



## طراحی کامپایلرها

نیم‌سال دوم ۹۹-۹۸

نام و نام خانوادگی: حسن ذاکر، علیرضا دقیق، سپهر فعلی

موعد تحویل: ۹۸/۱۲/۲۵

پارسرها

پاسخ تمرین سری دوم

ما ۳ نفر همه سوالات را با هم حل کردیم  
مسئله‌ی ۱. گرامر مبهم

پاسخ.

$S \rightarrow As \mid \epsilon$   
 $A \rightarrow 01B \mid 0A1$   
 $B \rightarrow 1B \mid \epsilon$

- ابهام ویژگی یک گرامر است نه زبان. ابهام در یک گرامر این قابلیت را ایجاد می‌کند که برای یک ورودی چند درخت پارس بتوان تشکیل داد و ممکن است بر حسب قواعد معنایی جوابی خارج که مورد نظرمان نیست را با پارس کردن درختی دیگر ایجاد کند. گاهی هم ممکن است نتوان پارس را انجام داد.

مسئله‌ی ۲. عبارت آرمانی

پاسخ.

۱. علاوه بر پارس‌استک PS یک استک جدید به نام TS تعریف می‌کنیم که در هنگام پارس کردن به کمک ما می‌آید. به این صورت که هر موقع + داشتیم آن را به TS اضافه می‌کنیم و هرگاه - رسیدیم از استک POP TS می‌کنیم. عملیات پارس به صورت معمول با PS انجام می‌شود و علاوه بر حالت‌های invalid که ممکن است در PS رخ دهد زمانی که باید از POP TS کنیم و TS خالی باشد نیز باید حالت invalid تشخیص داده شود.



۲. تنها با استفاده از گراف نحو و یک استک این کار امکان‌پذیر نیست.

### مسئله‌ی 3. Recursive Descent

پاسخ. چون این گرامر left recursive است قابلیت پارس با Recursive descent را ندارد. برای حل این مشکل ابتدا آنرا رفع ابهام می‌کنیم سپس شبکه‌کد مربوط به آنرا می‌زنیم.

$$E \rightarrow T + E \mid T - E \mid T$$

$$T \rightarrow 11T' \mid 10T'$$

$$T' \rightarrow 10T \mid 11T \mid \epsilon$$

```
1  int main()
2  {
3      // E is a start symbol.
4      E();
5
6      // if lookahead = $, it represents the end of the string
7      // Here l is lookahead.
8      if (l == '$')
9          printf("Parsing Successful");
10 }
11
12 // Definition of E, as per the given production
13 E()
14 {
15     T();
16     if (l == '+') {
17         match('+');
18         E();
19     } else if (l == '-') {
20         match('-');
21         E();
22     }
23     return();
24 }
25
26 // Definition of E' as per the given production
27 T()
28 {
29     if (l == '1') {
30         match('1');
31         if (l == '1') {
32             match('1');
33             T'();
34         } else if (l == '0') {
35             match('0');
36             T'();
37         }
38     }
39
40 }
41
```

```

42 T'()
43 {
44     if (l == '1') {
45         match('1');
46         if(l == '1'){
47             match('1');
48             T();
49         }else if(l == '0'){
50             match('0');
51             T();
52         }
53     }else{
54         return();
55     }
56 }
57
58 // Match function
59 match(char t)
60 {
61     if (l == t) {
62         l = getchar();
63     }
64     else
65         printf("Error");
66 }

```

مسئله‌ی 4. پرانتزگذاری معتبر

پاسخ. بله  $LL(1)$  است.

Follow	First	
) \$	$\epsilon$ (	S
( \$	(	P

\$	)	(	
$\epsilon \rightarrow S$	$\epsilon \rightarrow S$	$PS \rightarrow S$	S
		$(S) \rightarrow P$	P

دو نوع خطا ممکن است در گرامرهای  $LL(1)$  رخ دهد.

