تکلیف دوم

مجتبی ملائی
۴۰۱۳۱۳۸۳

۱

آ) با بررسی در S می‌بینیم که $S \stackrel{+}{\Rightarrow} S\alpha$ وجود دارد. ابتدا در $S \to AbS$ قانون A را با سمت راست آن جایگزین می‌کنیم:

$$
\begin{aligned}
&S \to SaS \mid SaAbS \mid BbS \\
&A \to SaA \mid B \\
&B \to bS \mid c
\end{aligned}
$$

سپس طبق قانون گفته شده در اسلاید داریم:

$$
\begin{aligned}
&S \to BbSS' \\
&S' \to aSS' \mid aAbSS' \mid \epsilon \\
&A \to SaA \mid B \\
&B \to bS \mid c
\end{aligned}
$$

ب) برای ساده سازی ابتدا B و C را جایگزین می‌کنیم.

$$
\begin{aligned}
&S \to abcA \mid abcb \mid abc \\
&A \to abA \mid abbA \mid abbc
\end{aligned}
$$

سپس abc را در S فاکتور می‌گیریم.

$$
\begin{aligned}
&S \to abcS' \\
&S' \to A \mid b \mid \epsilon \\
&A \to abA \mid abbA \mid abbc
\end{aligned}
$$

در A حروف ab را فاکتور می‌گیریم.

$$
\begin{aligned}
&S \to abcS' \\
&S' \to A \mid b \mid \epsilon \\
&A \to abA' \\
&A' \to A \mid bA \mid bc
\end{aligned}
$$

حال می‌توانیم دوباره b را فاکتور بگیریم.

$$
\begin{aligned}
&S \to abcS' \\
&S' \to A \mid b \mid \epsilon \\
&A \to abA' \\
&A' \to A \mid bA'' \\
&A'' \to A \mid c
\end{aligned}
$$

۲

Non-terminals	First	Follow
Program	{	$
Statements	id , if , ϵ	}
Statement	id , if	id , if , }
Expression	id	; ,)
Tail	+ , - , ϵ	; ,)

Table 1: First and Follow sets for non-terminals

```
// Assume we have a function `nextToken()` that gets the next token from the input stream
// Assume `currentToken` holds the current token
// Assume `match(expected)` matches the current token and advances to the next one
function Program():
  if currentToken == '{':
    match('{')
    Statements()
    match('}')
    match('eof') // Ensure the program ends correctly
  else:
    error("Expected '{' at the start of the program")

function Statements():
  if currentToken in {'id', 'if'}: // FIRST(Statements)
    Statement()
    Statements()
    else:
      // epsilon (FOLLOW(Statements) is { '}' }), so we return without consuming anything
function Statement():
  if currentToken == 'id':
    match('id')
    match('=')
    Expression()
    match(';')
  else if currentToken == 'if':
    match('if')
    match('(')
    Expression()
    match(')')
    Statement()
  else:
    error("Invalid statement")

function Expression():
  if currentToken == 'id':
    match('id')
    Tail()
  else:
    error("Expected identifier in expression")

function Tail():
  if currentToken == '+':
    match('+')
    Expression()
  else if currentToken == '-':
    match('-')
    Expression()
  else:
    // epsilon (FOLLOW(Tail) is { ';' , ')' }), so we return without consuming anything
```

Listing 1: Recursive Descent Parser Pseudo-Code

۳

۱.

Non-terminals	First	Follow
S	if	$
I	=, ϵ	$
E	(, id, num	$,), then
E’	+ , ϵ	$,), then
T	(, id, num	+, $,), then
T’	*, ϵ	+, $,), then
F	(, id, num	*, +, $,), then

۲.

	id	num	$($	$)$	$+$	$*$	$=$	if	$then$	$
S	idI							$ifEthenS$		
I										e
E	TE'	TE'	TE'							
E'				ϵ	$+TE'$				ϵ	ϵ
T	FT'	FT'	FT'							
T'				ϵ	ϵ	$*FT'$			ϵ	ϵ
F	id	num	(E)							

۳.

