حوری دهش 9821413

سوال 1:

الف)

طبق شکل زیر:

LL(1) زیرمجموعه ای از زبان‌های Context-Free یا CFG است که با استفاده از یک میزان خطا مشخص (1) و با استفاده از خواص Leftmost derivation و Lookahead قابل تولید می باشد.

LR(0) به عنوان یک زیرمجموعه از زبان های Context-Free شناخته می شود و ماشین های LR(0) از اصول کاهش استفاده می کنند و به وسیله تحلیل کاراکترهای ورودی با استفاده از زبان های دارای ساختار، یک درخت پارس را تولید می کنند.

LR(1) هم به عنوان یک زیرمجموعه از زبان های Context-Free شناخته میشود. همچنین LR(1) یک توسعه از LR(0) است که در آن علاوه بر استفاده از حالت کاهش، یک علامت پیش نیاز (Lookahead) نیز در نظر گرفته میشود.

بنابراین در سلسله مراتب زبان های چامسکی، زبان های LL(1)، LR(0) , LR(1) همه به عنوان زیرمجموعه‌هایی از زبان های Context-Free یا CFG شناخته میشوند.

1. Type 0 known as Unrestricted Grammar.
2. Type 1 known as Context Sensitive Grammar.
3. Type 2 known as Context Free Grammar.
4. Type 3 Regular Grammar.

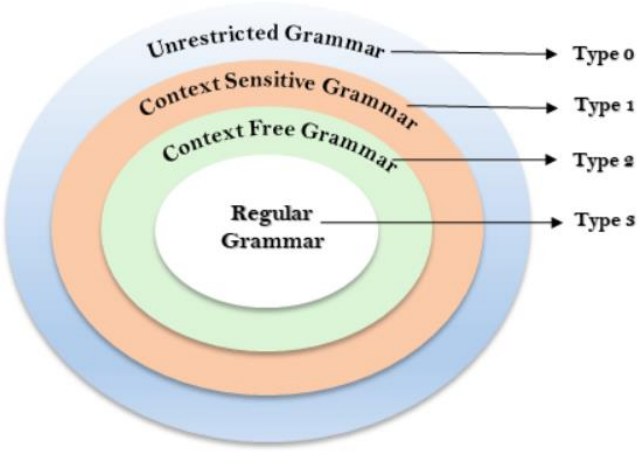


Fig: Chomsky Hierarchy

ب)

دلیل اصلی این امر این است که ادغام وضعیت ها با هسته‌های مشابه تنها تغییرات ساختاری در نمودار ایجاد می کند و توانایی تشخیص تصادمات بر اساس lookahead ورودی حفظ میشود. به عبارت دیگر وقتی که دو وضعیت با هسته های مشابه ادغام میشوند، lookahead های مربوط به هر دو وضعیت نیز با هم ترکیب میشوند به گونه ای که تصادمات shift/reduce با توجه به اطلاعات lookahead تشخیص داده میشوند. از این رو، ادغام وضعیت ها با هسته های مشابه نباید به ایجاد تصادمات جدیدی منجر شود زیرا معیارهای تصمیم گیری بر اساس lookahead های مربوطه همچنان حفظ میشود.

پس در حالت کلی در تبدیل نمودار LR(1) به LALR(1)، وضعیت هایی با هسته مشابه را با هم ادغام می‌کنیم و این ادغام ها نمی توانند منجر به تصادم shift/reduce جدیدی شوند چون در حقیقت ما فقط وضعیت‌هایی را ادغام می کنیم که هسته آنها یکسان است.

هسته یک وضعیت، مجموعه ای از آیتم های LR(1) است که بدون توجه به lookahead تولید میشوند یعنی هسته وضعیت شامل آیتم هایی است که می توانند با یکدیگر تصادم داشته باشند. اگر دو وضعیت هسته مشابه داشته باشند، این به این معنی است که آنها در واقع همان وضعیت هستند، فقط lookahead آنها متفاوت است. بنابراین ادغام این دو وضعیت نمی تواند تصادم جدیدی ایجاد کند زیرا هر تصادمی که وجود دارد قبلا در هسته وجود داشته است.

ج)

پارسرها برای تجزیه و تحلیل زبان های غیرمحدود (به عنوان مثال زبان های غیرقابل پذیرش توسط ماشین‌های تورینگ) به مشکل برمی‌خورند به علت اینکه پارسرها به تعداد نامحدودی حالت نیاز دارند که بتوانند با ورودی‌های مختلف سازگار شوند. از این رو، یک پارسر معمولی نمی تواند با استفاده از یک ماشین تورینگ ساخته شود زیرا ماشین های تورینگ محدود به تعداد متناهی حالت هستند.

به طور معمول برای تجزیه و تحلیل زبان های CFL از پارسرهایی استفاده میشود که قابلیت تطبیق با تعداد محدودی حالت را دارند و این تطابق به اندازه کافی برای پوشش زبان های CFL است. اما زبان هایی که توسط ماشین های تورینگ شناسایی میشوند، ممکن است حاوی الگوها و ویژگی هایی باشند که نیازمند تعداد حالت نامحدودی برای تجزیه و تحلیل آنها باشند، که این امکان با استفاده از یک پارسر محدود مانند یک پارسر مبتنی بر ماشین تورینگ، فراهم نمی شود.

بنابراین علتی که پارسرهای مبتنی بر ماشین تورینگ برای زبان های غیرقابل پذیرش توسط ماشین های تورینگ معمولا ساخته نمی شوند، این است که نیاز به تطبیق با تعداد نامحدودی حالت وجود دارد که این با توجه به محدودیت ماشین های تورینگ ممکن نیست.

