Chapter 7

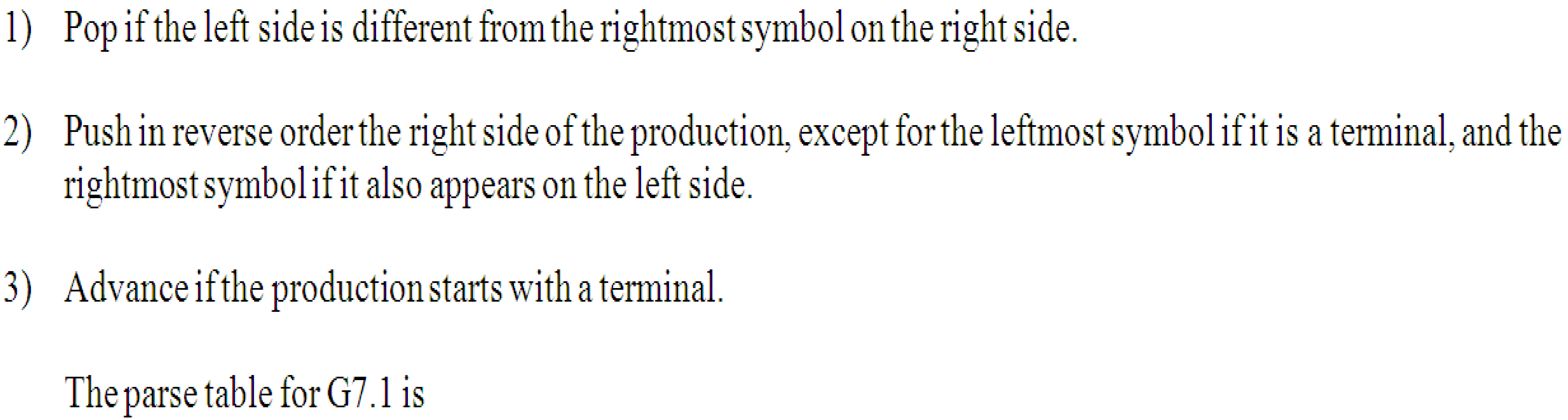
LL(1) Grammars

FIRST sets

G7.1

	FIRST set of right side
1) S → BC	{b, d}
2) S → CB	{c, e}
3) B → bB	{b}
4) B → d	{d}
5) C → cC	{c}
6) C → e	{e}

