Lecture 8

Monday
18/11/24

① $E \to E+T$

② $E \to T$

③ $T \to T*F$

④ $T \to F$

⑤ $F \to (E)$

⑥ $F \to id$

Parsing Simulation

SLR(1) → look at one token ahead.

* (→ will start the steps.

(I) From I_0 if we get '(' as the next token. It says we should shift to 4. The parser doesnot shift token in stack. It just shifts the state.

(II) Now 4 is top of stack & input has id. So shift to 5.

* For reducing, we go to the production & see how many things are in the body. Pop that many states from the stack.

(III) From 5 we get + ∴ pop 5 from stack since only one item on the body. But this time we will not remove + from input. If production body had T*F ∴ 3 things we would pop 5 items from stack. Now we will assume we are in state 4 and we got F ← since id reduced to F.

(IV) we are in 3 and we got rd4 from +. Now state 4 we seen T. ∴ go to state 2. → means reduce 4th production rule.

* always consume input only when we shift not reduce.

I0 th state

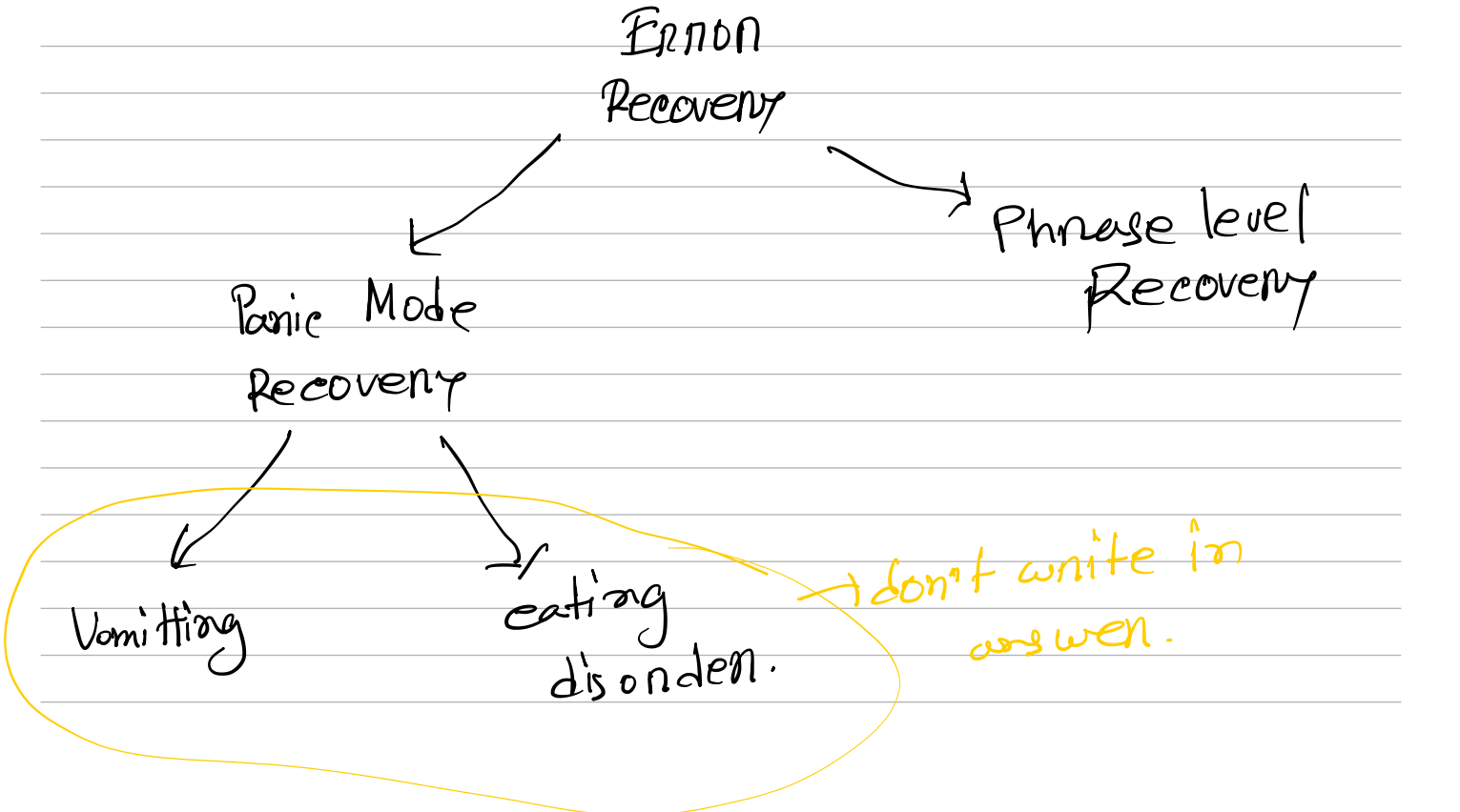
	Stack	Action	input string
	$0		(id + id) * id
			augmented → (id + id) * id $
1	$04	shift	id + id) * id $
2	$045 (include only this)	shift	+ id) * id $
3	$04 → $043 (step1 → step2)	reduce	+ id) * id $
4	$04 → $042	reduce	+ id) * id $
5	$04 → $048	reduce	+ id) * id $
6	$0486	shift	id) * id $
7	$04865	shift	) * id $
8	$0486 → $04863	reduce	) * id $
9	$0486 → $04869	reduce	) * id $
10	$04 → $048	reduce	) * id $
11	$04811	shift	* id $
12	$0 → $03	reduce	* id $
13	$0 → $02	reduce	* id $
14	$027	shift	id $
15	$0275	shift	$
16	$027 → $02710	reduce	$
17	$0 → $02	reduce	$
18	$0 → $01	reduce	$
19	**Accept**		

