \to E$

1) $E \to T + E$
2) $E \to T$
3) $T \to F * T$
4) $T \to F$
5) $F \to id$

صفحه بعد دیاگرام رسم شده است.



نحوه تکمیل جدول CLR(1) :

روش تهیه جدول CLR(1) بسیار شبیه روشی است که در رابطه با SLR(1) بیان گردید، با این تفاوت که اگر در این جا در وضعیت I آیتمی بـه فـرم $[A \to \alpha\bullet, \{b\}]$ داشته باشیم، در خانه های action[i, b] دستور $A \to \alpha$ Reduce قرار می دهیم، به عبارت دیگر در این جا به جای اسـتفاده از مجموعه Follow ها از مجموعه پیش بینی شده استفاده می کنیم.

مثال. جدول تجزیه را برای دیاگرام مثال قبلی تکمیل کنید.

state	action				goto		
	id	+	*	$	E	T	F
S_0	S_4				1	2	3
S_1				ac			
S_2		S_5		R_2			
S_3		R_4	S_6	R_4			
S_4		R_5	R_5	R_5			
S_5	S_4				7	2	3
S_6	S_4					8	3
S_7				R_1			
S_8		R_3		R_3			

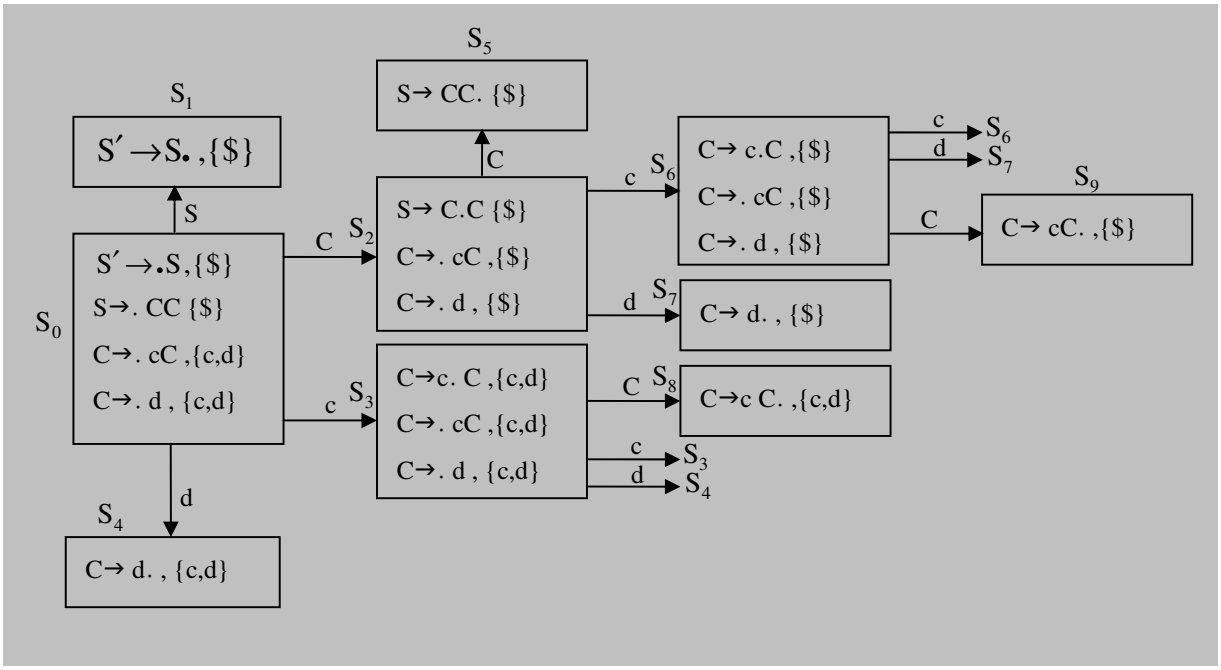
مثال. دیاگرام حالت CLR گرامر زیر را رسم کنید.

$S' \rightarrow S$

1) $S \rightarrow CC$

2) $C \rightarrow cC$

3) $C \rightarrow d$



جدول تجزیه دیاگرام رسم شده.

state	action			goto	
	c	d	$	S	C
S_0	S_3	S_4		1	2
S_1				ac	
S_2	S_6	S_7			5
S_3	S_3	S_4			8
S_4	R_3	R_3			
S_5			R_1		
S_6	S_6	S_7			9
S_7			R_3		
S_8	R_2	R_2			
S_9			R_2		