Operation sequence for a production



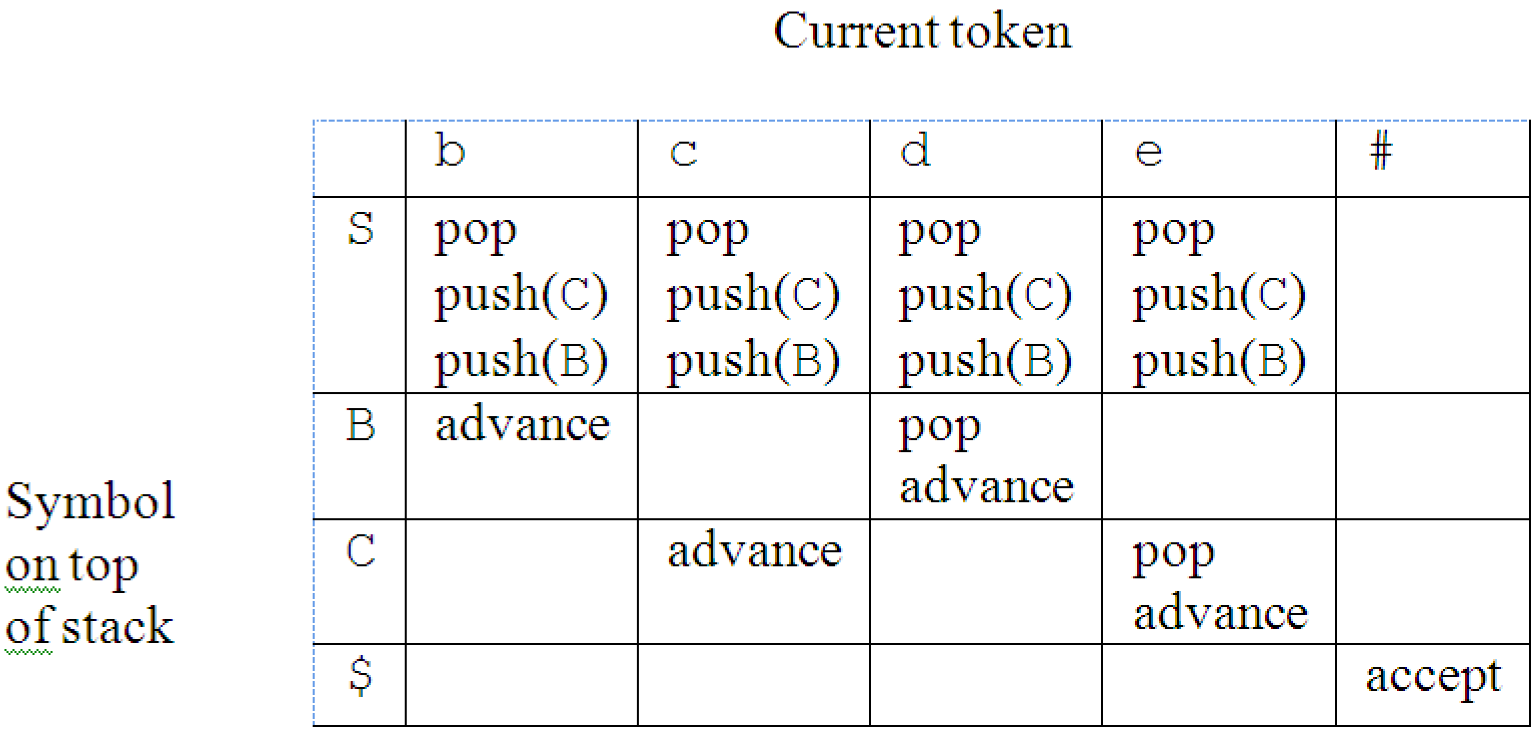
1) Pop if the left side is different from the rightmost symbol on the right side.

2) Push in reverse order the right side of the production, except for the leftmost symbol if it is a terminal, and the rightmost symbol if it also appears on the left side.

3) Advance if the production starts with a terminal.

The parse table for G7.1 is

Parse table for G7.1



Current token

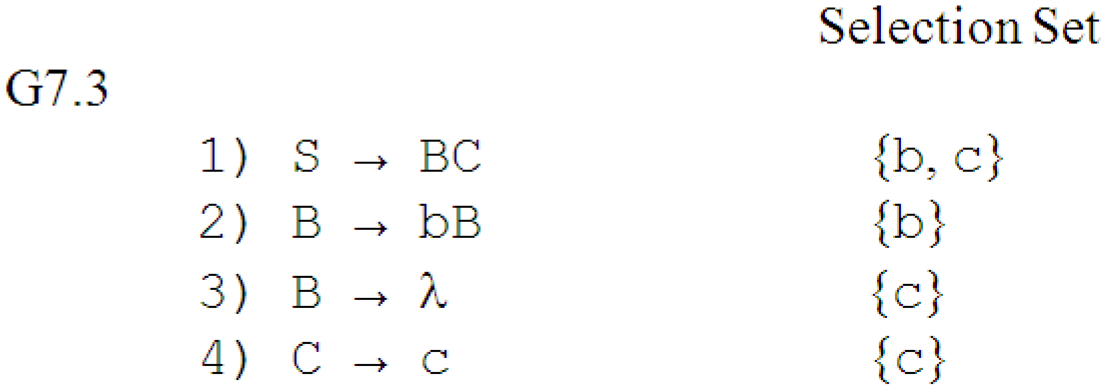
Symbol on top of stack

	b	c	d	e	#

S	pop
	push(C)
	push(B)
		pop
		push(C)
		push(B)
			pop
			push(C)
			push(B)
				pop
				push(C)
				push(B)	