- اولی اینکه هنگام پارس به در قسمت پیش‌بینی (سمت چپ) به \$ برسیم ولی هنوز کل رشته را پارس نکرده باشیم. (در سمت راست هنوز رشته برای پارس وجود دارد اما سمت چپ خالی است).  
برای گرامر این سوال می‌توان رشته (( را مثال زد.
- نوع دیگر خطا حالتی است که در یک محصول برای ترمینال بعدی که می‌بینیم در جدول خالی باشد. در این حالت پارس متوقف می‌شود.  
مثال: این حالت برای این سوال اتفاق نمی‌افتد.

## مسئله‌ی 5. با طعم اپسیلون

پاسخ.

$A \rightarrow aB$   
 $B \rightarrow bG$   
 $G \rightarrow cH \mid D \mid \epsilon$   
 $H \rightarrow C \mid d$   
 $C \rightarrow c \mid \epsilon$   
 $D \rightarrow dE$   
 $E \rightarrow cC \mid \epsilon$

۲.

Follow	First	
\$	a	A
\$	b	B
\$	$\epsilon$ d c	G
\$	$\epsilon$ d c	H
\$	$\epsilon$ c	C
\$	d	D
\$	$\epsilon$ c	E

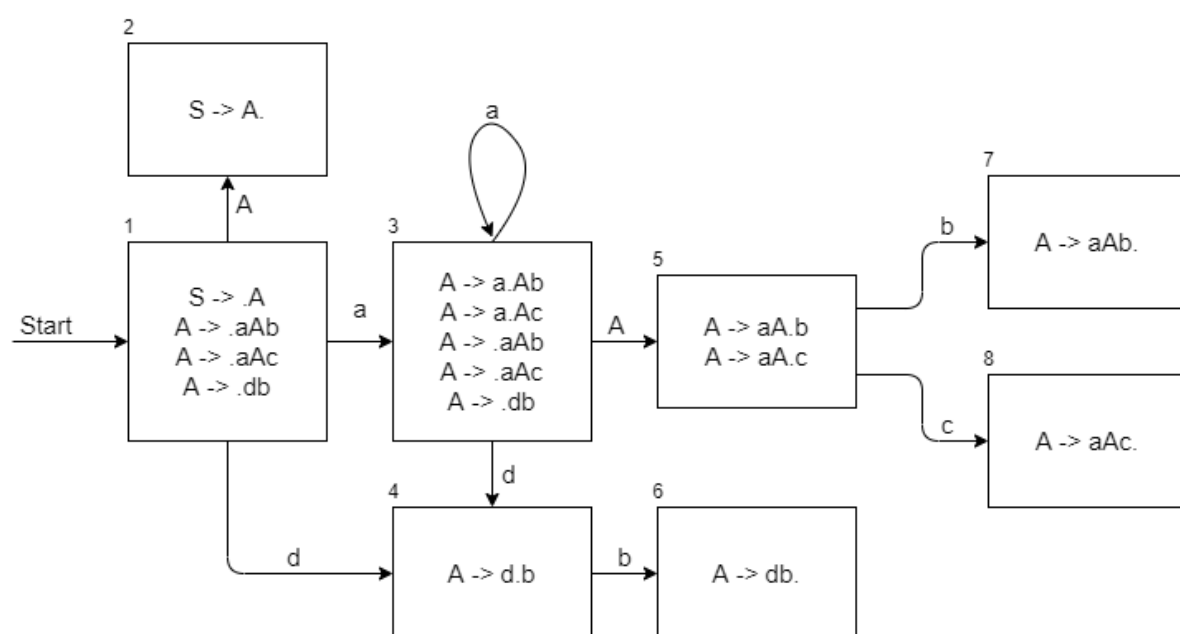
۳.

\$	d	c	b	a	
			1		A
			2		B
5	4	3			G
6	7	6			H
9		8			C
	10				D
12		11			E

abdcc\$	A\$
abdcc\$	aB\$
bdcc\$	B\$
bdcc\$	bG\$
dcc\$	G\$
dcc\$	D\$
dcc\$	dE\$
cc\$	E\$
cc\$	cC\$
c\$	C\$
c\$	c\$
\$	\$

مسئله ۶. پارسر LR(۰)

پاسخ.



