

Clay Ramey

COMP-BIZZ

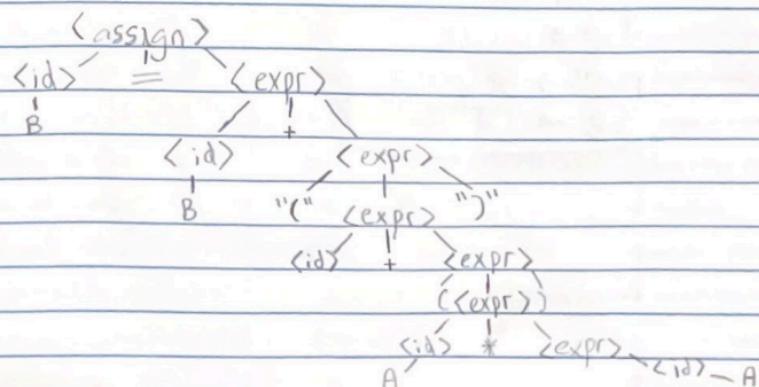
HW 3

2/16/24

- A : D are valid sentences  
B : C are incorrect

lexeme	token
=	Assign_OP
+	Add_OP
*	Mult_OP
(	L_PAREN
)	R_PAREN
H, B, C	Ident

- ③  $\text{Assign} \rightarrow \langle \text{id} \rangle = \langle \text{expr} \rangle$   
 $\rightarrow B = \langle \text{expr} \rangle$   
 $\rightarrow B = \langle \text{id} \rangle + \langle \text{expr} \rangle$   
 $\rightarrow B = B + \langle \text{expr} \rangle$   
 $\rightarrow B = B + (\langle \text{expr} \rangle)$   
 $\rightarrow B = B + (\langle \text{id} \rangle + \langle \text{expr} \rangle)$   
 $\rightarrow B = B + (c + \langle \text{expr} \rangle)$   
 $\rightarrow B = B + (c + (\langle \text{expr} \rangle))$   
 $\rightarrow B = B + (c + (\langle \text{id} \rangle * \langle \text{expr} \rangle))$   
 $\rightarrow B = B + (c + (A * \langle \text{expr} \rangle))$   
 $\rightarrow B = B + (c + (A * \langle \text{id} \rangle))$   
 $\rightarrow B = B + (c + (A * A))$



2/16/24 - Clay Ramey

## COMP-3770 HW 3

4 (4)