B	advance		pop advance		
C		advance		pop advance	
$					accept

Lambda productions



G7.3

		Selection Set
1)	S → BC	{b, c}
2)	B → bB	{b}
3)	B → λ	{c}
4)	C → c	{c}

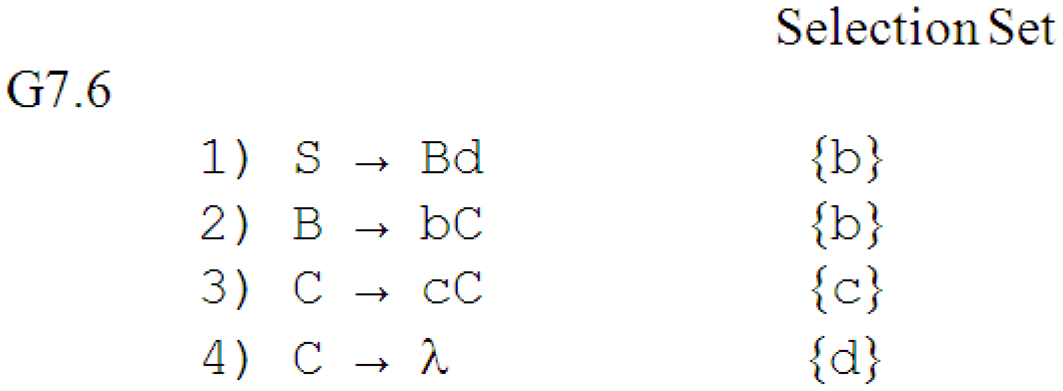
FOLLOW SET

The selection set for

B -> λ

is FOLLOW(B)

Whatever follows left follows rightmost

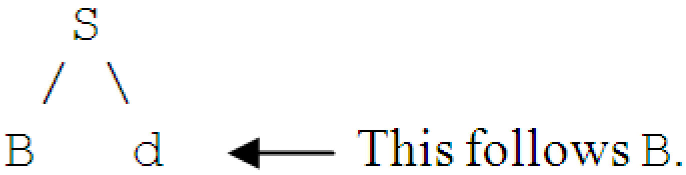


G7.6

	Selection Set
1) S → Bd	{b}
2) B → bC	{b}
3) C → cC	{c}
4) C → λ	{d}

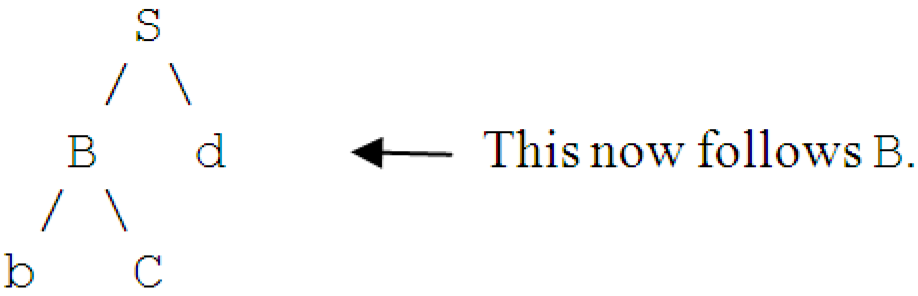
Whatever follows B follows C

a)



This follows B.

b)