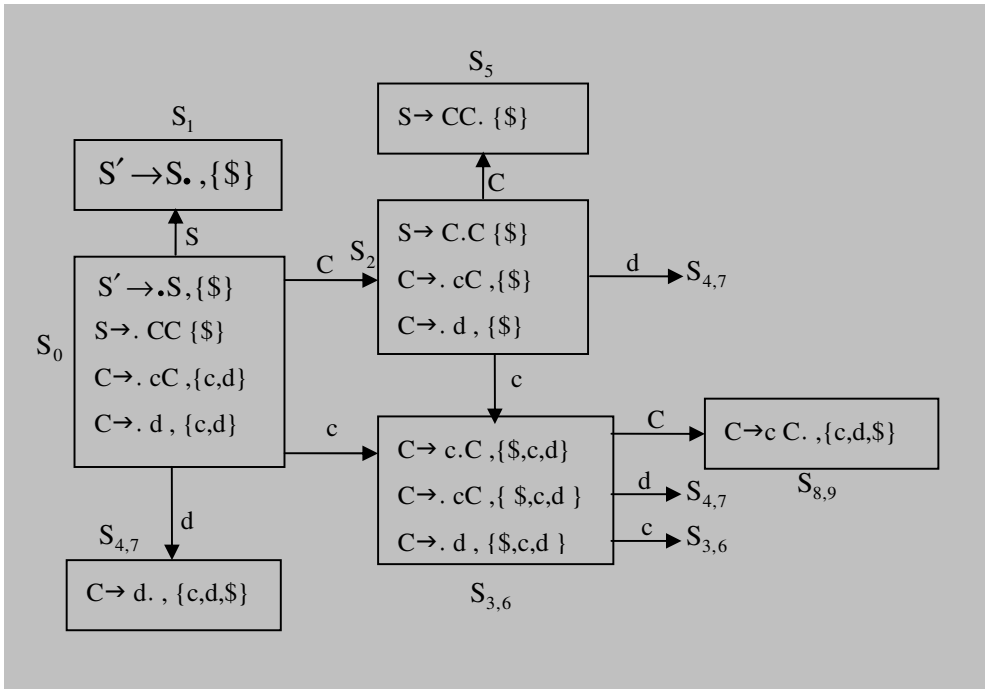
رسم دیاگرام و جدول تجزیه LALR :

برای رسم دیاگرام LALR یک گرامر باید، ابتدا دیاگرام CLR گرامر را رسم کرد. سپس از روی دیاگرام CLR با انجام دو عمل زیر دیاگرام LALR بدست می آید.

ابتدا در دیاگرام CLR به دنبال وضیعت هائی می گردیم که بخش آیتم های LR(0) آنها(که به آن اصطلاحا هسته یا Core یک وضیعت می گویند) یکسان باشد. سپس در دیاگرام وضیعت هائی را که هسته یکسانی دارند ادغام می کنیم و مجموعه پیش بینی آیتم های این وضیعت ها را نظیر به نظیر با هم ادغام می کنیم

مثال. با توجه به دیاگرام CLR صفحه 39 ، دیاگرام LALR مربوطه را رسم کنید.

ملاحظه می شود که در دیاگرام مربوطه S_3 با S_6، S_4 با S_7 و S_8 با S_9 قابل ادغام هستند(هسته یکسان دارند)



اگر گرامری CLR باشد و پس از ادغام وضیعت های مزبور در جدول تجزیه LALR گرامر نیز تداخلی بوجود نیاید می توان نتیجه گرفت که گرامر LALR است. حال اگر گرامری CLR بوده و LALR نباشد، پس از ادغام در جدول تجزیه LALR گرامر تداخل نوع کاهش/کاهش بروز خواهد کرد، هیچگاه در اثر ادغام در جدول LALR تداخل نوع انتقال-کاهش بوجود نخواهد آمد مگر آنکه گرامر CLR هم نبوده باشد که در این صورت قطعا LALR هم نیست، به عبارتی در چنین حالتی، تداخل قطعا قبل از ادغام نیز وجود داشته است. طریقه تهیه جدول تجزیهLALR از روی دیاگرام ادغام شده عینا همانند روش تهیه جدول CLR است. به عنوان یک نمونه جدول تجزیه LALR مثال قبل به صورت زیر خواهد بود.

state	action			goto	
	c	d	$	S	C
0	$S_{3,6}$	$S_{4,7}$		1	2
1			acc		
2	$S_{3,6}$	$S_{4,7}$			5
36	$S_{3,6}$	$S_{4,7}$			89
47	R_3	R_3	R_3		
5			R_1		
89	R_2	R_2	R_2		

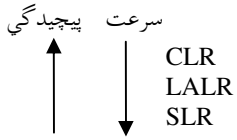
تعداد وضعیت های جداول تجزیه SLR و LALR هر گرامری دقیقا یکسان است لیکن تعداد وضعیت های جدول تجزیهCLR گرامرها به مراتب بزرگتر از جداول SLR و LALR است. تنها عیب جزئی روش LALR نسبت به CLR این است که خطا های نحوی را ممکن است قدری دیرتر کشف کند. البته هیچ یک از این سه روش پس از رسیدن به یک توکن غلط آن را به داخل انباره انتقال نخواهد داد. در CLR پس از رسیدن به یک توکن خطا، دیگر حتی عمل کاهش نیز انجام نخواهد شد، لیکن در مورد SLR و LALR ممکن است خطا پس از اجرای چند عمل کاهش اضافی کشف شود. با این حساب CLR سریعتر است.