سوال 2:

الف)

نادرست است چون ترتیب اشتباهی بین LR(2) , LR(1) در نظر گرفته شده است یعنی باید بگوییم که:

LR(1) ⊂ LR(2) است چون هر گرامری که به LR(1) قابل تجزیه باشد به LR(2) نیز قابل تجزیه است پس LR(1) ⊂ LR(2) باید باشد نه برعکس

پس صحیح آن به صورت زیر است:

LR(0) ⊂ SLR(1) ⊂ LALR(1) ⊂ LR(1) ⊂ LR(2) ⊂ LR(k)

مثال نقض:

$S \rightarrow abc \mid Abd$

$A \rightarrow a$

ب)

درست است. از آنجایی که LR(1) ⊂ LR(k) برای هر K>1 می باشد بنابراین این کار امکان پذیر است.

همچنین برای هر گرامر LR(k) با K>1، می توان یک گرامر LR(1) معادل یافت که همان زبان را تشخیص می‌دهد. این کار از طریق استفاده از combined states انجام میشود که در آنها اطلاعات مربوط به K توکن به جلو (lookahead) به صورت یکپارچه در حالت های LR(1) گنجانده میشود. به عبارت دیگر، گرامر LR(k) می تواند به یک گرامر LR(1) معادل تبدیل شود که همان توکن های نگاه به جلو را در حالت های خود گنجانده است.

ج)

نادرست است چون هر گرامر غیر مبهمی SLR(1) نیست مثلا گرامر زیر غیر مبهم است ولی SLR(1) نیست چون دارای تضادف shift-reduce می باشد.

مثال نقض:

$S \rightarrow aAd \mid bBd$

$A \rightarrow \varepsilon$

$B \rightarrow \varepsilon$

سوال 3:

$$n3 = n2 < n1$$

سوال 4:

می دانیم:

گرامر بدون قوانین اپسیلون یا واحد (A->B) است.

در یک تجزیه کننده پایین به بالا، کاهش زمانی اتفاق می افتد که یک رشته از نمادهای ورودی با یک سمت راست یک قانون گرامر مطابقت داشته و جایگزین آن با سمت چپ قانون گرامر شود.

هر کاهش به معنای جایگزینی تعدادی نماد در رشته ورودی با یک نماد دیگر است.

اگر رشته ای به طول n داشته باشیم، در هر کاهش حداقل یک نماد از رشته کاهش می یابد.

پس فرض میکنیم تجزیه کننده پایین به بالا یک رشته به طول n را تجزیه می کند و در هر کاهش، حداقل یک نماد از رشته حذف می شود پس برای یک رشته به طول n، حداکثر تعداد کاهش ها برابر است با تعداد نمادهای موجود در رشته اصلی یا به عبارت دیگر:

حداکثر تعداد کاهش ها = n-1

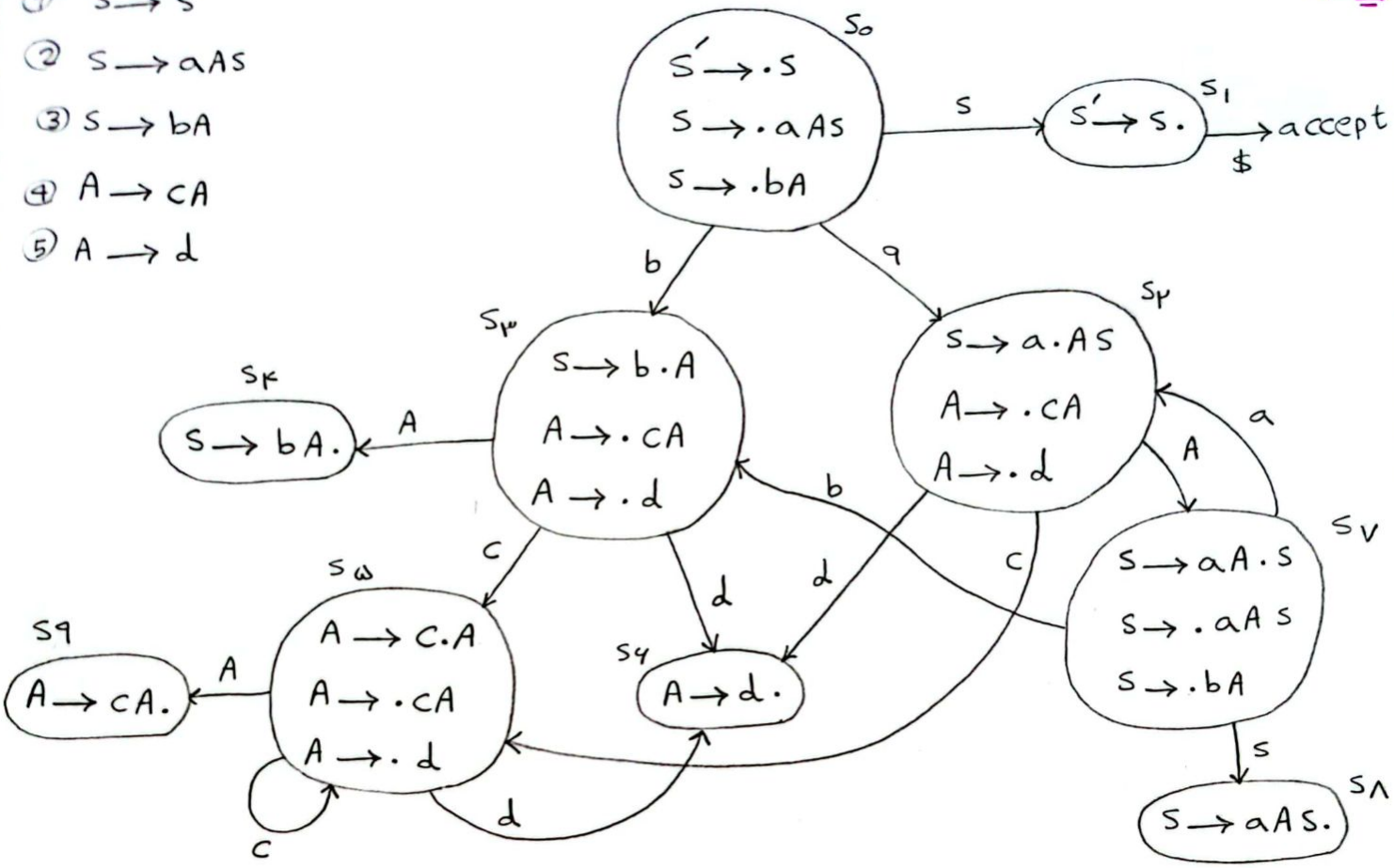
① $S' \rightarrow S$

② $S \rightarrow aAS$

③ $S \rightarrow bA$

④ $A \rightarrow cA$

⑤ $A \rightarrow d$



	action					goto	
حالت	a	b	d	c	$	S	A
0	s_2	s_3				1	
1					accept		
2			s_4	s_5			7
3			s_4	s_5			8
4					r_3		
5			s_4	s_5			9
6	r_5	r_5			r_5		
7	s_2	s_3				10	
8					r_2		
9	r_4	r_4			r_4		

Nodes:

۴ : follow(S) = {$}

۹ : follow(A) = {$, a, b}

پشته	رشته ورودی	عملیات
۰	acdbd$	S۲
۰a۲	cdbd$	S۳
۰a۲c۳	dbd$	S۹
۰a۲c۳d۹	bd$	r۳: A → d
۰a۲c۳A	bd$	goto[۳, A] = ۶
۰a۲c۳A۶	bd$	r۴: A → cA
۰a۲A	bd$	goto[۲, A] = ۷
۰a۲A۷	bd$	S۴
۰a۲A۷b۴	d$	S۹
۰a۲A۷b۴d۹	$	r۳: A → d
۰a۲A۷b۴A	$	goto[۴, A] = ۶
۰a۲A۷b۴A۶	$	r۲: S → bA
۰a۲A۷S	$	goto[۷, S] = ۸
۰a۲A۷S۸	$	r۱: S → aAS
۰S	$	goto[۰, S] = ۱
۰S۱	$	accept

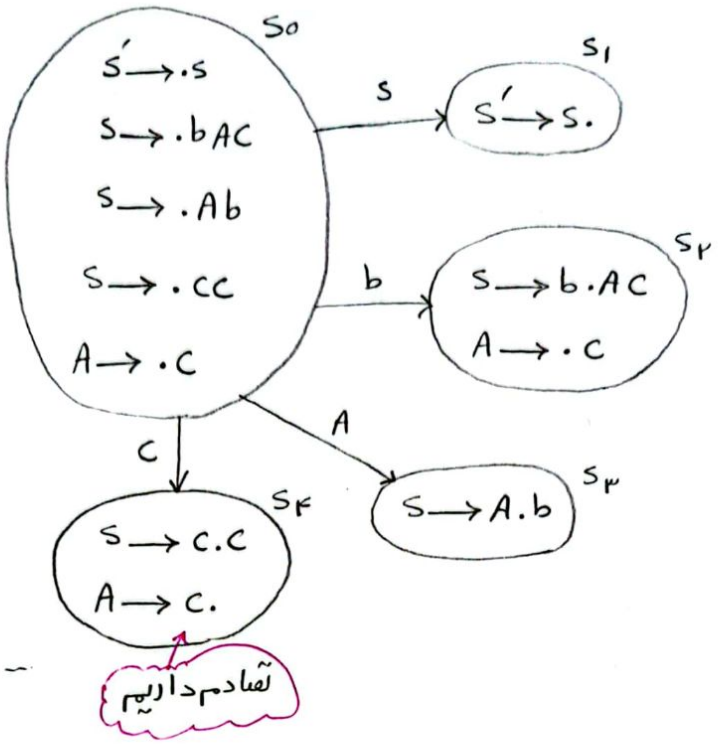
$S' \to S$

$S \to bAC$

$S \to Ab$

$S \to cc$

$A \to c$



توی حالت S4 تصادم داریم بخاطر وجود shift - reduce ← و چون توی این حالت به این برخوردم دیلم