Step	Matched	Stack	Input	Action
1	ϵ	S	$if\ id\ then\ id\ =\ (\ num\ *\ id\ )\ +\ num\ \$$	
2	ϵ	$if\ E\ then\ S$	$if\ id\ then\ id\ =\ (\ num\ *\ id\ )\ +\ num\ \$$	output $S \to if\ E\ then\ S$
3	if	$E\ then\ S$	$id\ then\ id\ =\ (\ num\ *\ id\ )\ +\ num\ \$$	match if
4	if	$T\ E'\ then\ S$	$id\ then\ id\ =\ (\ num\ *\ id\ )\ +\ num\ \$$	output $E \to T\ E'$
5	if	$F\ T'\ E'\ then\ S$	$id\ then\ id\ =\ (\ num\ *\ id\ )\ +\ num\ \$$	output $T \to F\ T'$
6	if	$id\ T'\ E'\ then\ S$	$id\ then\ id\ =\ (\ num\ *\ id\ )\ +\ num\ \$$	output $F \to id$
7	$if\ id$	$T'\ E'\ then\ S$	$then\ id\ =\ (\ num\ *\ id\ )\ +\ num\ \$$	match id
8	$if\ id$	$E'\ then\ S$	$then\ id\ =\ (\ num\ *\ id\ )\ +\ num\ \$$	output $T' \to \epsilon$
9	$if\ id$	$then\ S$	$then\ id\ =\ (\ num\ *\ id\ )\ +\ num\ \$$	output $E' \to \epsilon$
10	$if\ id\ then$	S	$id\ =\ (\ num\ *\ id\ )\ +\ num\ \$$	match $then$
11	$if\ id\ then$	$id\ I$	$id\ =\ (\ num\ *\ id\ )\ +\ num\ \$$	output $S \to id\ I$
12	$if\ id\ then\ id$	I	$=\ (\ num\ *\ id\ )\ +\ num\ \$$	match id
13	$if\ id\ then\ id$	$=\ E$	$=\ (\ num\ *\ id\ )\ +\ num\ \$$	output $I \to =\ E$
14	$if\ id\ then\ id\ =$	E	$(\ num\ *\ id\ )\ +\ num\ \$$	match $=$
15	$if\ id\ then\ id\ =$	$T\ E'$	$(\ num\ *\ id\ )\ +\ num\ \$$	output $E \to T\ E'$
16	$if\ id\ then\ id\ =$	$F\ T'\ E'$	$(\ num\ *\ id\ )\ +\ num\ \$$	output $T \to F\ T'$
17	$if\ id\ then\ id\ =$	$(\ E\ )\ T'\ E'$	$(\ num\ *\ id\ )\ +\ num\ \$$	output $F \to (E)$
18	$if\ id\ then\ id\ =($	$E\ )\ T'\ E'$	$num\ *\ id\ )\ +\ num\ \$$	match (
19	$if\ id\ then\ id\ =($	$T\ E'\ )\ T'\ E'$	$num\ *\ id\ )\ +\ num\ \$$	output $E \to T\ E'$
20	$if\ id\ then\ id\ =($	$F\ T'\ E'\ )\ T'\ E'$	$num\ *\ id\ )\ +\ num\ \$$	output $T \to F\ T'$
21	$if\ id\ then\ id\ =($	$num\ T'\ E'\ )\ T'\ E'$	$num\ *\ id\ )\ +\ num\ \$$	output $F \to num$
22	$if\ id\ then\ id\ =(\ num$	$T'\ E'\ )\ T'\ E'$	$*\ id\ )\ +\ num\ \$$	match num
23	$if\ id\ then\ id\ =(\ num$	$*\ F\ T'\ E'\ )\ T'\ E'$	$id\ )\ +\ num\ \$$	output $T' \to *\ F\ T'$
24	$if\ id\ then\ id\ =(\ num\ *$	$F\ T'\ E'\ )\ T'\ E'$	$id\ )\ +\ num\ \$$	match $*$
25	$if\ id\ then\ id\ =(\ num\ *$	$id\ T'\ E'\ )\ T'\ E'$	$id\ )\ +\ num\ \$$	output $F \to id$
26	$if\ id\ then\ id\ =(\ num\ *\ id$	$T'\ E'\ )\ T'\ E'$	$)\ +\ num\ \$$	match id
27	$if\ id\ then\ id\ =(\ num\ *\ id$	$E'\ )\ T'\ E'$	$)\ +\ num\ \$$	output $T' \to \epsilon$
28	$if\ id\ then\ id\ =(\ num\ *\ id$	$)\ T'\ E'$	$)\ +\ num\ \$$	output $E' \to \epsilon$
29	$if\ id\ then\ id\ =(\ num\ *\ id\ )$	$T'\ E'$	$+\ num\ \$$	match)
30	$if\ id\ then\ id\ =(\ num\ *\ id\ )$	$+\ T\ E'$	$num\ \$$	output $E' \to +TE'$
31	$if\ id\ then\ id\ =(\ num\ *\ id\ )+$	$T\ E'$	$num\ \$$	match $+$
32	$if\ id\ then\ id\ =(\ num\ *\ id\ )+$	$F\ T'\ E'$	$num\ \$$	output $T \to FT'$
33	$if\ id\ then\ id\ =(\ num\ *\ id\ )+$	$num\ T'\ E'$	$num\ \$$	output $F \to num$
34	$if\ id\ then\ id\ =(\ num\ *\ id\ )\ +\ num$	$T'\ E'$	$\$$	match num
35	$if\ id\ then\ id\ =(\ num\ *\ id\ )\ +\ num$	E'	$\$$	output $T' \to \epsilon$
36	$if\ id\ then\ id\ =(\ num\ *\ id\ )\ +\ num$		$\$$	output $E' \to \epsilon$
37	$if\ id\ then\ id\ =(\ num\ *\ id\ )\ +\ num\ \$$			match $