A	d	c	b	a	
G2	S4			S3	1
acc	acc	acc	acc	acc	2
G5	S4			S3	3
			S6		4
		S8	S7		5
R4	R4	R4	R4	R4	6
R2	R2	R2	R2	R2	7
R3	R3	R3	R3	R3	8

۲. ابتدا در حالت ۱ قرار داریم.  $(a|adbc)$   
 با خواندن a به حالت ۳ میرویم و توکن بعدی را میخوانیم.  $(aa|dbc)$   
 در این حالت نیز، باز با خواندن a به حالت ۳ میرویم و توکن بعدی را میخوانیم.  $(aad|bc)$   
 حال با خواندن d به حالت ۴ میرویم و توکن بعدی را میخوانیم.  $(aadb|c)$   
 حال با خواندن b به حالت ۶ میرویم که در این حالت، reduction داریم به صورت  $db \rightarrow A$  پس ۲ حالت به عقب میرویم (حالت ۳) و db را نیز به A تبدیل میکنیم.  $(aaA|c)$   
 حال در حالت ۳ هستیم و با خواندن A به حالت ۵ میرویم. توکن بعدی را میخوانیم.  $(aaAc|)$   
 حال با خواندن c به حالت ۸ میرویم که در این حالت، reduction داریم به صورت  $aAc \rightarrow A$  پس ۳ حالت به عقب میرویم (حالت ۳) و aAc را نیز به A تبدیل میکنیم.  $(aA|)$   
 حال در حالت ۳ هستیم و با خواندن A به حالت ۵ میرویم. و همین جا گیر میفتیم. چرا که aA را داریم و دیگر هیچ.  
 پس نمیتوان این رشته را پارس کرد.

## مسئله ۷. LALR یا SLR!

پاسخ. اثبات به برهان خلف. فرض خلف:  $slr(1)$  بدون conflict داریم اما مق(۱) بدون conflict نداریم. از آنجایی که در  $lalr(1)$  برخورد داشته ایم پس یک terminal وجود دارد که در آنجا هم می‌توانیم کاهش دهیم هم شیفت دهیم. پس یک قاعده تولید وجود دارد که terminal ذکر شده در بالا در سمت راست فرضی nonterminal وجود دارد پس این nonterminal در فالو خود terminal ذکر شده را دارد. پس در  $slr(1)$  هم داریم conflict تناقض.  
 حال می‌خواهیم مثالی بزنیم که در  $SLR(1)$  برخورد دارد ولی در  $Lalr(1)$  ندارد