ادامه ندادم ←

shift => $S \to c.c$

← یعنی توی حالت ۴ به ازای c به این حالت دیگر می رود مثلا S_K

reduce => $A \to c.$

← چون گرامر ما LR(1) است باید فالوی A را حساب کنیم که $follow(A) = \{c, b\}$

یعنی توی حالت ۴ به ازای c, b این r_K مثلا داریم

پس توی حالت ۴ به ازای c هم shift داریم و هم reduce

① $S' \rightarrow S$

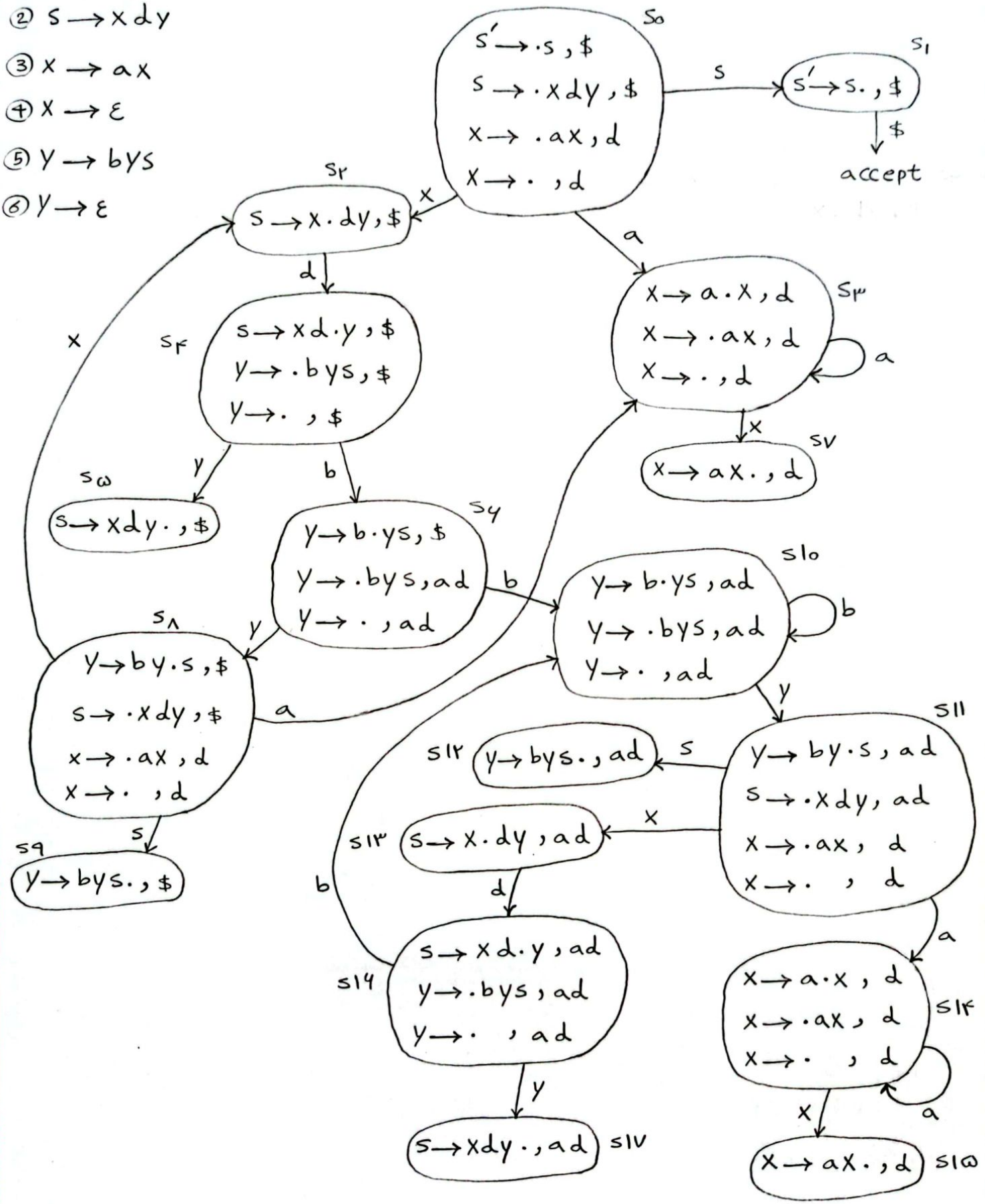
② $S \rightarrow xdy$

③ $x \rightarrow ax$

④ $x \rightarrow \varepsilon$

⑤ $y \rightarrow bys$

⑥ $y \rightarrow \varepsilon$



	action				goto		
حالت	a	b	d	$	S	X	Y
0	s۳		r۴		۱	۲	
1				accept			
2			s۶				
3	s۳		r۶			۷	
4		s۹		r۹			ω
5				r۲			
6	r۹	s۱۰	r۹				۸
7			r۳				
8	s۳		r۴		۹	۲	
9				rω			
10	r۹	s۱۰	r۹				۱۱
11	s۱۴		r۴		۱۲	۱۳	
12	rω		rω				
13			s۱۹				
14	s۱۴		r۶			۱ω	
15			r۳				
16	r۹	s۱۰	r۹				۱۷
17	r۲		r۲				

پشته	رشته ورودی	عملیات
0	aadbbadd $	S_3
0a3	adbbadd $	S_3
0a3a3	dbbadd $	$r_4: X \to \varepsilon$
0a3a3X	dbbadd $	goto [3, X] = 7
0a3a3X7	dbbadd $	$r_3: X \to aX$
0a3X	dbbadd $	goto [3, X] = 7
0a3X7	dbbadd $	$r_3: X \to aX$
0X	dbbadd $	goto [0, X] = 2
0X2	dbbadd $	S_4
0X2d4	bbadd $	S_9
0X2d4b9	badd $	S_{10}
0X2d4b9b10	add $	$r_9: Y \to \varepsilon$
0X2d4b9b10Y	add $	goto [10, Y] = 11
0X2d4b9b10Y11	dd $	$r_4: X \to \varepsilon$
0X2d4b9b10Y11X	dd $	goto [11, X] = 13
0X2d4b9b10Y11X13	dd $	S_{14}
0X2d4b9b10Y11X13d14	d $	$r_9: Y \to \varepsilon$
0X2d4b9b10Y11X13d14Y	d $	goto [14, Y] = 17
0X2d4b9b10Y11X13d14Y17	d $	$r_2: S \to Xdy$
0X2d4b9b10Y11S	d $	goto [11, S] = 12
0X2d4b9b10Y11S12	d $	$r_5: Y \to byS$
0X2d4b9Y	d $	goto [9, Y] = 8
0X2d4b9Y8	d $	$r_4: X \to \varepsilon$
0X2d4b9Y8X	d $	goto [8, X] = 2
0X2d4b9Y8X2	d $	S_4
0X2d4b9Y8X2d4	$	$r_9: Y \to \varepsilon$
0X2d4b9Y8X2d4Y	$	goto [4, Y] = 5
0X2d4b9Y8X2d4Y5	$	$r_2: S \to Xdy$

