Questions & Answers 001 for compiler course

Teacher: Lin Yi , School of Computer Science, Northwestern Polytechnical University
E-mail: ly_cs@nwpu.edu.cn
Mobile Phone: 13572591128

Date: May. 15, 2022

Student Name : ABID ALI
Student ID : 201930141

Content

Question & Answer

【Q1】left-most derivation, syntax tree

【Q1】Given G[E]:

E->TE'　　E'->ATE' | ε　　T->FT'

T'->MFT' | ε　　F->(E) | i

A->+ | -　　M->* | /

① Give the left-most derivation sequence for i+i*i

② Compute the First Set, Follow Set and LL(1) Table.

Then give the LL(1) table based left-most derivation for (i+i)*i

③ From the steps of above derivation from②,

write the syntax tree that could represent (i + i) MFT'E'

Answer:

（1）left most derivation for "i+i*i"。

(1) Left-most derivation "i+i*i"

$E \Rightarrow TE' \Rightarrow FT'E' \Rightarrow iT'E' \Rightarrow iE'$ [$\because T' \Rightarrow \varepsilon$]

$\Rightarrow iATE'$

$\Rightarrow i+TE'$

$\Rightarrow i+FT'E'$

$\Rightarrow i+iT'E'$

$\Rightarrow i+iMFT'E'$

$\Rightarrow i+i*FT'E'$

$\Rightarrow i+i*iT'E'$

$\Rightarrow i+i*iE'$　　$(T' \Rightarrow \varepsilon)$

$\Rightarrow i+i*i$　　$(E' \Rightarrow \varepsilon)$

（2）left most derivation for “(i+i)*i”

(2) Left-most derivation for "(i+i)*i"

