Push-down automata → PDA

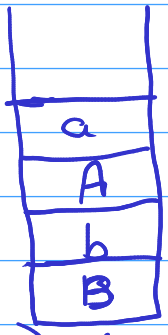
$S \to aAbB$
$A \to aA \mid c$
$B \to bB \mid a$

LL – Topdown
LR – Bottom up

"aacbba" a, a, c, b, b, a

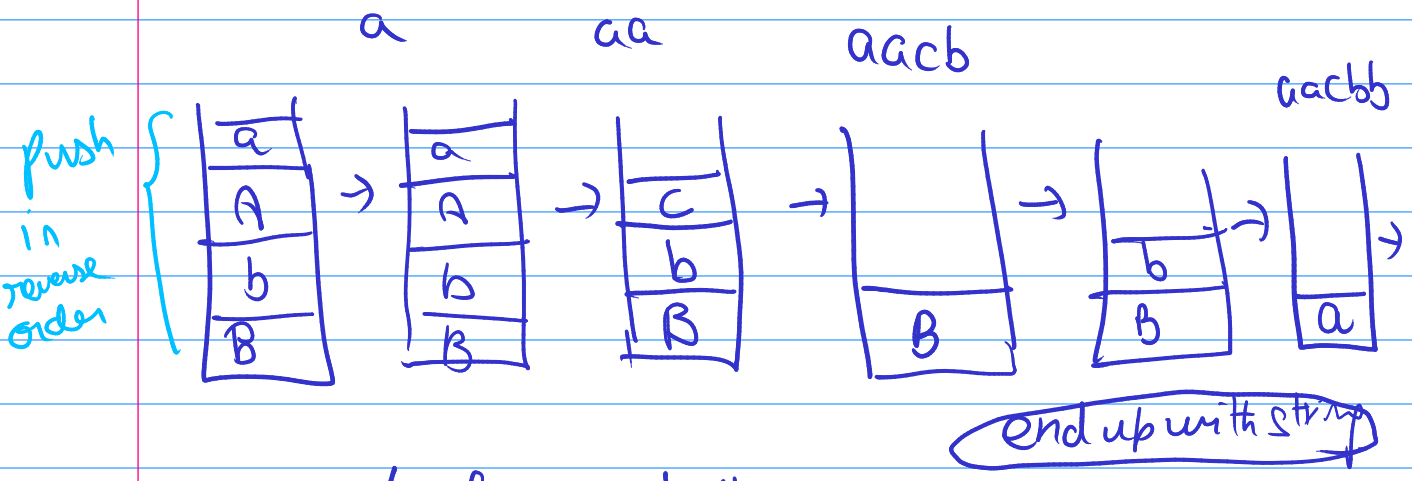
LL ← Topdown

Start with S



Whenever you see a non terminal on top of a stack, you remove it & replace with terminals or ~~not~~ whatever else it derives to

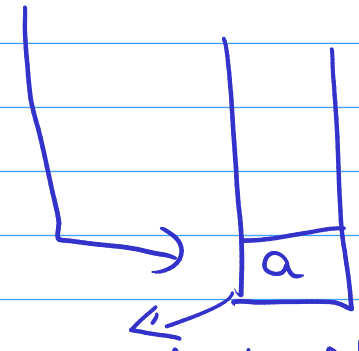
if terminal, just remove and compare with string provided it matches string



LR parser bottom up

Shift – shifting input symbol on stack
Reduce – reduce top of the stack with its LHS ~~desired~~