Table 2: LL(1) Parsing Trace for `if id then id = ( num * id ) + num`

۴

۱. ابتدا فاکتور گیری چپ انجام می‌دهیم.

$S \to iEtSA|a$
$A \to \epsilon|eS$
$E \to B$

سپس گرامر را به فرم مورد نیاز تبدیل می‌کنیم

$0 : S' \to S$
$1 : S \to iEtSA$
$2 : S \to a$
$3 : A \to eS$
$4 : A \to \epsilon$
$5 : E \to b$

I_0 :
$S' \to .S\$$
$S \to .iEtSA$
$S \to .a$

I_1 :
$S' \to S.\$$

I_2 :
$S \to a.$

I_3 :
$S \to i.EtSA$
$E \to .b$

I_4 :
$E \to b.$

I_5 :
$S \to iE.tSA$

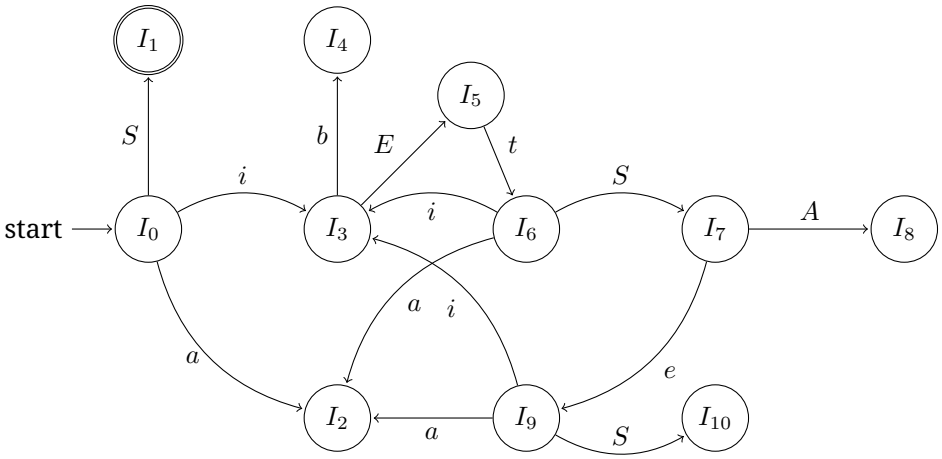
I_6 :
$S \to .iEt.SA$
$S \to .iEtSA$
$S \to .a$