$0X_2d_4b_6y_9y_{10}S$	$	goto [10, S] = 9
$0X_2d_4b_6y_9y_{10}S_9$	$	r_ω: y → bys
$0X_2d_4y$	$	goto [4, y] = ω
$0X_2d_4y_\omega$	$	r_2: S → x dy
0S	$	goto [0, S] = 1
0S1	$	accept

right sentential	handle	reducing production
aaa*a++	a	S → a
aaS*a++	a	S → a
aSS*a++	SS*	S → SS*
aSa++	a	S → a
aSS++	SS+	S → SS+
aS+	a	S → a
SS+	SS+	S → SS+

right sentential	handle	reducing production
SSS+a*+	SS+	S → SS+
SSa*+	a	S → a
SSS*+	SS*	S → SS*
SS+	SS+	S → SS+

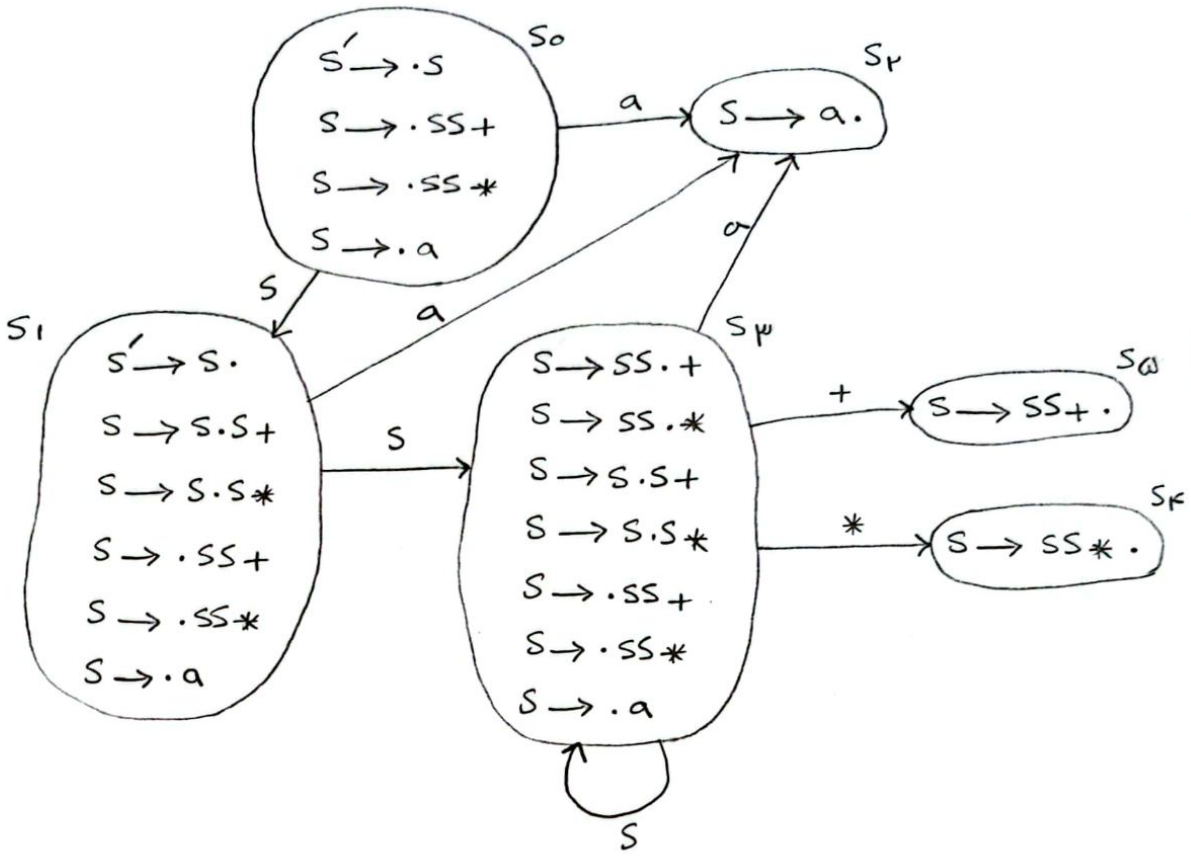
SLR(1)