$E \Rightarrow TE' \Rightarrow FT'E' \Rightarrow (E)T'E' \Rightarrow (TE')T'E'$

$\Rightarrow (FT'E')T'E'$

$\Rightarrow (iT'E')T'E'$

$\Rightarrow (iE')T'E' \quad [\because T' \rightarrow \varepsilon]$

$\Rightarrow (iT'E')T'E'$

$\Rightarrow (i\varepsilon E')T'E'$

$\Rightarrow (iATE')T'E'$

$\Rightarrow (i+TE')T'E'$

$\Rightarrow (i+FT'E')T'E'$

$\Rightarrow (i+iT'E')T'E'$

$\Rightarrow (i+i\ E')T'E' \quad [\because T' \rightarrow \varepsilon]$

$\Rightarrow (i+i)T'E' \quad [\because E' \rightarrow \varepsilon]$

$\Rightarrow (i+i)\ MFT'E'$

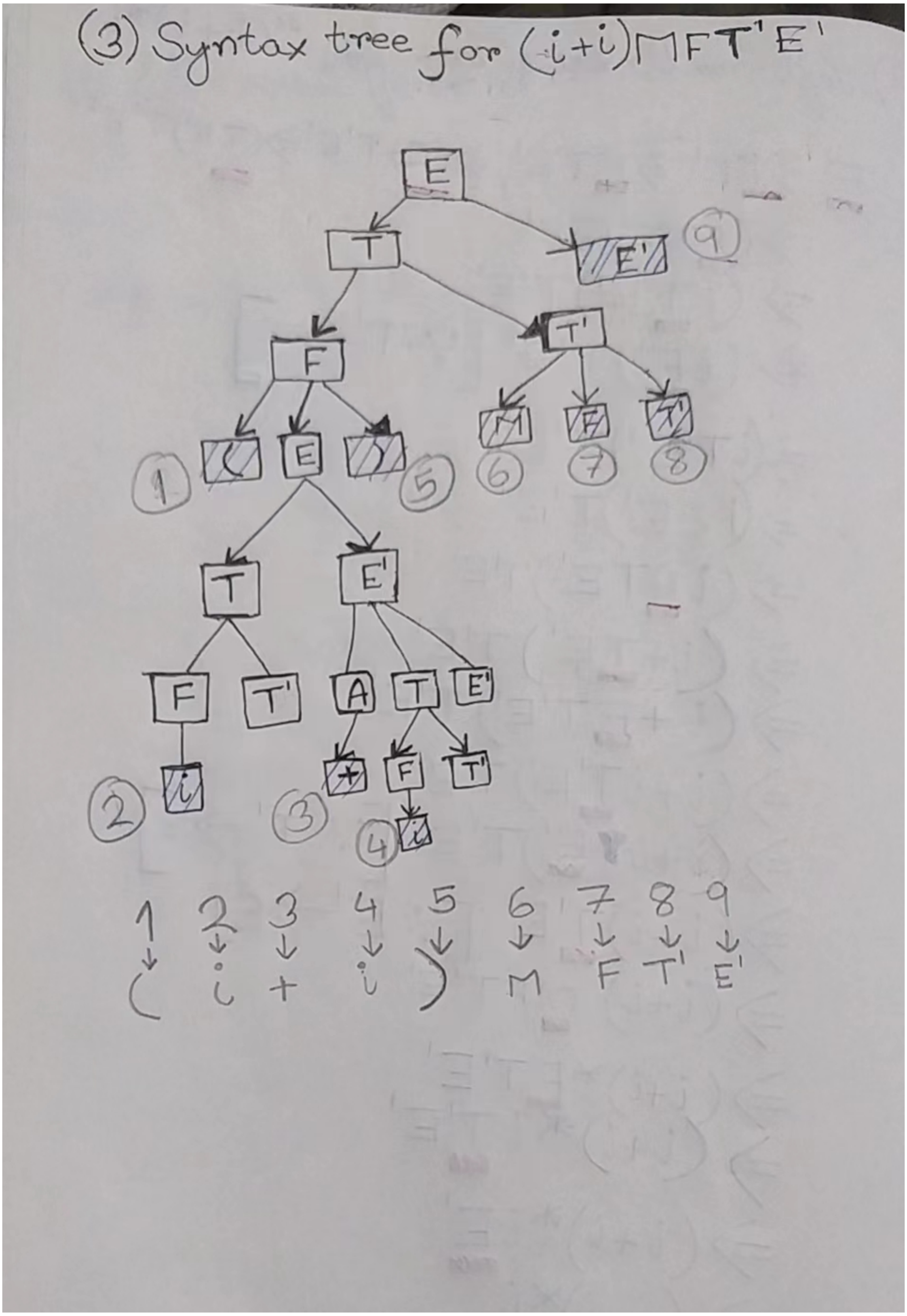
$\Rightarrow (i+i)*FT'E'$

$\Rightarrow (i+i)*iT'E'$

$\Rightarrow (i+i)*iE'$

$\Rightarrow (i+i)*i$

（3）syntax tree for　 (i + i) MFT’E’



【Q 2】Ambiguous / Syntax Tree, etc.

【Q2】Given G[E]

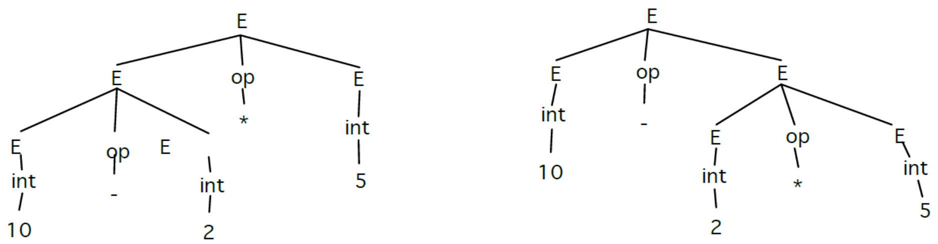
E -> E op E | (E) | int

op -> + | - | * | /

int -> 0 | 1 | ... | 9

①Judge whether G[E] is ambiguous. Give two different reasons.

② For string 10-2*5, two different syntax tree could be left-most derived.



Give your computation sequence for the two trees.

Answer:

（1）It is ambiguous.

Ans No: 2

(1) It's ambiguous

Reason 1: Both left recursive & right recursive are defined in the BNF rules.

Reason 2: For 2+3*4, there will be at least two different syntax tree.

Reason 3: For 2+3*4, there will be at least two different left most derivation.

Reason 4: For 2+3*4, there will be at least two different right-most derivation.

(2)

left tree ⇒ (10-2)*5

right tree ⇒ 10-(2*5)

【Q3】Write BNF syntax rules from given language

【Q3】Given language as { $a^n b^n c^m$ | n>=1， m>=0}, write BNF syntax rules.