* If the corresponding entry is blank then program is syntatically error.

↳ we can't do either shift or reduce.

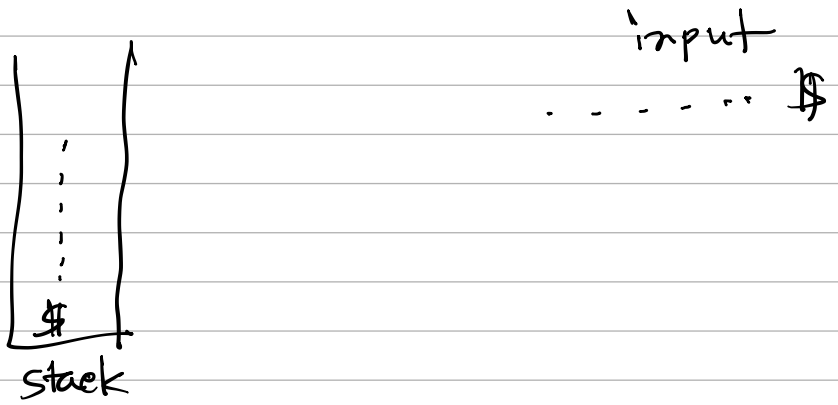
But how can the compiler indicate the errors? → Error Recovery in Compiler.

Though the error is in 3rd line, the compiler will recover itself from that stack position & go to the end to detect if there are more errors. Here the program is not recovered, the compiler recovered itself to reach end.



Panic Mode Recovery:

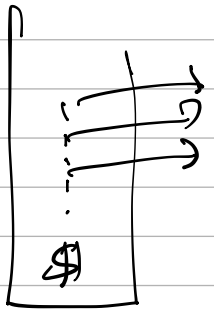
It has a stack & partial input.



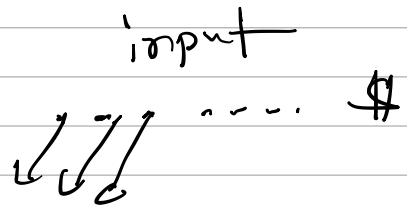
vomiting: I will keep dropping states from stack until we get a transition for the particular input symbol.

losing part information

Thus some part of the program I already read and validated gets lost. This is why sometimes compiler throws error like variable not assigned even though we assigned the variable.



Keep eating: keep consuming input tokens until we get a transition for current state. Thus we lose future informations.



Phrase level recovery

For each of the blank entries there will be instructions about what to do.

End of Syntax Analysis