

Construct CFG for the language

$$L = \{a^n b^{2n} \mid n \geq 0\}$$

when $n=0$ $L = \epsilon$

$n=1$ $L = \{a, ab, aabb, \dots\}$

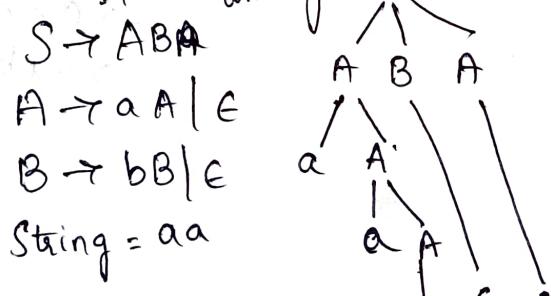
Ambiguous Grammar

- More than one left most derivation or more than one left most parse tree.
- More than one right most derivation or more than one right most parse tree.
- More than one parse tree.

Main application of this grammar is Compiler Construction

If the grammar is having ambiguity then it is not suitable for Compiler construction. The compiler will be evaluated with only one expression. If the same grammar derives two or more expressions then it is some what difficult for the compiler to choose an expression

Example: \rightarrow this grammar is ambiguous



Left Deriv

$S \rightarrow ABA$
 $\rightarrow aABA$
 $\rightarrow aaABA$
 $\rightarrow aaeBA$
 $\rightarrow aaBA$
 $\rightarrow aa\epsilon A$
 $\rightarrow aa\epsilon$
 $\rightarrow aa$

Now, we will consider one more LMD

$S \rightarrow ABA$
 $\rightarrow \epsilon BA$
 $\rightarrow \epsilon \epsilon A$
 $\rightarrow A$
 $\rightarrow aA$
 $\rightarrow aaA$
 $\rightarrow a\epsilon A$
 $\rightarrow aa$

RMD

$$E \rightarrow E + E$$

$$\rightarrow E + E * E$$

$$\rightarrow E + E * id$$

$$\rightarrow E + id * id$$

$$\rightarrow id + id * id$$

Right
Ex2:

$$E \rightarrow E + E$$

$$E \rightarrow E * E$$

$$E \rightarrow id$$

$$id + id * id$$

String

$$E \rightarrow E * E$$

$$\rightarrow E * id$$

$$\rightarrow E + E + id$$

$$\rightarrow E + id + id$$

$$\rightarrow id + id * id$$

this grammar is ambiguous.

from the above example, we will understand using the operators.

$$id + id + id$$

It has two operators on either side of it which operator should I associate this with?
If use associative with left operator \rightarrow left associative operator. $(id + id) + id$

If we associative with right associative \rightarrow right associative
 $id + \underline{(id + id)}$

In that both, + is in the right associative left, so 1st parse tree is correct.

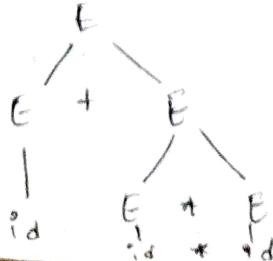
$$(id + id) + id$$

⑧ Y the grammar failed?

b/c rules of associativity failed.

In the operators +, -, *, /, the based on the priority of operators, the expression will evaluate first

$$id + (id + id)$$



$$id + (id * id)$$

first we will solve
(id * id) first then it's operator.

2nd derivation

$(id + id) * id$

$E \rightarrow E * E$

$\rightarrow (E + E) * id$

$\rightarrow \underline{(id + id)} * id$

here in this $(id + id)$ evaluates first then it goes to $*$ operator.

So this parse tree and derivation is not correct.

Rule : b/c precedence is not taken care of the expression

So the first parse tree is correct based on the Precedence.

So, if we can take care of i) associativity & ii) precedence then the grammar can be unambiguous.

② Give a PDA to accept the following language

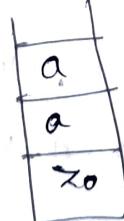
$L = \{a^n b^{2n} \mid n \geq 1\}$ by final state.

{abb, aabbbb, aaabbbbbb, ---}

concept is

aabbbb

two a's pushed into the stack.



