Given, CFG,

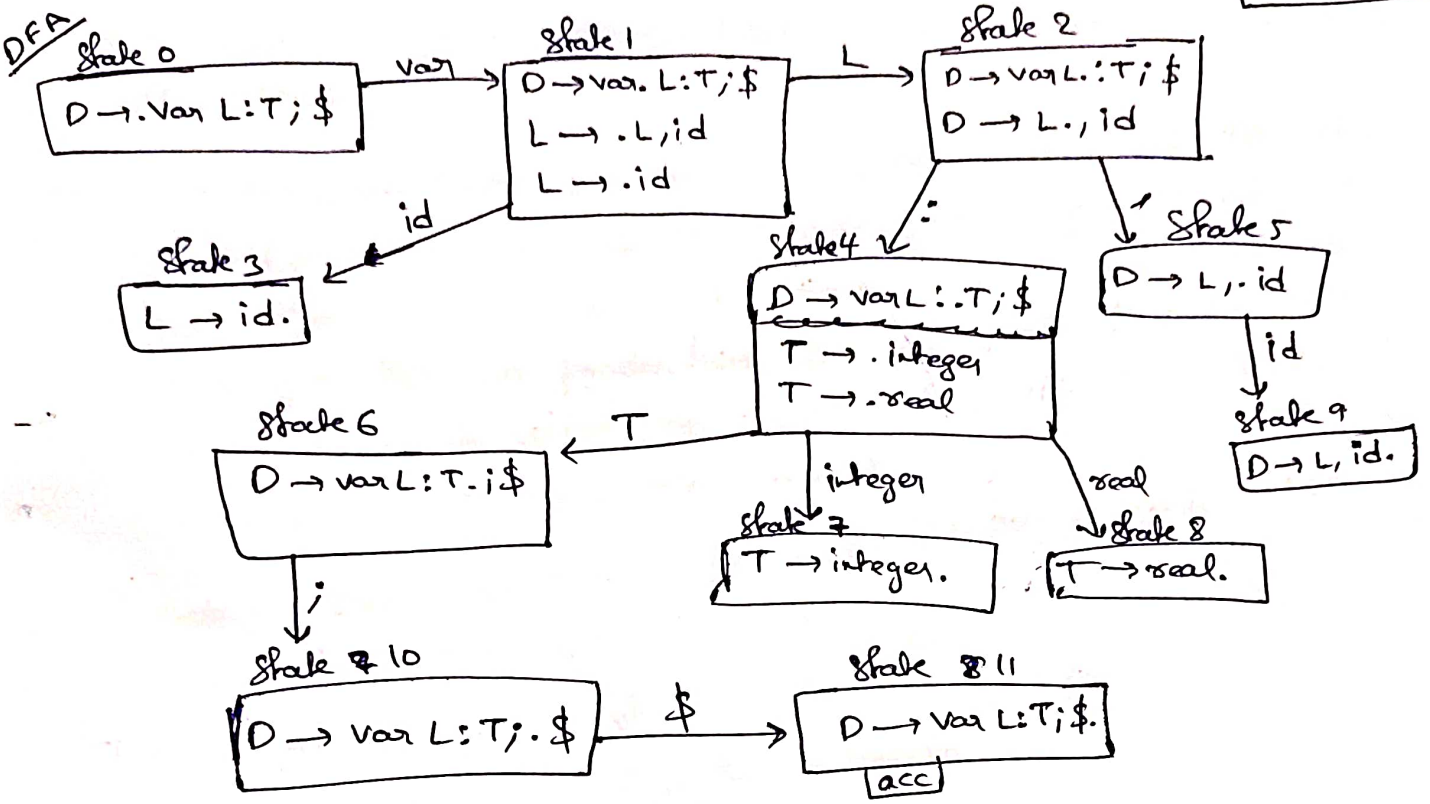
1) D → var L : T ; $
2) T → integer
3) T → real
4) L → L , id
5) L → id

Terminals = { id, ~~var~~ :, ;, ,, var, integer, real }

non-terminals = { D, T, L }

N. Yasaswin
20CS01040

DFA



LR(0) Parse Table is as follows

	action								Goto		
	id	:	;	,	var	integer	real	$	D	T	L
0					S1						
1	S3										2
2	~~S3~~	S4		S5							
3	R5	R5	R5	R5	R5	R5	R5	R5			
4						S7	S8			6	
5	S9										
6			S10								
7	R2	R2	R2	R2	R2	R2	R2	R2			
8	R3	R3	R3	R3	R3	R3	R3	R3			
9	R4	R4	R4	R4	R4	R4	R4	R4			
10								S11			
11	R1	R1	R1	R1	R1	R1	R1	R1			

Description of the process followed to build automaton and the parse table

1) First start with the start symbol production, placing a . infront of the production

2) Now for a state take the productions of each non-terminal that is present just right of the dot until and put dot at the start of those productions until no new productions get added

3) Fa a transition from one state to another, look at the element (terminal/non-terminal) of those are just right of dot and group those productions and move dot a step towards right and give them as ~~productions~~ starting productions for the next state to which we transit from the curr state on seeing the terminal/non terminal

4) Apply step 2 to step 3 for each and every state until no new states can adds up

Thus we get the automata

For Parse Table Generation, from the automata.

1) Fa each state in the automata, on seeing a terminal, the state which we move is mentioned as a shift action to that state in the corresponding cell of the state and the terminal in the parse table.

2) If we see a non-terminal transition from the current state to another state, we mark it at Goto part of the parse table corresponding to that row and column

3) If we see the dot (marker) at the end of a production in the list of productions the state has then we mark ~~all the~~ ~~terminal~~ the entire row of the corresponding state with a reduce action of that production.

4) we do the above steps for each and every state of the automata

Yes, the grammar is LR(0), no-conflicts in the table