① $S' \rightarrow S$

② $S \rightarrow SS+$

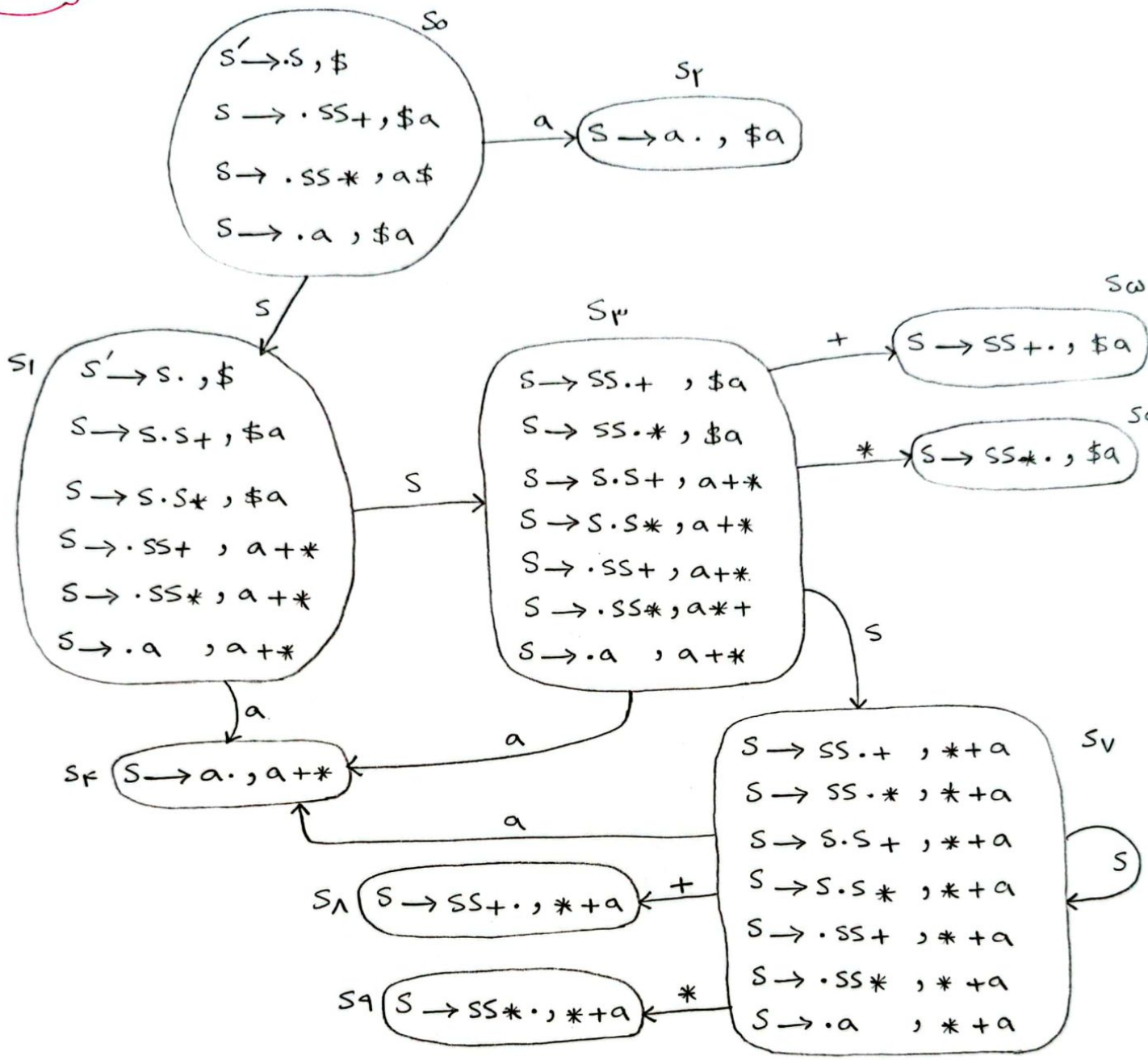
③ $S \rightarrow SS*$

④ $S \rightarrow a$



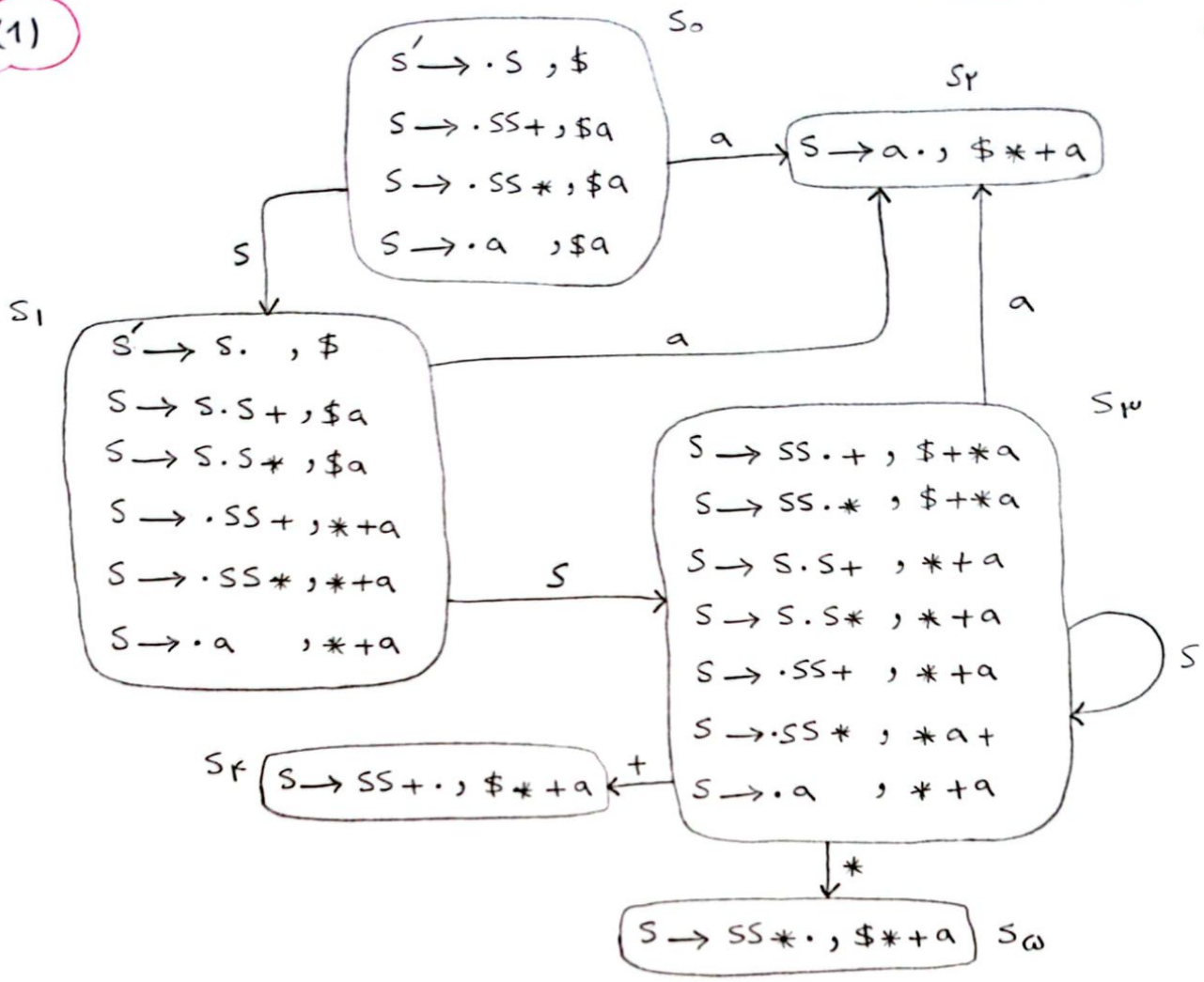
حالت	*	+	a	$	S
0			S_2		1
1			S_2	accept	3
2	r_4	r_4	r_4	r_4	
3	S_4	S_5	S_2		3
4	r_3	r_3	r_3	r_3	
5	r_2	r_2	r_2	r_2	