After pushing two a's on the stack, and the next input string is 'b' for the 2nd b, 'a' should be popped.

a a b b b b
↑ ↑ b ↑ b ↑ b for the 3rd b, no change in the stack should be performed for the 4th b be popped.

for the 2nd b, no change in the stack should be performed for the 4th b be popped.

transition

$$\delta(q_0, a, z_0) = (q_0, a z_0)$$

$$\delta(q_0, a, a) = (q_0, a a)$$

$$\delta(q_0, b, a) = (q_1, a)$$

we have to
match
1st 'a' to 2nd'b's

$$\delta(q_1, b, a) = (q_2, \epsilon)$$

for the 1st'b' don't do
any operation, no change
in stack should be performed.

$$\delta(q_2, b, a) = (q_1, a)$$

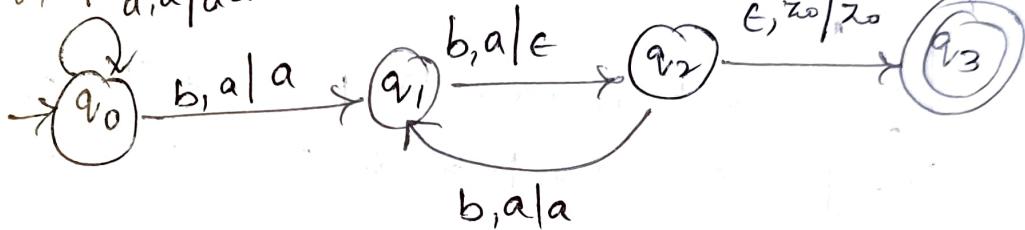
in stack should be performed.

$$\delta(q_2, \epsilon, z_0) = (q_3, \epsilon)$$

2nd'b', pop 'a' from the
3rd'b', no change in stack

$$a z_0 | a a | a a a$$

4th'b', pop 'a' from the
stack.



$$PDA = (Q = \{q_0, q_1, q_2, q_3\}, \Sigma = \{a, b\}, \Delta = \{a, z_0\}, \delta, q_0, z_0, q_3)$$

$$L = \{a^n b^m c^n \mid n \geq 1, m \geq 1\}$$

Suppose $L = \{abbcc, aabbcc, \dots\}$

the string = aabbcc

$$\delta(q_0, a, z_0) = (q_0, a z_0)$$

$$\delta(q_0, a, a) = (q_0, a a)$$

$$\delta(q_0, b, a) = (q_1, a)$$

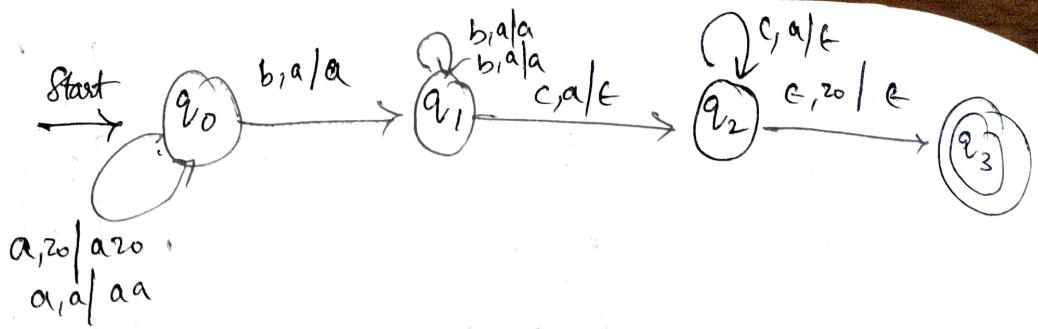
$$\delta(q_1, b, a) = (q_1, a)$$

$$\delta(q_1, c, a) = (q_2, \epsilon)$$

$$\delta(q_2, c, a) = (q_2, \epsilon)$$

$\delta(q_2, \epsilon, z_0) = (q_3, z_0) \rightarrow$ final state
 $(q_3, \epsilon) \rightarrow$ empty stack

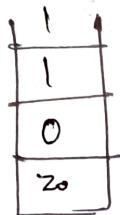
a	2
a	1
z ₀	



$$\text{PDA, } P_2 = \left(\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \{a, z_0\}, \delta, q_0, \{q_3\} \right)$$

Construct a DPDA for $L = \{w \in W^* \mid w \text{ is in } (0+1)^*\}$

Consider, 011C110



Just we have to push all the symbols either 0 or 1 on to the stack until 'c' is encountered.

If 'c' is the input, change the state from q_0 to q_1 , and for the remaining input, we have to do match, that should be matched to the top of the stack.

remaining is 110

011C110



$$\delta(q_0, 0, z_0) = (q_0, 0z_0)$$

$$0, z_0 | 0z_0$$

$$1, 1 | e$$

$$\delta(q_0, 1, 0) = (q_0, 10)$$

$$1, 0 | 10$$

$$0, 0 | e$$

$$\delta(q_0, 1, 1) = (q_0, 11)$$

$$1, 1 | 11$$

$$C, 1 | 1$$

$$e, z_0 | e$$

$$\delta(q_0, C, 1) = (q_1, 1)$$

$$1, 1 | e$$

$$\delta(q_1, 1, 1) = (q_1, e)$$

$$0, 0 | e$$

$$\delta(q_1, 1, e) = (q_2, e)$$

$$e, z_0 | e$$

$$\delta(q_1, e, e) = (q_2, e)$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e$$

$$1, 1 | e$$

$$0, 0 | e$$

$$e, z_0 | e</math$$