$E \rightarrow S$

$E \rightarrow a + b + S + c$

$S \rightarrow A + B$

$B \rightarrow B + c$

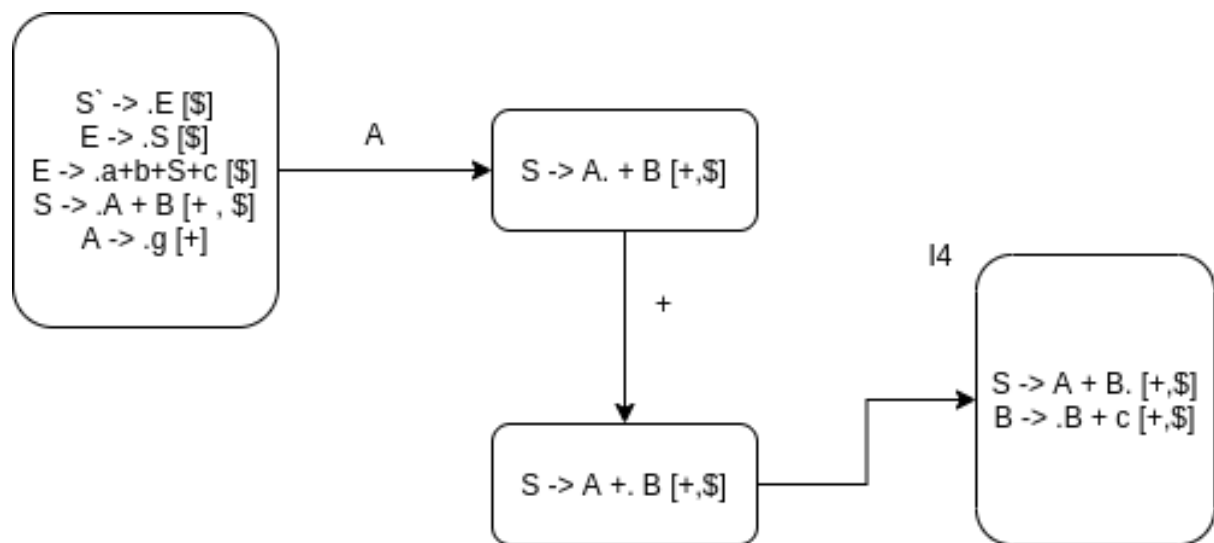
$A \rightarrow g$

Follow E = \$

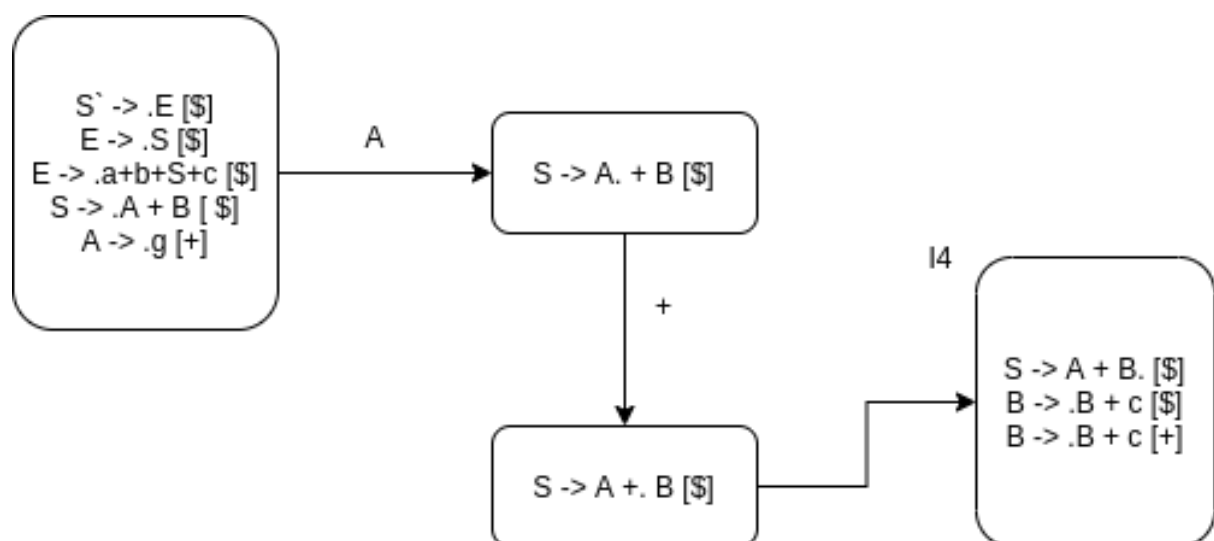
Follow S = \$ , +

Follow B = + , \$  
Follow A = not important

ابتدا نمودار SLR1 را می کشیم و نشان می دهیم در آن برخورد داریم



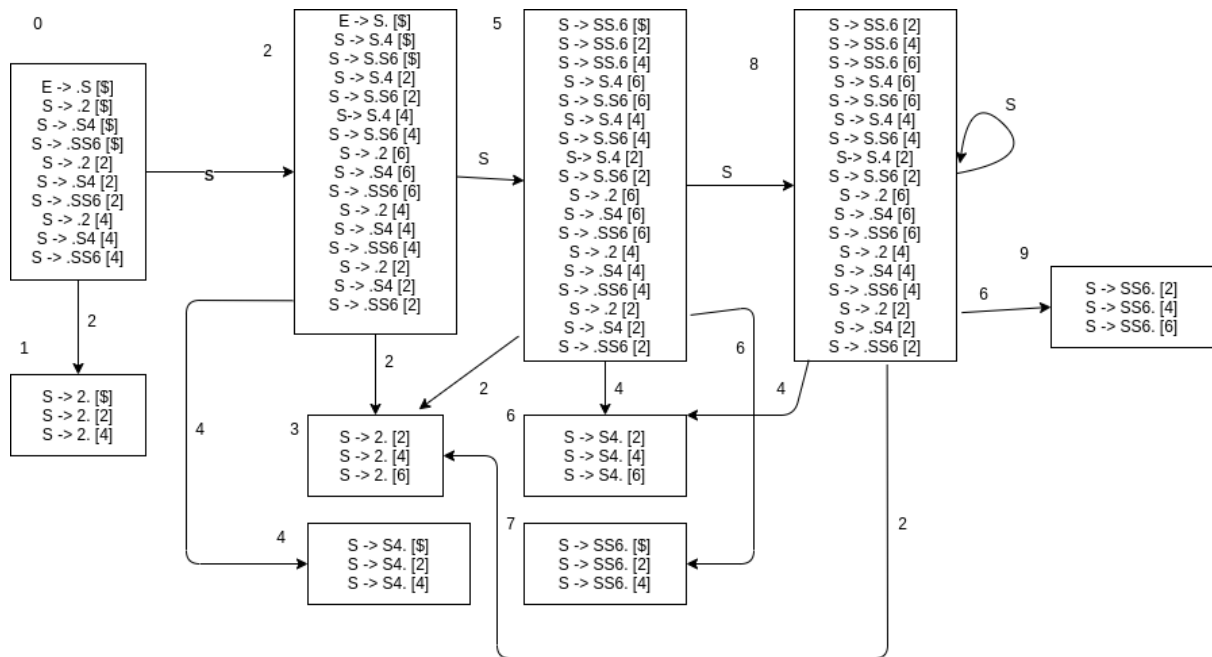
همانطور که مشاهده می کنید در شماره state ۴ هم می توانیم در + عمل کاهش را انجام دهیم چون در فالو S آمده هم می توانیم با + عمل شیفت را انجام دهیم و از آن خارج شویم حال دیاگرام را ۱۱ رسم می کنیم.



مشاهده می کنیم که در state شماره ۴ در \$ عمل کاهش را انجام می دهیم و با + می توانیم خارج شویم در نتیجه برخورد ندارند

مسئله ۸. پارسر LR(۱)

پاسخ.



S	&	۶	۴	۲	
G۲				S۱	۰
	R۱		R۱	R۱	۱
G۵	acc		S۴	S۳	۲
		R۱	R۱	R۱	۳
	R۲		R۲	R۲	۴
G۸		S۷	S۶	S۳	۵
		R۲	R۲	R۲	۶
	R۳		R۳	R۳	۷
G۸		S۹	S۶	S۳	۸
		R۳	R۳	R۳	۹

ابتدا در state شماره ۰ هستیم. ۲ را می بینیم و به state شماره ۱ می رویم. سپس دوی بعدی را به عنوان lookahead انتخاب می کنیم. در ۱ به ازای a look ahead ۲ عمل reduce با قاعده تولید شماره یک را داریم. پس به ۲ و ۱ را خط می زنیم و S را جایگزین می کنیم. همین روند را تا آخر به صورت زیر ادامه می دهیم