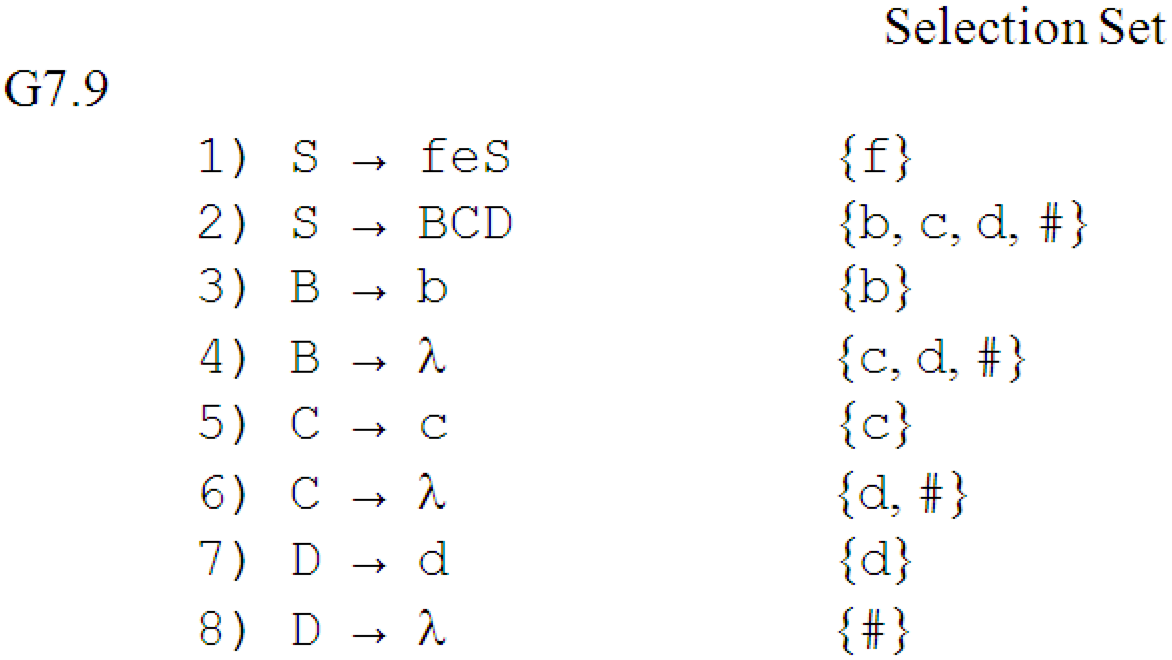
This now follows B.

Selection sets for nullable right sides

G7.8

			Selection Set
1)	S → AD	FIRST(AD) =	{b, c, d}
2)	A → BC	FIRST(BC) | FOLLOW(A) =	{b, c} | {d} = {b, c, d}
3)	B → b	FIRST(b) =	{b}
4)	B → λ	FOLLOW(B) =	{c, d}
5)	C → c	FIRST(c) =	{c}
6)	C → λ	FOLLOW(C) =	{d}
7)	D → d	FIRST(d) =	{d}

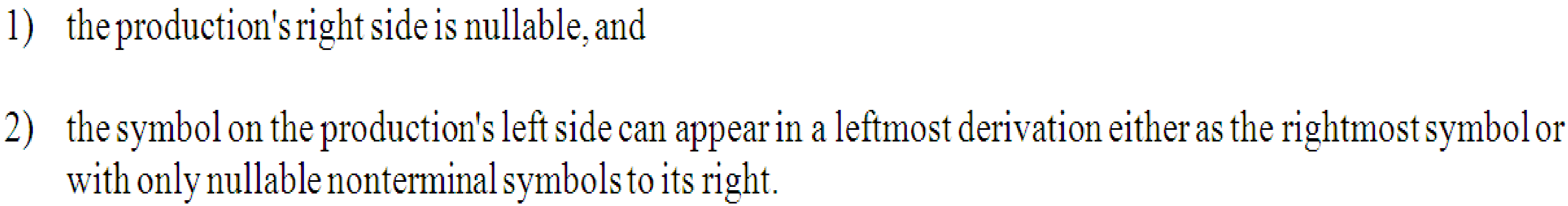
Selection sets with end of input



G7.9

	Selection Set
1) S → feS	{f}
2) S → BCD	{b, c, d, #}
3) B → b	{b}
4) B → λ	{c, d, #}
5) C → c	{c}
6) C → λ	{d, #}
7) D → d	{d}
8) D → λ	{#}

End of input in selection set if



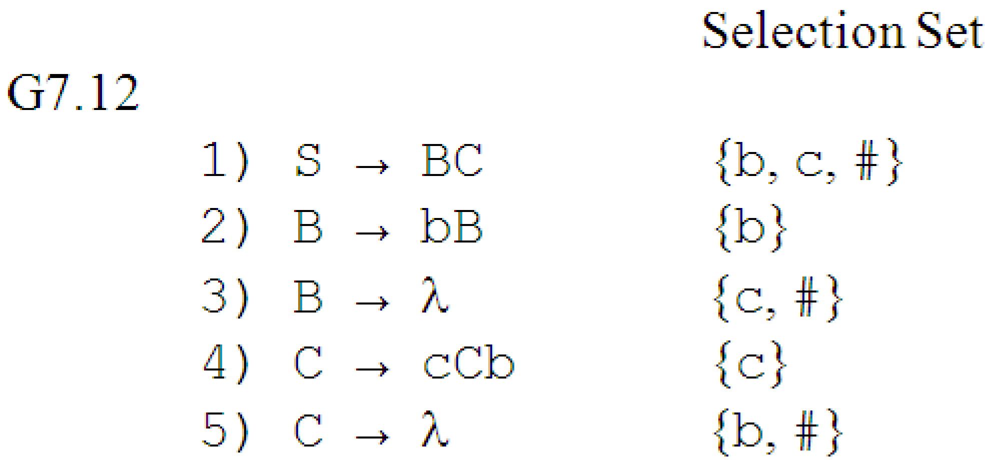
1) the production's right side is nullable, and

