Ambiguity

Saturday, April 12, 2025   1:30 PM

There are two ways of generating strings from CFG.
1. Parse Tree
2. Derivation
   a. Leftmost derivation
   b. Rightmost Derivation

① S -> AB   Derivations for string ab
  A -> a
  B -> b

Leftmost derivation :
S => AB
  =>a B
  => a b

Rightmost derivation:
S => AB
  =>A b
  => ab
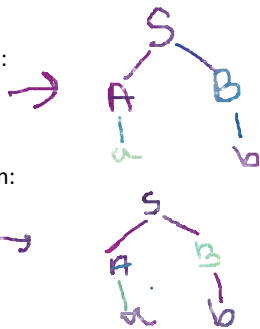
Ambiguity: If there exists two or more different leftmost derivations or rightmost derivations which generates different parse trees for a string, then the grammar is ambiguous.

Question 1 :
S -> S + S | S*S | a | b

Answer the following questions:

1. Give a leftmost derivation for string a + a * b
2. Sketch a parse tree for the derivation in (1)
3. Give a rightmost derivation for the string a + a * b
4. Sketch a parse tree for the derivation in (3)
5. Give another parse tree apart from the one you found in 2 and 4 and prove ambiguity.
6. Write the unambiguous version of the following grammar.
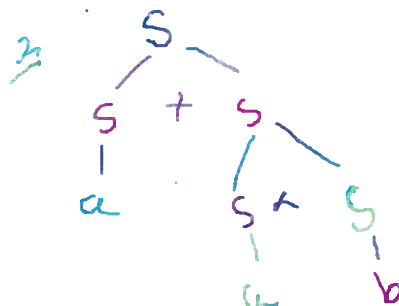
1. Leftmost derivation:
   S => S + S
     => a + S
     => a + S * S
     => a + a * S
     => a + a * b

3. Rightmost derivation:
   S => S + S
     => S + S * S
     => S + S * b
     => S + a * b
     => a + a * b
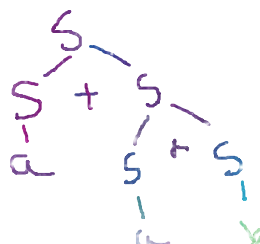
```
=> S + a * b
=> a + a * b
```

5. Another Leftmost derivation:
```
S => S * S
  => S + S * S
  => a + S * S
  => a + a * S
  => a + a * b
```
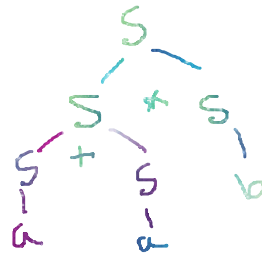
Since there exists two parse tree for a string the grammar is ambiguous.

Unambiguous grammar:
```
S -> S + T | T
T -> T * F | F
F -> a | b
```

```
S -> S* T | T
T -> T + F | F
F -> a | b
```
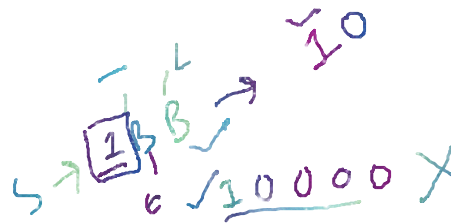
Question 2:
```
S -> B 1 B
B -> 0B | 1B | e
```

Answer the following questions:

1. Give a leftmost derivation for string 0101
2. Sketch a parse tree for the derivation in (1)
3. Give another parse tree apart from the one you found in 2 and prove ambiguity.
4. Find a string of length five which has exactly one parse tree.

1. Leftmost derivation :
```
S => B 1 B
  => 0B 1 B
  => 0 e 1 B
  => 0 1 0B
  => 0101B
  => 0101e => 0101
```
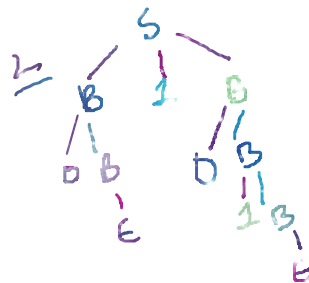
3. Leftmost derivation:
```
S => B 1 B
  => 0B 1 B
  => 0 1 B 1 B
  => 01 0 B 1B
  => 010 e 1B
  => 0101e => 0101
```

```
10000
01000
00100
00010
00001
```