Answer:

Ans No: 3

G[S], S is the start symbol

$S \rightarrow AB$

$A \rightarrow aAb \mid ab$ (Since n>=1, so $A \rightarrow \varepsilon$ is not included in the syntax)

$B \rightarrow cb \mid \varepsilon$ (another BNF is equal: $B \rightarrow Bc \mid \varepsilon$)

Checking

Minimum length string is ab | abc

Set of possible strings are a^2b^2c, $a^3b^3c^2$

Case 1

String ab

$S \rightarrow \underline{A}B$

$\rightarrow ab\underline{B}$

$\rightarrow ab\underline{\varepsilon}$

$\rightarrow ab$

Case 2

String abc

$S \rightarrow \underline{A}B$

$\rightarrow ab\underline{B}$

$\rightarrow abc\underline{B}$

$\rightarrow abc\underline{\varepsilon}$

$\rightarrow abc$

Case 3

String a^2b^2c

$S \rightarrow \underline{A}B$

$\rightarrow a\underline{A}bB$

$\rightarrow aa\ bb\underline{B}$

$\rightarrow aabbc\underline{B}$

$\rightarrow a^2b^2c\underline{B}$

$\rightarrow a^2b^2c\underline{\varepsilon}$

$\rightarrow a^2b^2c$

【Q4】Write BNF from language

【Q4】Suppose a language contains all positive even integers (0, 2, 4, 6, 8, ...)

① Write BNF for the above language supposing that the number could begin with number 0 ---- e.g: 03, 09 is allowed.

② If all number could not begin with 0. Write BNF.

Answer:

（1）

Ans No: 4

(1) Positive even number is 2, 4, 6, 8, 0

Define as symbol E

$E \rightarrow 2 \mid 4 \mid 6 \mid 8 \mid 0$

① $S \rightarrow E$

$D \rightarrow 0 \mid 1 \mid 2 \mid \ldots \mid 9$

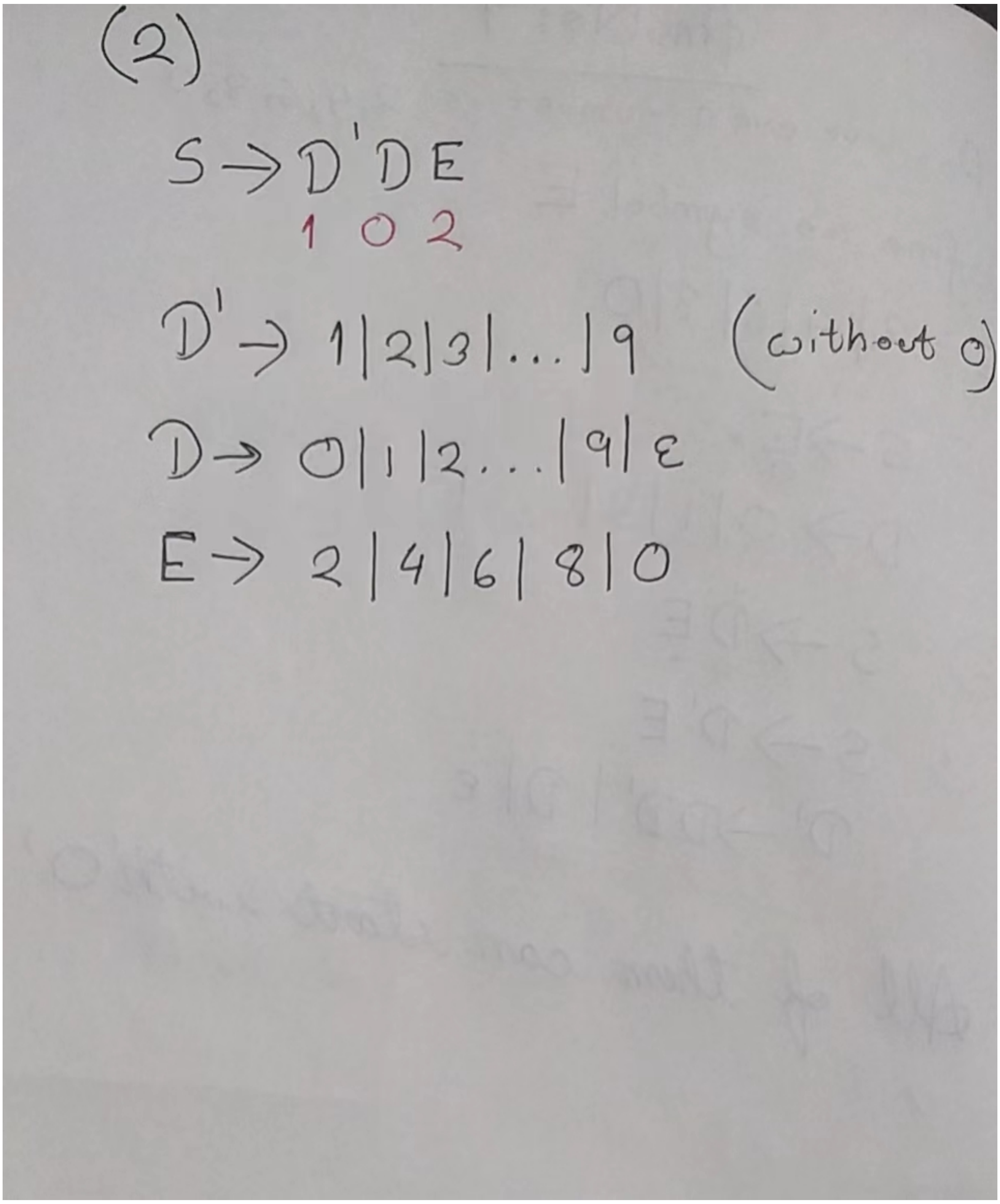
② $S \rightarrow DE$

③ $S \rightarrow D'E$

$D' \rightarrow DD' \mid D \mid \varepsilon$

All of them can start with '0'

（2）



(2)

$S \rightarrow D'DE$

1 0 2

$D' \rightarrow 1|2|3|\ldots|9$ (without 0)

$D \rightarrow 0|1|2\ldots|9|\varepsilon$

$E \rightarrow 2|4|6|8|0$

【Q5】Eliminate left recursive and left factor

【Q5】Given G[A]：

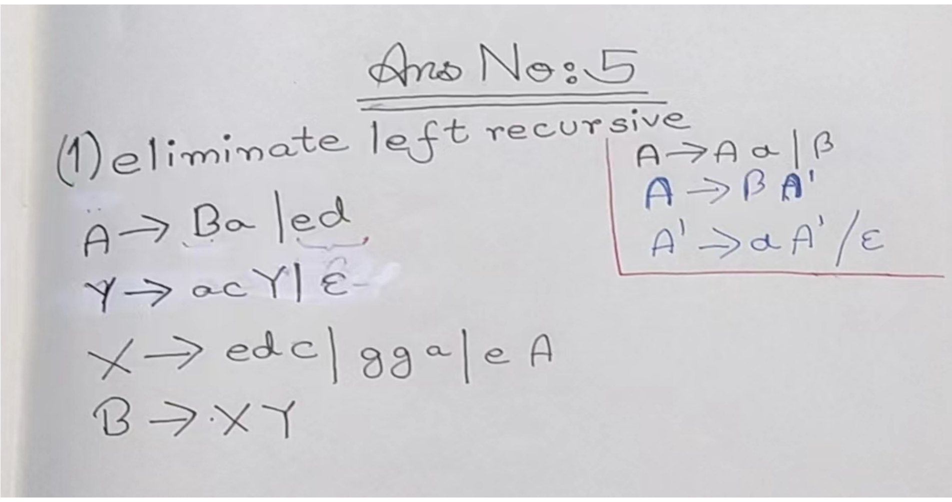
A->Ba | ed

B->Bac | edc | ggA |eA

Please eliminate direct left recursive and left factor.