aacbba

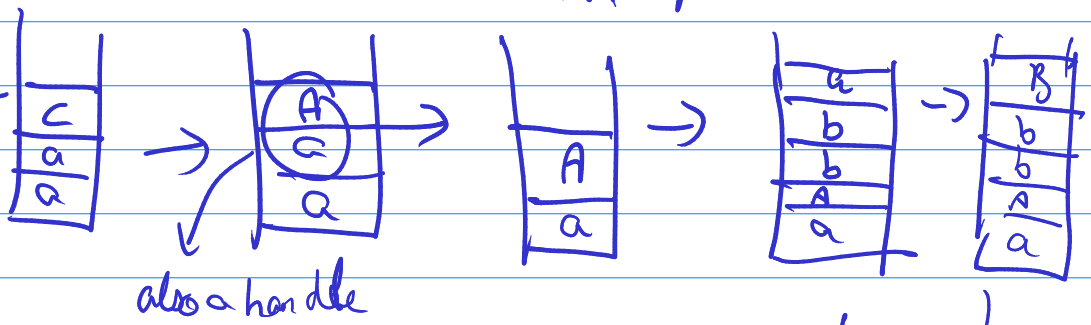


a is a handle for 'B'

now do I shift another a or do I reduce this 'a' to B?

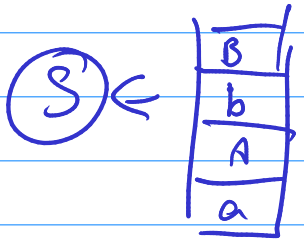
This is the fundamental problem of LR parsers

c is a handle

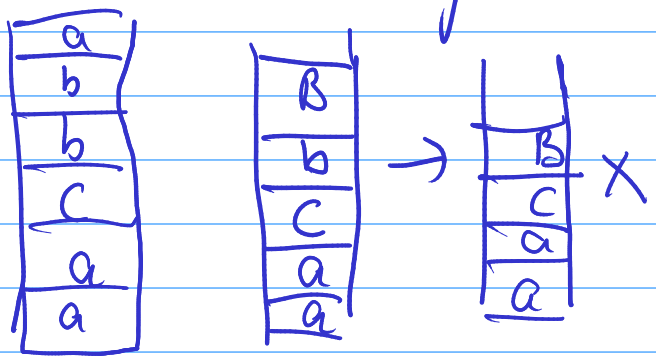


also a handle

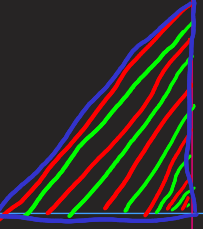
(Resolving this is done in ToC or compilers)



This wont work by shifting everything first BTW



Why is LR more powerful



$E \to E+T$

$E \to T$

$T \to F$

$F \to id$

a non terminal must have a rule

LL grammar
direct, indirect recursion

$E \to TE'$

$E' \to +T\,E' \mid \epsilon$

$T \to F$

$F \to id$

id + id

But no left recursion

$$A \to A\alpha_1 \mid A\alpha_2 \cdots$$

$$A \to B_1 \mid B_2$$

$$A \to B_1 A' \mid B_2 A'$$

$$A' \to \alpha_1 A' \mid \alpha_2 A' \mid \epsilon$$

$E \to E+T$

$A \quad A \quad \alpha_1$

$A \to A\alpha_1 \mid A\alpha_2 \mid \cdots A\alpha_n$

$A \to B_1 \mid B_2 \mid B_n$

$A' \to B_1 A' \mid B_2 A' \mid \cdots$

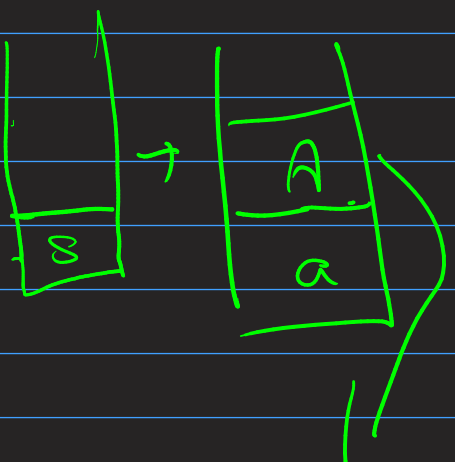
Check 1'B

id + id

$S \rightarrow aA$

$A \rightarrow bb\ \alpha_1$

$A \rightarrow a\ \alpha_2$



abb

ambiguity

what to replace with?

Look at next symbol

a b b

$F(\alpha_1) = b$ Follows

$F(\alpha_2) = a$

next symbol is 'b'