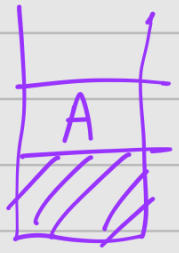
Lec 18/03

- always assume that strings end with string end marker $.

$S = abc \Rightarrow \overbrace{abc}^{S}\$$



? b c d

what all can be formed from current state of the parsing ?

i) all that is in first(A) can be produced by A when A was at ToS. so for all cells $[A, x]$ fill $A \to \alpha$ if $x \in$ first(A)

ii) if $\varepsilon \in$ first(A) then all in follow(A) can also be produced by the prodn $A \to \alpha$. so $\forall\ b \in$ follow(A) $M[A, b]$ we include $A \to \alpha$

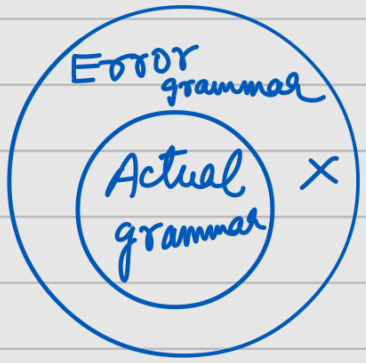
* We can rewrite the grammar only if language is LL(1). If lang is not LL(1) then we can't write the grammar

→ Error Recovery

Panic Mode:
use synchronising characters (; in C)

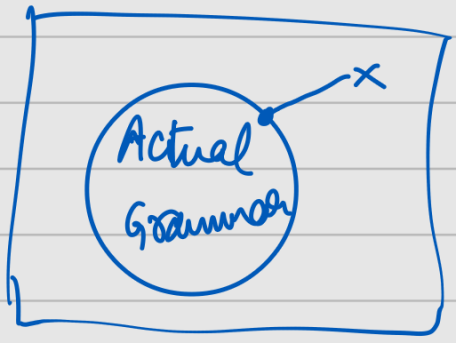
and keep dropping the lines containing error.

Error Production:
have some error prodn rules and create an extended grammar and if we are able to parse using an EP then we know exactly what error came



Global Corr:

Find the nearest string which belongs to the actual grammar from the error string. (Edit distance)



→ Bottom Up Parsing

productions
S → ..... → w Top Down

S → ..... → w Bottom Up
reductions

(given a terminal, produces set of all sentential forms that can produce it)

Rightmost derivation but read str from left to right

↳ LR(K) grammar.

bigger than LL(K) grammar

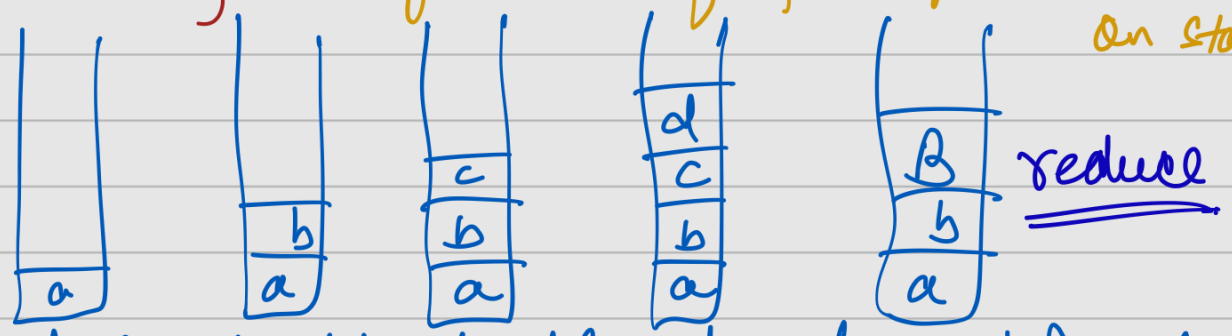
$$\frac{\frac{\frac{S}{B}}{A}}{abcd}$$

portion left to be read. — ab . cd — portion converted to sent form

separator .