2) the symbol on the production's left side can appear in a leftmost derivation either as the rightmost symbol or with only nullable nonterminal symbols to its right.

Computing selections that contain

Put # in FOLLOW(S) can compute selection sets accordingly.

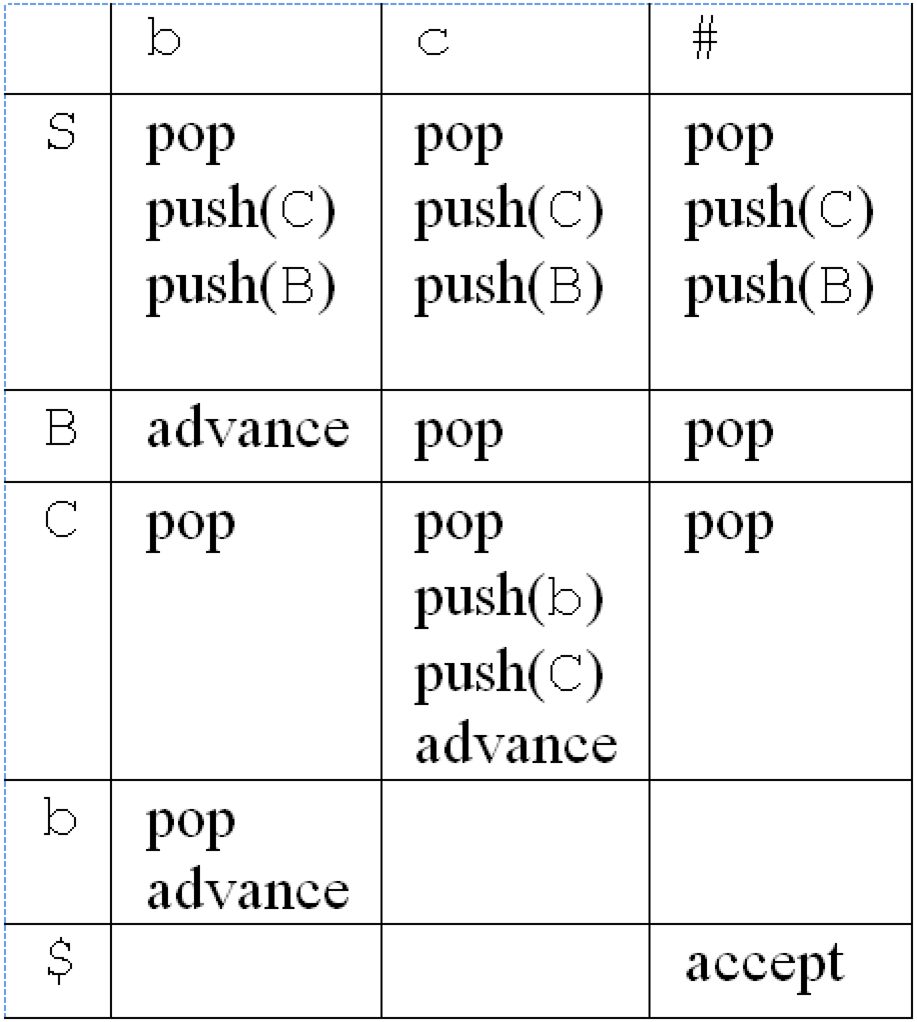
Lambda productions in stack parser



G7.12

				Selection Set
1)	S	→	BC	{b, c, #}
2)	B	→	bB	{b}
3)	B	→	λ	{c, #}
4)	C	→	cCb	{c}
5)	C	→	λ	{b, #}