$$A \rightarrow Aa|Abc|C \quad A \rightarrow Aa|Abc \quad A \rightarrow C$$

$$S \rightarrow Aa|Bb \quad A \rightarrow Aa|Abc$$

$$A \rightarrow Aa|Abc \quad A \rightarrow A(a|bc)$$

$$A \rightarrow C \quad A \rightarrow AA' : A' \rightarrow a|bc$$

$$B \rightarrow S|bb$$

$$C \rightarrow \epsilon$$

$$S \rightarrow Aa|Bb$$

$$A \rightarrow CA'$$

$$A' \rightarrow bCA'|baA'|e$$

$$B \rightarrow AaB'|bbB'$$

$$B' \rightarrow bB'|e$$

$$C \rightarrow c$$

5

$$A \rightarrow abclacla$$

$$B \rightarrow blab$$

$$A \rightarrow aA'$$

$$A' \rightarrow Bclclce$$

$$B \rightarrow blab$$

$$A \rightarrow a\alpha|a$$

$$A \rightarrow aA'$$

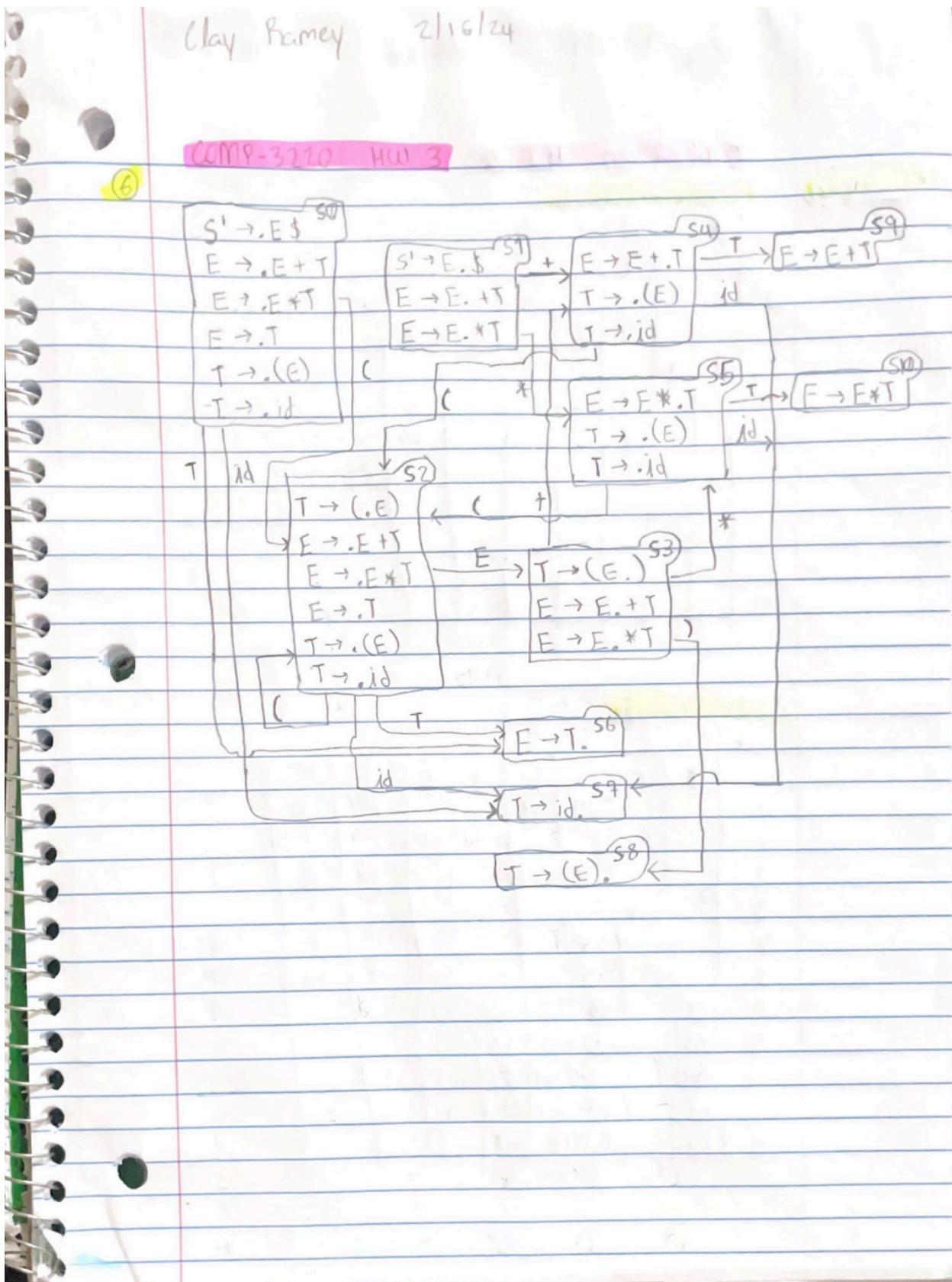
$$A' \rightarrow \alpha$$

$$A \rightarrow aA'$$

$$A' \rightarrow A''|e$$

$$A'' \rightarrow Bclc$$

$$B \rightarrow blab$$



Clay Ramey

2/16/24

## COMP-3220 HW 3

# 6 - continued

## Transition Table

	*	+	id	(	)	E	T
S0			7	2		1	6
S1	5	4					
S2			7	2		3	6
S3	5	4			8		
S4			7	2			9
S5			7	2	-		10
S6							
S7							
S8							
S9							
S10							

## Parser Table

State	Action							Go to	
	*	+	id	(	)	\$	E		
0			S7	S2			1	6	
1	S5	S4				acc			
2			S7	S2			3	6	
3	S5	S4							
4			S7	S2				9	
5			S7	S2				10	
6	R3	R3	R3	R3	R3	R3			
7	R5	R5	R5	R5	R5	R5			
8	R4	R4	R4	R4	R4	R4			
9	R1	R1	R1	R1	R1	R1			
10	R2	R2	R2	R2	R2	R2			