on the stack, read right to left for Top to bottom on stack

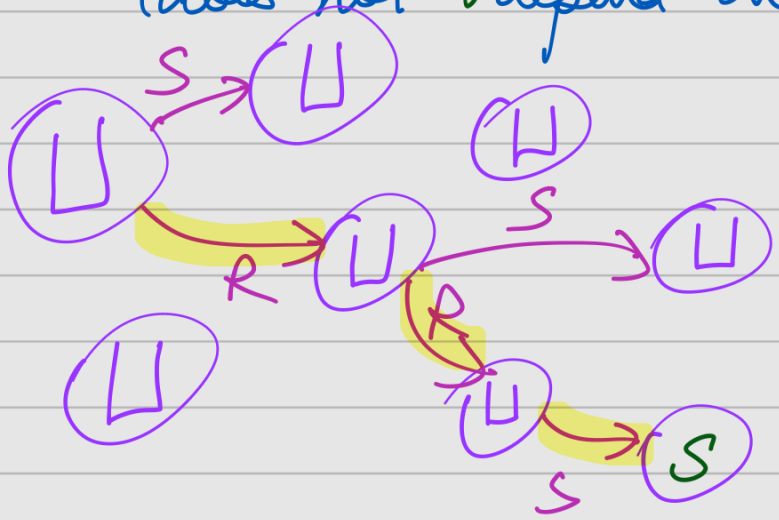
abcdef



a.bcdef ab.cdef abc.def abcd.ef abB.ef

State of the Parser

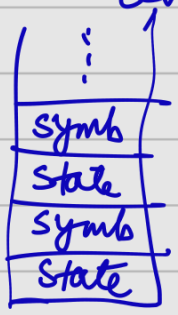
↳ State of the stack. (only).
(does not depend on state of string)



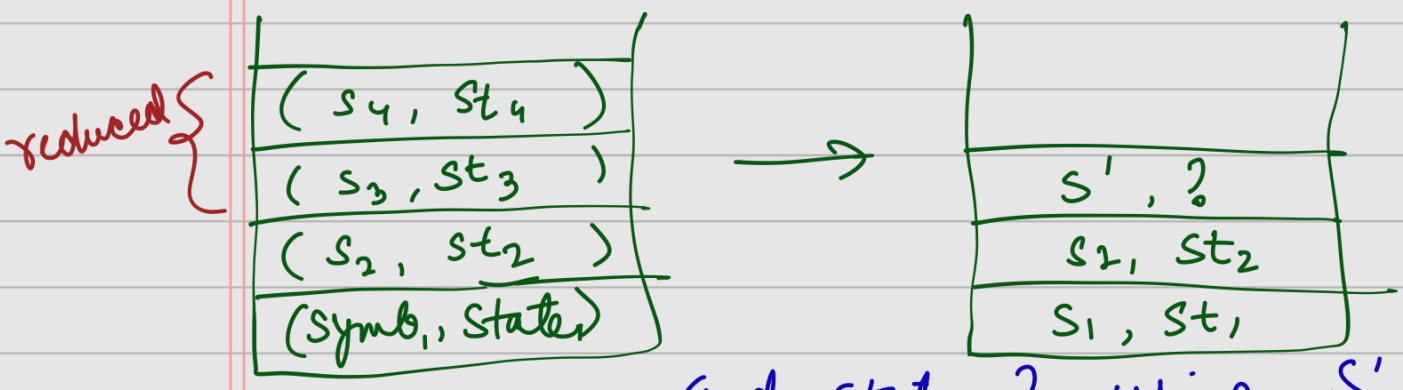
If the parser is able to disambiguate b/w S and R transitions, picking just one correctly on each state, such that the path correctly lead to the start state, then the grammar is LR.

* s4 — shift and goto state 4
* r4 — reduce using production #4

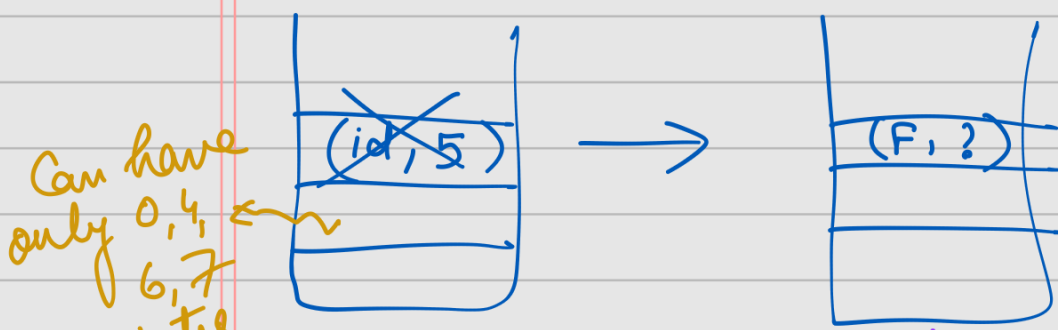
- We can either keep 2 stacks — one for state other for symbol or (alt pairwise)



Reduction



find state ? using S' and st_2
(new symbol came and prev state)



in the configuration of slides

β → sentential form
|β| → len(β)

if we reduce β we have to pop 2|β| elements
|β| states |β| symbols
(|β| tuples)