Parse table



	b	c	#

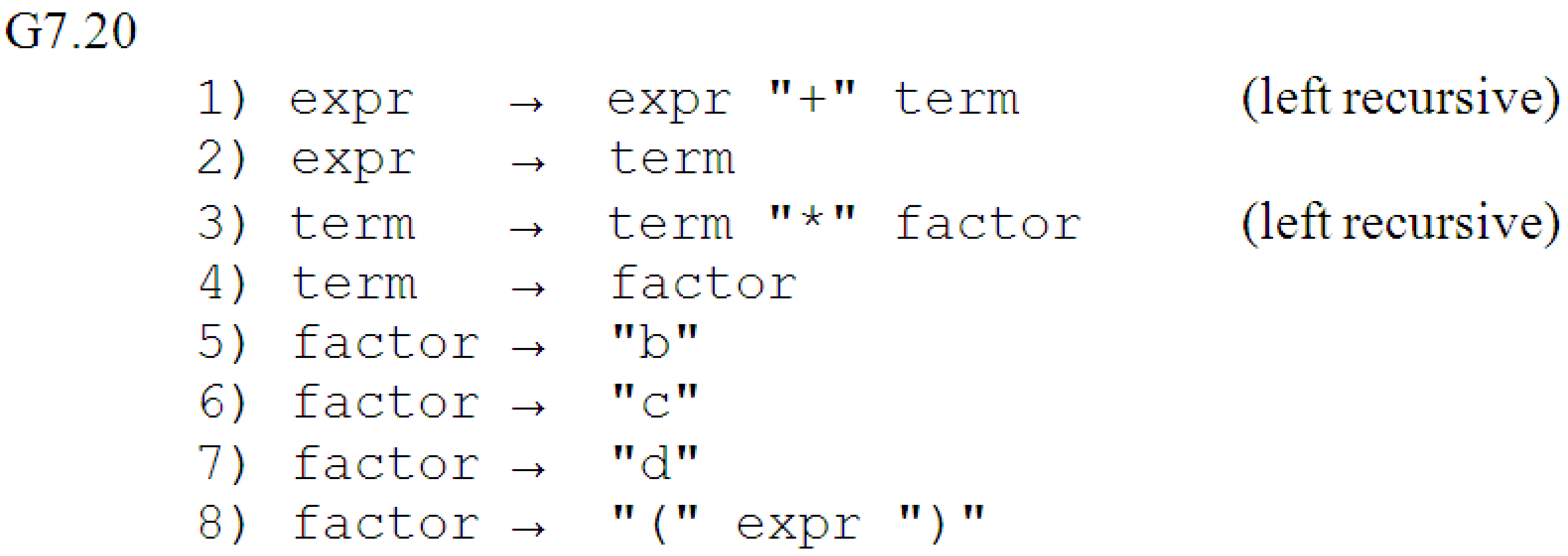
S	pop
	push(C)
	push(B)
		pop
		push(C)
		push(B)
			pop
			push(C)
			push(B)