Follow(S) = {$, a, +, *}



حالت	*	+	a	$	S
۰			S۲		۱
۱			S۴	accept	۳
2			r۴	r۴	
3	S۴	S۵	S۴		۷
۴	r۴	r۴	r۴		
5			r۲	r۲	
6			r۳	r۳	
7	S۹	S۸	S۴		۷
8	r۲	r۲	r۲		
۹	r۳	r۳	r۳		

LALR(1)



حالت	*	+	a	$	S
0			S_2		1
1			S_2	accept	۳
2	r_4	r_4	r_4	r_4	
3	S_5	S_4	S_2		۳
4	r_2	r_2	r_2	r_2	
5	r_3	r_3	r_3	r_3	

ج-۱ ←

پشته	رشته ورودی	عملیات
0	aa*aa+a+*$	s_2
0a2	a*aa+a+*$	r_4: S → a
0S	a*aa+a+*$	goto[0,S] = 1
0S1	a*aa+a+*$	s_2
0S1a2	*aa+a+*$	r_4: S → a
0S1S	*aa+a+*$	goto[1,S] = 3
0S1S3	*aa+a+*$	s_5
0S1S3*5	aa+a+*$	r_3: S → SS*
0S	aa+a+*$	goto[0,S] = 1
0S1	aa+a+*$	s_2
0S1a2	a+a+*$	r_4: S → a
0S1S	a+a+*$	goto[1,S] = 3
0S1S3	a+a+*$	s_2
0S1S3a2	+a+*$	r_4: S → a
0S1S3S	+a+*$	goto[3,S] = 3
0S1S3S3	+a+*$	s_4
0S1S3S3+4	a+*$	r_2: S → SS+
0S1S	a+*$	goto[1,S] = 3
0S1S3	a+*$	s_2
0S1S3a2	+*$	r_4: S → a
0S1S3S	+*$	goto[3,S] = 3
0S1S3S3	+*$	s_4
0S1S3S3+4	*$	r_2: S → SS+
0S1S	*$	goto[1,S] = 3
0S1S3	*$	s_5
0S1S3*5	$	r_3: S → SS*
0S1	$	accept

برای هر A_i که در بازهٔ $1 < i < n$ قرار دارد یک قاعده به صورت $S \rightarrow A_i b_i$ داریم بنابراین تعداد این قواعد به اندازهٔ تعداد مقادیر i در این بازه است که برابر است با $n-2$ (چون $1 < i < n$)

برای هر A_i ۲ نوع قاعده داریم:

$$A_i \rightarrow a_j A_i$$

$$A_i \rightarrow a_j$$

برای هر A_i که در بازهٔ $1 < i < n$ قرار دارد تعداد jهایی که $j \neq i$ هستند برابر است با $n-2$

(چون $1 < j < n$ و $j \neq i$)

پس برای هر A_i ۲ نوع قاعده داریم که برای هر j تکرار می‌شوند ← پس برای هر A_i ، $2(n-2)$ قاعده داریم