I_7 :
$S \to iEtS.A$
$A \to .$
$A \to eS$

I_8 :
$S \to iEtSA.$

I_9 :
$A \to e.S$
$S \to .iEtSA$
$S \to .a$

I_{10} :
$A \to eS.$



در I_1 با $ به اکسپت می‌رویم.

۲.

	Action						**GOTO**		
state	a	b	e	i	t	$	A	E	S
0	s2			s3					1
1						acc			
2	r2								
3		s4						2	
4	r5								
5					s5				
6	s2			s3					7
7	r4		r4 & s9		r4		8		
8	r1								
9	s2			s3					10
10	r3								

action های خالی به معنی error هستند!

در حالت ۷ برای $\texttt{Action}[7,e]$ هم reduce داریم و هم shift. بنابراین گرامر LR(0) نیست.

۵

۱. نیاز داریم گرامر را به شکل زیر تبدیل دهیم.

$0 : S' \to S$
$1 : S \to i$
$2 : S \to SS+$
$3 : S \to SS*$

	ACTION				GOTO	
state	$+$	$*$	i	$\$$	S	**Explanation**
0			s1		2	Initial state; shift on ‘i‘ to state 1, go to 2 on ‘S‘
1	r1					Reduce using rule 1: $S \to i$
2			s1	acc	3	After first ‘S‘; shift ‘i‘ to 1 or accept if input ends
3	s4	s5	s1		3	After ‘S‘; can shift ‘+‘, ‘*‘, or another ‘i‘
4	r2					Reduce using rule 2: $S \to SS+$
5	r3					Reduce using rule 3: $S \to SS*$

۲.

step	stack	input	handle	action
0	$\$$	$iii*i+*$		shift
1	$\$i$	$ii*i+*\$$	i	reduce 1
2	$\$S$	$ii*i+*\$$		shift
3	$\$Si$	$i*i+*\$$	i	reduce 1
4	$\$SS$	$i*i+*\$$		shift
5	$\$SSi$	$*i+*\$$	i	reduce 1
6	$\$SSS$	$*i+*\$$		shift
7	$\$SSS*$	$i+*\$$	$SS*$	reduce 3
8	$\$SS$	$i+*\$$		shift
9	$\$SSi$	$+*\$$	i	reduce 1
10	$\$SSS$	$+*\$$		shift
11	$\$SSS+$	$*\$$	$SS+$	reduce 2
12	$\$SS$	$*\$$		shift
13	$\$SS*$	$\$$	$SS*$	reduce 3
14	$\$S$	$\$$		acc

۶

گرامر را به شکل زیر تبدیل می‌کنیم.

$0 : S' \to S$
$1 : S \to XdY$
$2 : X \to aX$
$3 : X \to \epsilon$
$4 : Y \to bYS$
$5 : Y \to \epsilon$

حال در I_0 داریم:

$0 : S' \to .S$
$1 : S \to .XdY$
$2 : X \to .aX$
$3 : X \to .$

در این حالت چون نقطه به احر رسیده است یک reduce مربوط به $3 : X \to \epsilon$ همچنین یک شیفت داریم $2 : X \to .aX$ بنابراین اگر جدول را بکشیم، در یکی از خانه ها collision رخ خواهد داد پس LR(0) نیست.