B	advance	pop	pop

C	pop	pop
		push(b)
		push(C)
		advance
			pop

b	pop advance		
$			accept

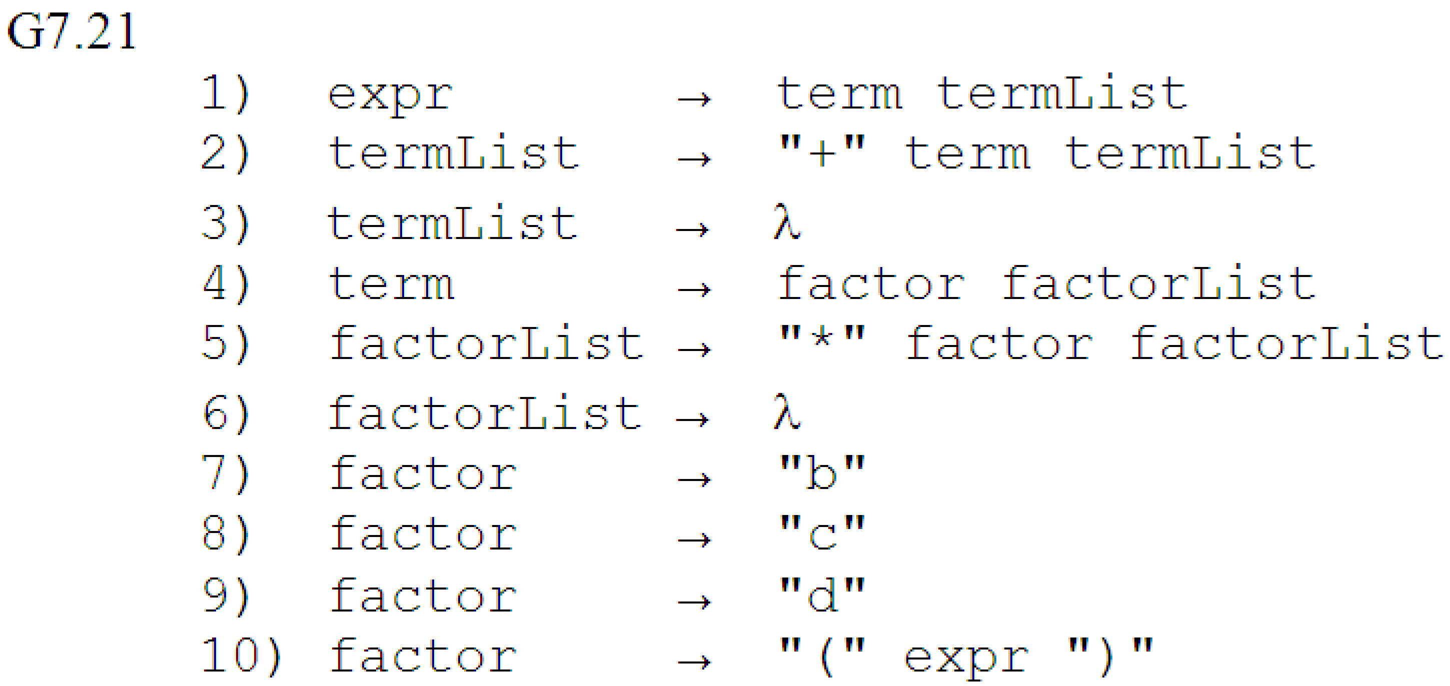
Left recursions means not LL(1)



G7.20

```
1) expr   →  expr "+" term        (left recursive)
2) expr   →  term
3) term   →  term "*" factor      (left recursive)
4) term   →  factor
5) factor →  "b"
6) factor →  "c"
7) factor →  "d"
8) factor →  "(" expr ")"
```

Eliminate left recursion



G7.21

```
1)  expr        →  term termList
2)  termList    →  "+" term termList
3)  termList    →  λ
4)  term        →  factor factorList
5)  factorList  →  "*" factor factorList
6)  factorList  →  λ
7)  factor      →  "b"
8)  factor      →  "c"
9)  factor      →  "d"
10) factor      →  "(" expr ")"
```

Left factoring

S -> bB	{b}	not LL(1)
S -> bC	{b}	
B -> bbb	{b}	
C -> ccc	{c}	

S -> bX	{b}	ll(1)
X -> B	{b}	
X -> C	{c}	
B -> bbb	{b}	
C -> ccc	{c}	

Left/right factoring

S -> bBd	{b}	not LL(1)
S -> bCd	{b}	
B -> bbb	{b}	
C -> ccc	{c}	

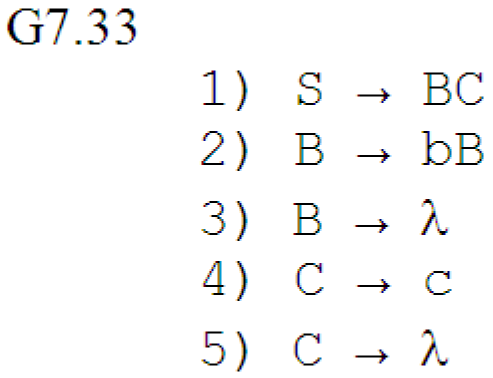
S -> bXd	{b}	LL(1)
X -> B	{b}	
X -> C	{c}	
B -> bbb	{b}	
C -> ccc	{c}	