Answer:

（1）eliminate left recursive



Ans No:5

(1) eliminate left recursive

$A \to Ba \mid ed$

$Y \to acY \mid \varepsilon$

$X \to edc \mid gga \mid eA$

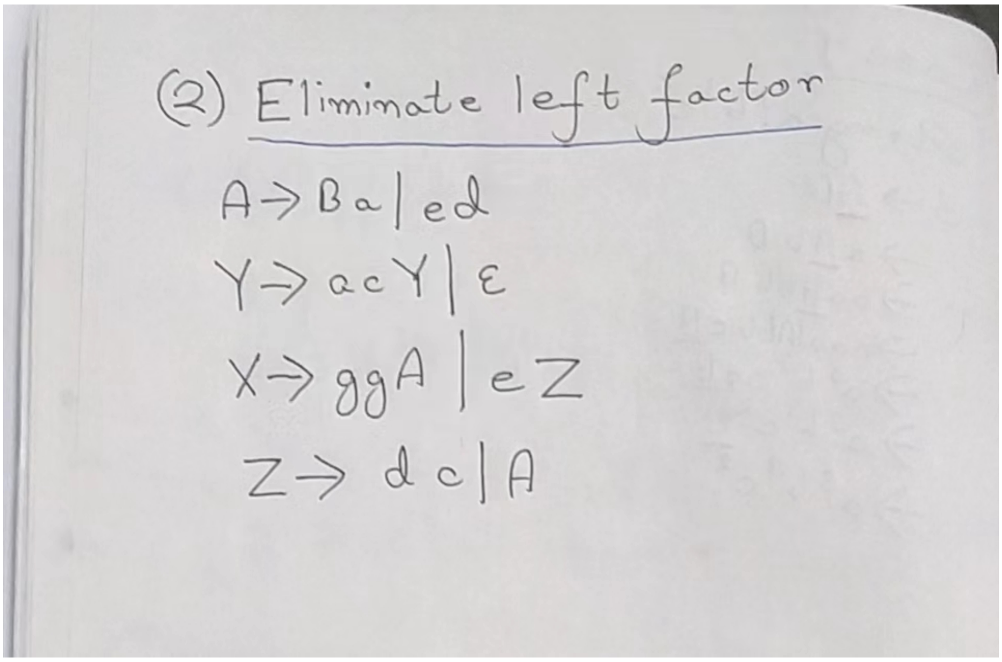
$B \to XY$

$A \to A\alpha \mid \beta$

$A \to \beta A'$

$A' \to \alpha A' / \varepsilon$

（2）



(2) Eliminate left factor

$A \to Ba \mid ed$

$Y \to acY \mid \varepsilon$

$X \to ggA \mid eZ$

$Z \to dc \mid A$

【Q6】LL(1) parsing table，First/Follow set

【Q6】Given G[S]:

S→XY

X→Ya｜ε

Y→Zb｜Z

Z→d｜ε

Calculate First set, Follow set and LL(1) parsing table. Then judge whether the syntax is or is not LL(1) grammar.

Answer:

（1）First and Follow

	First	Explain	Follow	explain
S->XY	ε,a,b,d,	First(S)=First(XY) （1）ε： XY=>εY=>εZ=>εε （2）a： XY=>YaY=>ZaY =>εaY=>aY （3）b： XY=>εY=>εZb=>εεb Or XY=>YaY=>ZbaY=>εbaY=>baY （4）d： XY=>YaY=>ZbaY=>dbaY Or XY=>YaY=>ZaY=>daY Or XY=>εY=>εZb=>εdb Or XY=>εY=>εZ=>εd	#	S#
X->Ya	a,b,d	First(Ya) （1）a： Ya=>εa （2）b： Ya=>Zba=>εba （3）d： Ya=>Zba=>dba Or Ya=>Za=>da	#,b,d	X only appear in S->XY 【Thinking path】S#=>XY# （1）#： XY#=>XZ#=>Xε#=>X# （2）b： XY#=>XZb# =>Xεb#=>Xb# （3）d： XY#=>XZb#=>Xdb# Or XY#=>XZ#=>Xd#
X->ε	ε			
Y->Zb	b,d	First(Zb) （1）b： Zb=>εb （2）d： Zb=>db	#,a	Y only appear in S->XY and X->Ya （1）#： S#=>XY# （2）a： S=>XY#=>YaY#
Y->Z	ε,d	First(Z)		
Z->d	d		#,a,b	Z only appear in：Y->Zb and Y->Z。Then consider scenario that Y appears. To cover all scenarios, each time Y appeared, the two appearances of Z must be considered. （1）#： S#=>XY#=>XZ# （2）a： X=>Ya=>Za （3）b： Y=>Zb
Z->ε	ε			