# COMP 3220 – HW3 – Clay Ramey

1. Input: $(id + id)^* id \$$ Stack: 0 Output:	7. Input: $(id + id)^* id \$$ Stack: 0 (2 E 3 Output: 5, 3	13. Input: $(id + id)^* id \$$ Stack: 0 (2 E 3 + 4 T 9 Output: 5, 3, 5	19. Input: $(id + id)^* id \$$ Stack: 0 T 6 Output: 5, 3, 5, 1, 4
2. Input: $(id + id)^* id \$$ Stack: 0 (2 Output:	8. Input: $(id + id)^* id \$$ Stack: 0 (2 E 3 + Output: 5, 3	14. Input: $(id + id)^* id \$$ Stack: 0 (2 E Output: 5, 3, 5, 1	20. Input: $(id + id)^* id \$$ Stack: 0 E Output: 5, 3, 5, 1, 4, 3
3. Input: $(id + id)^* id \$$ Stack: 0 (2 id 7 Output:	9. Input: $(id + id)^* id \$$ Stack: 0 (2 E 3 + 4 Output: 5, 3	15. Input: $(id + id)^* id \$$ Stack: 0 (2 E 3 Output: 5, 3, 5, 1	21. Input: $(id + id)^* id \$$ Stack: 0 E Output: 5, 3, 5, 1, 4, 3
4. Input: $(id + id)^* id \$$ Stack: 0 (2 T Output: 5	10. Input: $(id + id)^* id \$$ Stack: 0 (2 E 3 + 4 id Output: 5, 3	16. Input: $(id + id)^* id \$$ Stack: 0 (2 E 3) Output: 5, 3, 5, 1	22. Input: $(id + id)^* id \$$ Stack: 0 E 1 Output: 5, 3, 5, 1, 4, 3
5. Input: $(id + id)^* id \$$ Stack: 0 (2 T 6 Output: 5	11. Input: $(id + id)^* id \$$ Stack: 0 (2 E 3 + 4 id 7 Output: 5, 3	17. Input: $(id + id)^* id \$$ Stack: 0 (2 E 3) 8 Output: 5, 3, 5, 1	23. Input: $(id + id)^* id \$$ Stack: 0 E 1 * Output: 5, 3, 5, 1, 4, 3
6. Input: $(id + id)^* id \$$ Stack: 0 (2 E Output: 5, 3	12. Input: $(id + id)^* id \$$ Stack: 0 (2 E 3 + 4 T Output: 5, 3, 5	18. Input: $(id + id)^* id \$$ Stack: 0 T Output: 5, 3, 5, 1, 4	24. Input: $(id + id)^* id \$$ Stack: 0 E 1 * 5 Output: 5, 3, 5, 1, 4, 3
25. Input: $(id + id)^* id \$$ Stack: 0 E 1 * 5 id Output: 5, 3, 5, 1, 4, 3	28. Input: $(id + id)^* id \$$ Stack: 0 E 1 * 5 T 10 Output: 5, 3, 5, 1, 4, 3, 5	29. Input: $(id + id)^* id \$$ Stack: 0 E 1 Output: 5, 3, 5, 1, 4, 3, 5, 2	
26. Input: $(id + id)^* id \$$ Stack: 0 E 1 * 5 id 7 Output: 5, 3, 5, 1, 4, 3	30. Input: $(id + id)^* id \$$ Stack: 0 E 1 \$ Output: 5, 3, 5, 1, 4, 3, 5, 2	31. Input: $(id + id)^* id \$$ Stack: accept! Output: 5, 3, 5, 1, 4, 3, 5, 2	
27. Input: $(id + id)^* id \$$ Stack: 0 E 1 * 5 T Output: 5, 3, 5, 1, 4, 3, 5			

$(id + id)^* id$

I used chatGPT and Chegg for this question. I also used an online PowerPoint for bottom up parsing to help me solve.

Ppt link: <https://karkare.github.io/cs335/lectures/07BottomUpParsing.pdf>

Clay Ramey

2/16/2024

## (COMP - 3220) HW 3

(8) Derivation	Rule Used
E	starting
$\rightarrow E * T$	r <sub>2</sub>
$\rightarrow E * id$	r <sub>5</sub>
$\rightarrow T * id$	r <sub>3</sub>
$\rightarrow (E) * id$	r <sub>4</sub>
$\rightarrow (E + T) * id$	r <sub>1</sub>
$\rightarrow (E + id) * id$	r <sub>5</sub>
$\rightarrow (T + id) * id$	r <sub>3</sub>
$\rightarrow (id + id) * id$	r <sub>5</sub>

\* gives output that is reverse of the order of rules that the rightmost derivation uses. This confirms it finds the correct handles in the input string \*