Parsing with ambiguous grammar

G7.30

	Selection Set	
1) S → λ	{c, #}	
2) S → bSQ	{b}	
3) Q → cS	{c}	
4) Q → λ	{c, #}	← delete c to disambiguate

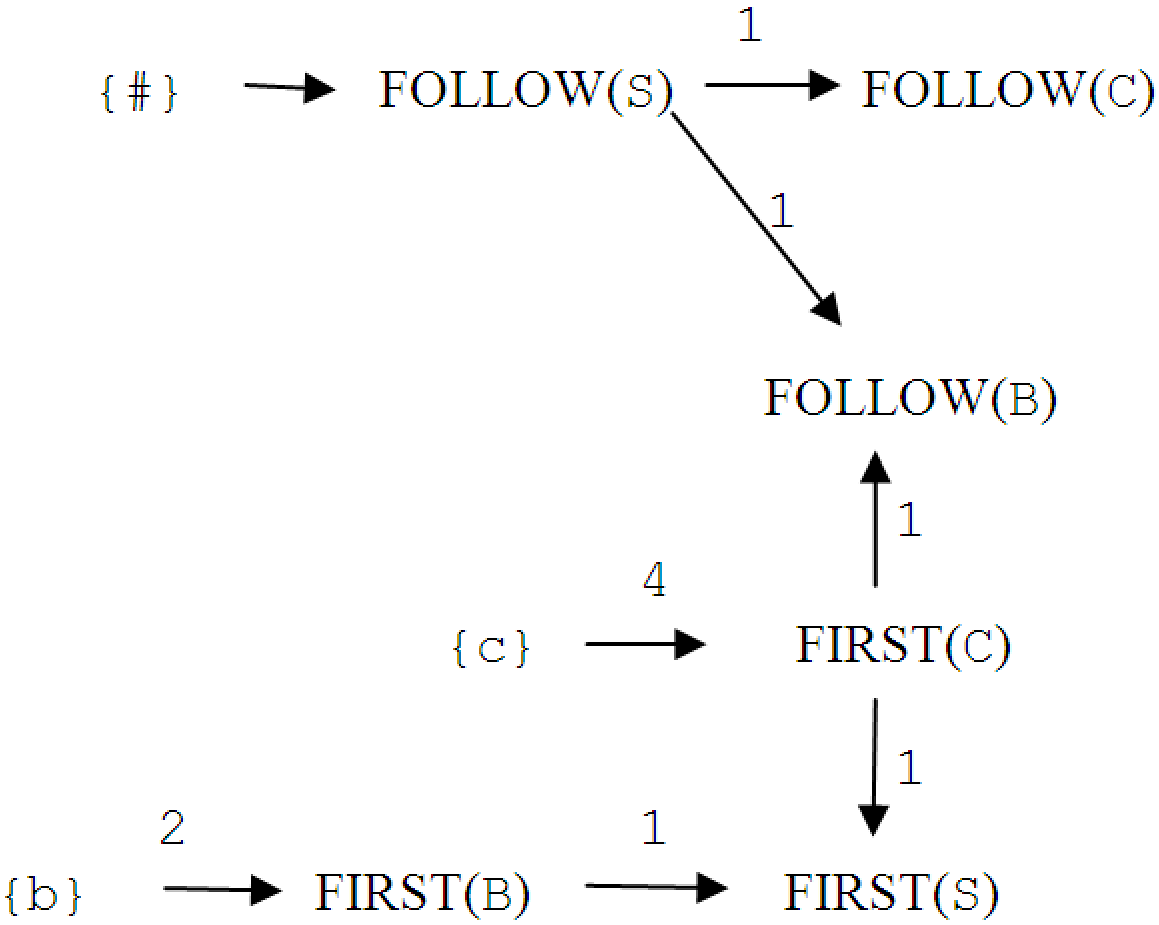
Computing FIRST and FOLLOW sets



G7.33

1) S → BC
2) B → bB
3) B → λ
4) C → c
5) C → λ

Create graph



1
{#}
FOLLOW(S)
FOLLOW(C)
1
FOLLOW(B)
1
4
{c}
FIRST(C)
1
2
1
{b}
FIRST(B)
FIRST(S)