اگر گرامر CLR باشد آیا می توان نتیجه گرفت که گرامر LALR است؟

خیر زیرا در جدول تجزیه LALR ممکن است تداخل r-r داشته باشیم.

اگر گرامری SLR باشد آن گرامر CLR و هم LALR است

مقایسه سرعت و پیچیدگی



تحلیل معنایی Type checking

به صورت پویا : در زمان اجرا عمل تحلیل معنایی انجام میشود.کامپایلر باید علامت خاصی را در لابه لای کد اجرا قرار دهد به عنوان مثال فرض کنید آرایه ۱۰ تایی تعریف کرده اید در برنامه طبق زیر داریم

```
Int A[10];          A[i]=5
```

در اینجا مقدار i در زمان اجرا مشخص میشود در این جا کامپایلر علامتی را قرار میدهد تا مشخص کند که مقدار i در حین اجرا تعیین میشود

به صورت ایستا:

۱- یکسان بودن گونه(Type checking): اجازه نداریم آرایه را با متغییر جمع کنیم. عملگرها باید عملوند یکسان داشته باشند.

۲- یکسان بودن نام (واحد بودن نام): نمیتوان یک نام را دو جا تعریف کرد

۳- کنترل جریان: مثلا در swich در case باید Break داشته باشیم وگرنه خطا رخ میدهد

۴- بررسی ساختارهای تودرتو: مثلا در زبان Ada هر حلقه یک نام مخصوص برای خود دارد

```
For1(……)
      For2(……..)
      End for2
End for1
```

مثال:

اگر a یک متغییر از نوع آرایه یک بعدی با اندیس های ۱ تا۱۰ و I یک عدد صحیح باشد خطای خارج از محدوده(sub scrip out of vong) که در دستورات B:=A[i] ؛ I=11 وجود دارد در حالت کلی در چه زمانی و توسط چه برنامه ای قابل کشف است؟

۱- زمان اجرا توسط سیستم عامل

۲- زمان کامپایل توسط تجزیه کننده دستوری

۳- زمان کامپایل توسط تجزیه کننده معنایی

۴- زمان اجرا توسط خود برنامه حاوی دستورات فوق

جواب: گزینه 4 درست است، اگر B := A[11] بود آنگاه خطا در مرحله کامپایل توسط تحلیل گر مفهومی کشف می شد

نکته: پیش پردازنده ها در زمان کامپایل مشخص میشوند.

سوال :کدام یک از گزینه ها در خصوص گرامر نوع مقابل صحیح است؟

1) $S \rightarrow aACb$

2-3) $A \rightarrow b \mid \lambda$

3-4) $C \rightarrow cC \mid \lambda$

۱)گرامر LL(1) نیست، SLR(1) هم نیست

۲) گرامر LL(1) نیست ولی SLR(1) هست

۳) گرامر LL(1) و SLR(1) است

۴) گرامر LL(1) هست ولیSLR(1) نیست

جواب. گزینه 1 درست است

	S	A	C
First	{a}	{b,ε}	{c,ε}
Follow	{$}	{c,b}	{b}

جدول پارسینگ LL(1) به شکل زیر خواهد شد.

	a	b	c	$
S	1			
A		2/3	3	
C		4	4	

پس گرامر بالا LL(1) نیست ، اگر دیاگرام و جدول تجزیه این گرامر را رسم کنیم ملاحظه می شود که ، SLR(1) هم نیست

مثال: گرامر زیر برای یک پارسر بالا به پایین پیشگو کننده مناسب نیست مشکل را مشخص کنید و گرامر را طوری بنویسید که برای بالا به پایین پیشگو کننده مشخص کنید و بعد برای گرامر جدید جدول تجزیه LL(1) را بسازید.

$L \rightarrow Ra \mid Qba$

$R \rightarrow aba \mid Caba \mid Rbc$

$Q \rightarrow bbc \mid bc$

اگر گرامری بازگشتی چپ داشته باشد آیا LL(1) هست یا نه؟

خیر چون برای LL(1) بودن باید مبهم نباشد و بازگشتی چپ و.. نداشته باشد

نکته) اگر گرامری مبهم نباشد و فاکتورگیری چپ روی آن اعمال شود وبازگشتی راست دارد LL(1) است و اگر گرامری بازگشتی راست داشته باشد ممکن استLL(1) باشد یا نباشد

نکته: اگر گرامرCLR نباشد حتما LALR نیز نیست

پایان

موفق باشید