0 -> 2 -> S1 , look a head = 2

0 -> S -> G2 -> 2 -> S3 , look a head = 6

0 -> S -> G2 -> S -> G5 -> 6 -> S7 , look a head = 4

0 -> S -> G2 -> 4 -> S4 , look ahead = 2

0 -> S -> G2 -> 2 -> S3 , look a head = 2

0 -> S -> G2 -> S -> G5 -> 2 -> S3 , look a head = 4

0 -> S -> G2 -> S -> G5 -> S -> G8 -> 4 -> S6 , look a head = 6

0 -> S -> G2 -> S -> G5 -> S -> G8 -> 6 -> S9 , look a head = 6

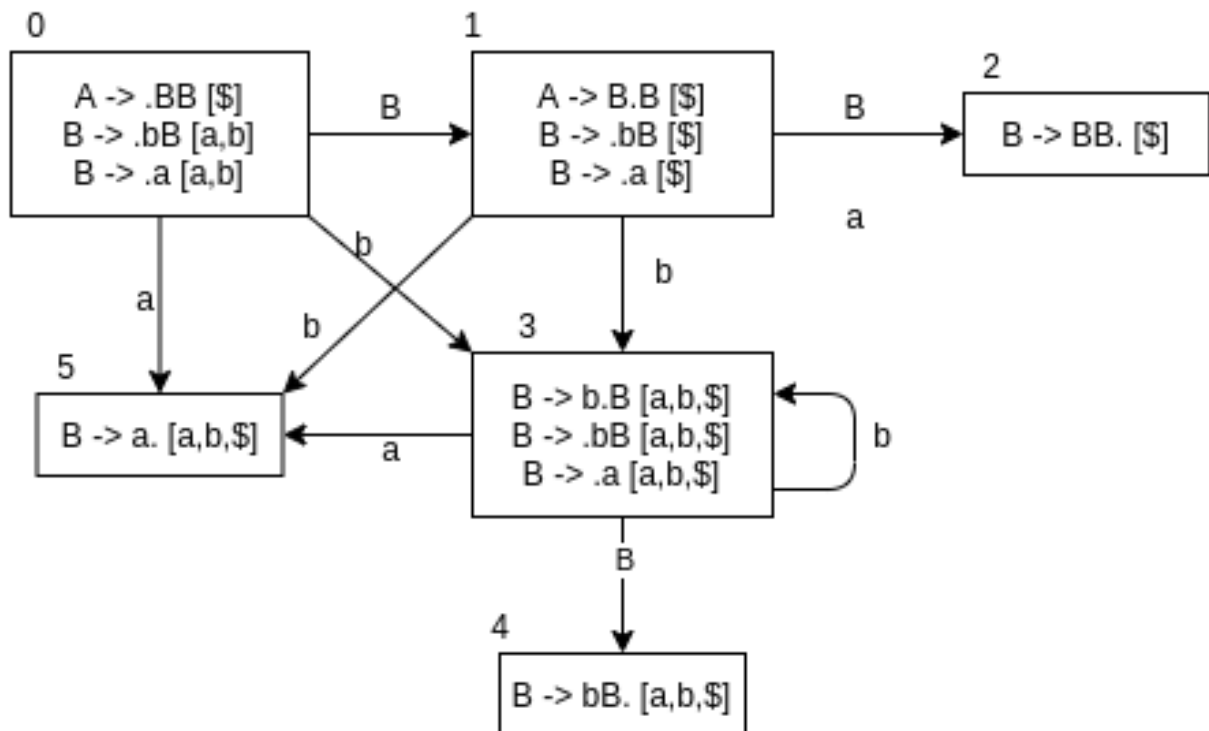
0 -> S -> G2 -> S -> G5 -> 6 -> S7 , look a head = \$



0 -> S -> G2 -> \$ -> Accept

مسئله 9. پارسر LALR(1)

پاسخ.



۱.

B	\$	b	a	
1		S3	S5	0
2		S3	S5	1
	R1			2
4		S3	S5	3
	R2	R2	R2	4
	R3	R3	R3	5

می‌خواهیم رشته babba را پارس کنیم. مشابه سوال قبل عمل می‌کنیم.

0 -> b -> 3 -> a -> 5, look a head = b

0 -> b -> 3 -> B -> 4, look a head = b

0 -> B -> 1 -> b -> 3 -> b -> 3 -> a -> 5, look a head = \$

0 -> B -> 1 -> b -> 3 -> b -> 3 -> B -> 4, look a head = \$

0 -> B -> 1 -> b -> 3 -> B -> 4 , look a head = \$

0 -> B -> 1 -> B -> 2 , look a head = \$

0 -> A -> \$ -> acc

## مسئله 10. مقایسه LR(1) و SLR(1)

پاسخ.

۱. نادرست چون در  $slr(1)$  کاهش را در مجموعه follow انجام می دهد در صورتی که در  $Lr(1)$  در lookahead که عملا زیر مجموعه فالو است انجام می دهد پس تعداد برخوردها کم می شود و با احتمال بیشتری درخت پارس را می سازد

۲. هر دو دیاگرام، به شکل زیر خواهند بود و به دلیل رخ دادن SR Conflict در حالت ۳، نه  $SLR(1)$  است و نه  $LR(1)$ .