*

تعداد قواعد برای S ← $n-2$

برای هر A_i تعداد قواعد $2(n-2)$ و تعداد A_i ها نیز $n-2$ است پس :

$$(n-2) \times 2(n-2) = 2(n-2)^2$$

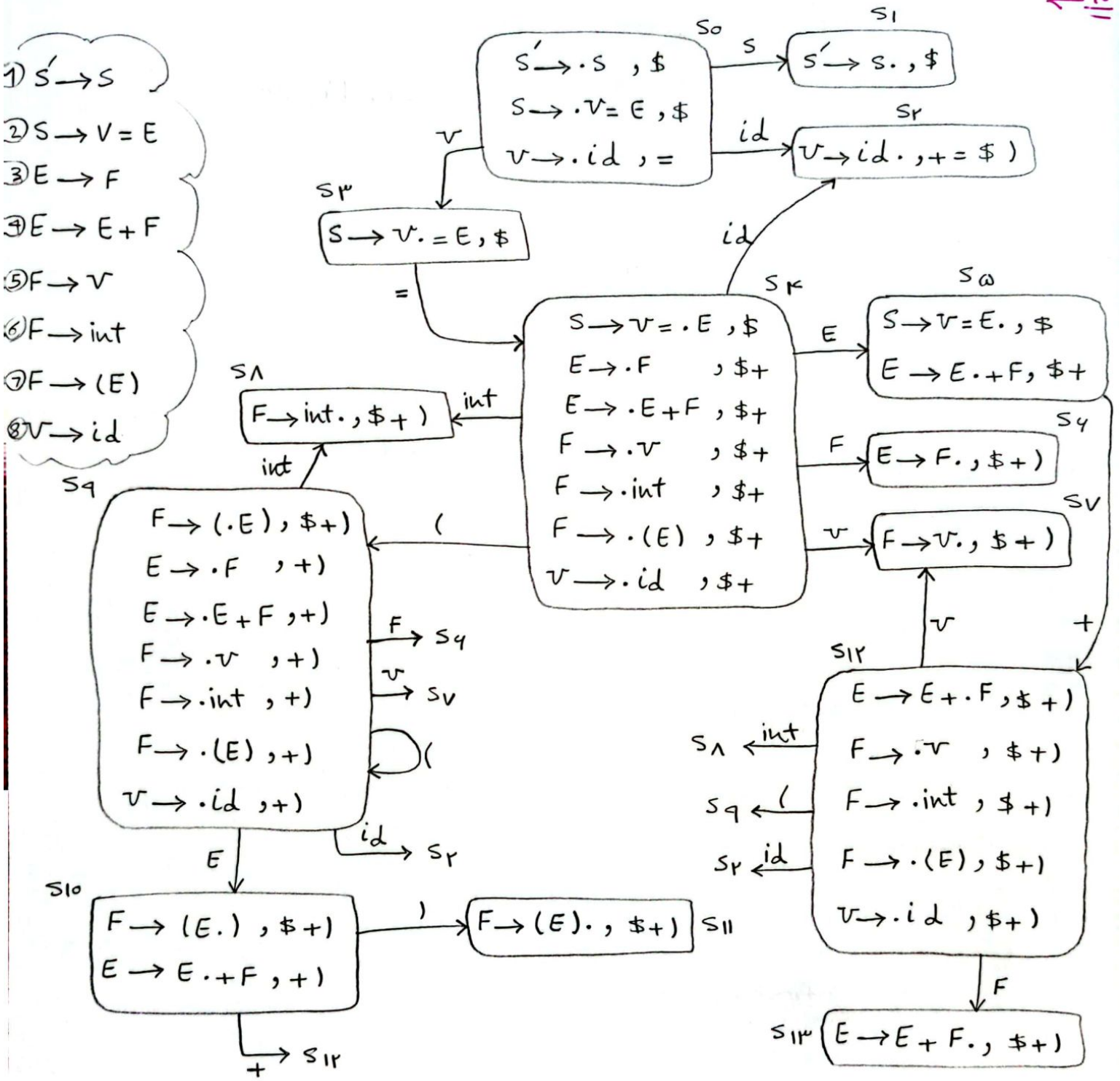
جمع کل تعداد قواعد : $(n-2) + 2(n-2)^2$

بررسی SLR(1) بودن : برای اینکه بگوییم گرامر SLR(1) باشد باید مشخص شود که آیا گرامر دارای تناقض در جدول تجزیه SLR(1) است یا نه . برای این کار باید مجموعه‌های follow و first محاسبه شوند و در نهایت گرفته شوند

بررسی تناقضات : گرامری که به ما داده شده دارای تولیداتی است که باعث می‌شوند یک ساختار بازگشتی به سمت چپ ایجاد شود. ($A_i \rightarrow a_j A_i$) در نتیجه ممکن است در شرایطی که تولیدات مختلفی می‌توانند از یک نقطه شروع شوند، امکان وجود تناقضات در جدول تجزیه SLR(1) باشد.

از آنجا که تولیدات $A_i \rightarrow a_j A_i \mid a_j$ می‌توانند باعث ایجاد تناقض بین shift و reduce شوند (به‌خصوص در مواردی که چنین تولیدهایی مشابه وجود دارد) احتمالاً این گرامر SLR(1) نیست .

در نهایت با توجه به ساختار تولیدات و احتمال ایجاد تناقضات، به احتمال زیاد این گرامر SLR(1) نیست



←۱۵

حالت	=	+	int	(	)	id	$	S	E	F	T
۰						$S_۲$		۱			۳
۱							accept				
2	$r_۸$	$r_۸$			$r_۸$		$r_۸$				
3	$S_۴$										
۴			$S_۸$	$S_۹$		$S_۲$			۵	۶	۷
5		$S_{۱۲}$					$r_۲$				
6		$r_۳$			$r_۳$		$r_۳$				
7		$r_۵$			$r_۵$		$r_۵$				
8		$r_۶$			$r_۶$		$r_۶$				
۹			$S_۸$	$S_۹$		$S_۲$			۱۰	۶	۷
10		$S_{۱۲}$			$S_{۱۱}$						
11		$r_۷$			$r_۷$		$r_۷$				
12			$S_۸$	$S_۹$		$S_۲$				۱۳	۷
13		$r_۴$			$r_۴$		$r_۴$				

تضاد م ندارد