（2）LL(1) parsing table----cells with color contain conflicts

	A	b	d	#
S	S->XY（according to First(XY)）	S->XY（First(XY)）	S->XY（ according to First(XY)）	S->XY （ Folow(S). Reason: S=>XY=>ε）
X	X->Ya	**X->Ya（First(Ya)）** **X->ε（Follow(X)）**	**X->Ya（First(Ya)）** **X->ε（ Follow(X)）**	X->ε（Follow(X)）
Y	Y->Z（Follow）	Y→Zb（First(Zb)）	**Y → Zb （ according to First(Zb)）** **Y->Z（according to Follow）**	Y->Z（Follow）
Z	Z->ε（Follow）	Z->ε（ Follow）	Z->d（according to First(d)）	Z->ε（Follow）

（3）There are conflicts. It is not LL(1) grammar.

【Q7】LL(1) and ambiguous

【Q7】Given G[S]：

S->if e then S | if e then S else S

Eliminate left recursive and get G'[S]：

S->AB

A->if e then S

B->else S | ε

（1）Write the First set Follow Set and LL(1) table for G'[S].

（2）Judge if it is LL(1).

（3）Is the grammar ambiguous?

Answer:

（1）G'[S] is First、Follow set

	First	explain	Follow	explain
S->AB	if	First(AB)，A begin with if	#,else	（1）#：S# or S#=>AB#=A else S# （2）else： S#=>AB#=>if e then S B#=>if e then S else S# （3）if could not follow S directly Reason1: 'If' is the first symbol of A (A->if e then S) But it is impossible to appear If e then S A. So since SA could not appear, "S if" is impossible to appear. Reason2: SS is impossible to appear too. That is "if e then S S" is impossible. Since generally there must be a ';' between S (S;S).
A->if e then S	if		#,else	（1）#：S#=>AB#=>Aε#=A# （2）else：S#=>AB#=>A else S# （3）if is not in the follow set. Reason is as above rule.
B->else S	else		**#,else** **（B->else S and B->ε conflict）**	（1）#：S#=>AB# （2）else： S#=>AB#=>if e then S B#=>if e then AB B# =>if e then A B else S#
B->ε	ε			

LL(1) Parsing Table：

	If	then	else	#
S	S->AB			
A	A->if e then S			
B			**B->else S** **B->ε**	B->ε

（2）It is not LL(1). Reason: in the LL(1) table, there is conflict in cell of [B,else].

（3）Is the grammar ambiguous?

Yes.

Proof:

The conflict in [B,else] cell is caused by the famous ambiguous phenomena named if-else dangling.

Classic example : if e then if e then S else S

S#=>AB#

=>if e then S B#

=>if e then A B B#

=>if e then if e then S B B#

For the statement "if e then if e then S else S#", we could find two different left-most derivations:

First, if e then if e then S ε B#

Second, if e then if e then S else S ε#

【Q8】Left recursive grammar must not be LL(1)

【Q8】Proof the conclusion: if a grammar has left recursive, it must not be LL(1) grammar.

Answer:

Proof:

Suppose there is left recursive. Then there must be the two derivation segments as following:

X->β1

X->β2

In the two segments, we suppose that X->β1 is left recursive (then X->β2 must be able to derivate to a string only containing terminal symbols. Otherwise the grammar will contain useless symbol).

That is X=>β1=>X...

On the same time, in X=>β2, β1 can begin with any terminal symbol (we represent as a)。

Then First(β1) ∩First(β2) will at least contain a。

Thus it is not a LL(1) grammar。

【Q9】Left factor and LL(1)

【Q9】Proof: grammar containing left factor must not be LL(1)

Answer:

Proof:

Suppose there is left factoring. Then there must be the two derivation segments as following:

X->aβ1

X->aβ2

In the two segments, we suppose that X->aβ1 is left factoring (then X->aβ2 must be able to derivate to a string only containing terminal symbols. Otherwise, the grammar will contain useless symbol).

That is X=>β1=>X...

On the same time, in X=>β2, β1 can begin with any initial terminal symbol (we represent as a)。

Then First X ∩ Second X will at least contain initial terminal value a。

X won't know which production to consider because X->aβ1 and X->aβ2 must share some elements

Thus it is not a LL(1) grammar。

Note

We can see that initial symbol of each production is same. Therefore, it's having left factoring.

X won't know which production to consider, because of having same symbol at the right hand side.

The reason is that the first sets of X->aβ1 and X->aβ2 must share some elements among themselves。