GENERAL SIR JOHN KOTELAWALA DEFENCE UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Theory of Programing Languages - CS4022 Assignment 01

AHM Pushpakumara

6390

1. **Rewrite the BNF of Exercise 03 to give + precedence over * and force + to be right associative.**

This statement give in the Exercise 03 is x + y*z

<expr> ::= <term> | <term> + <expr> | (<expr>)

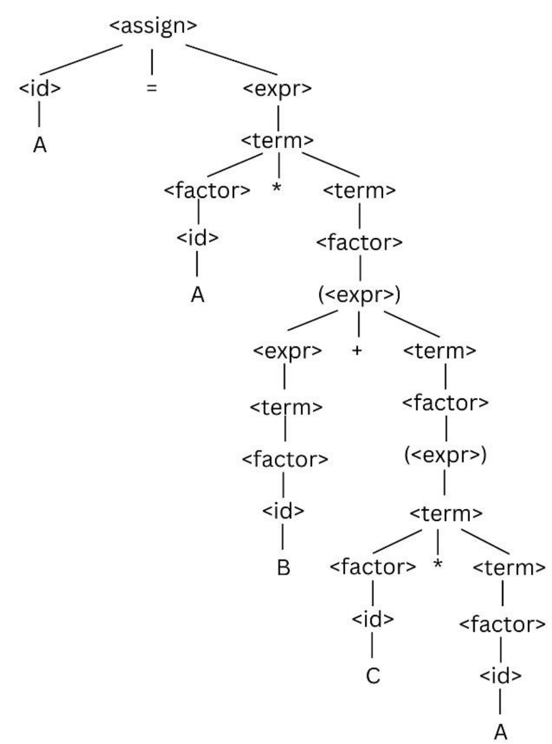
<term> ::= <factor> | <factor> * <term>

<factor> :: = <id>

<id> ::= x | y | z

2. **Using the grammar in Exercise 03, show a parse tree and a leftmost derivation for each of the following statements:**

    - **A = A * (B + (C * A))**

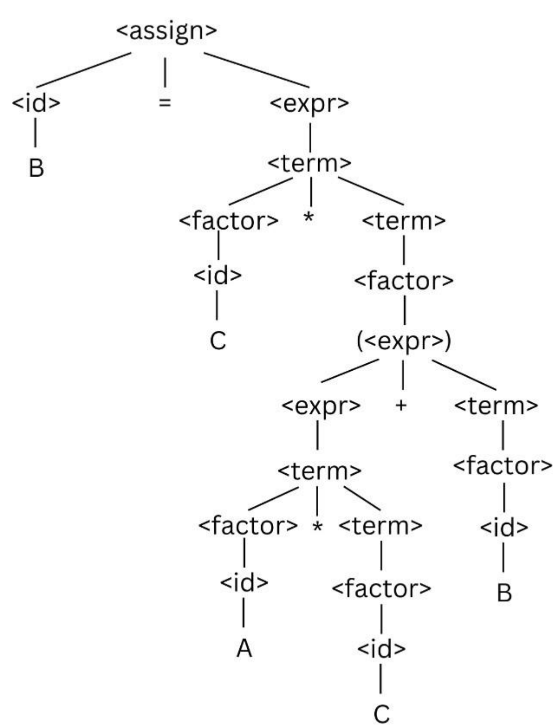


Leftmost derivation:

1. <assign> → <id> = <expr>
2. <id> = <expr> → A = <expr>
3. A = <expr> → A = <term>
4. A = <term> → A = <factor> * <term>
5. A = <factor> * <term> → A = <factor> * <term>

6. A = <factor> * <term> → A = A * <factor>
7. A = A * <factor> → A = A * (<expr>)
8. A = A * (<expr>) → A = A * (<expr> + <term>)
9. A = A * (<expr> + <term>) → A = A * (<term> + <term>)
10. A = A * (<term> + <term>) → A = A * (<factor> + <term>)
11. A = A * (<factor> + <term>) → A = A * (B + <term>)
12. A = A * (B + <term>) → A = A * (B + <factor>)
13. A = A * (B + <factor>) → A = A * (B + (<expr>))
14. A = A * (B + (<expr>)) → A = A * (B + (<term>))
15. A = A * (B + (<term>)) → A = A * (B + (<term> * <factor>))
16. A = A * (B + (<term> * <factor>)) → A = A * (B + (<factor> * <term>))
17. A = A * (B + (<factor> * <term>)) → A = A * (B + (C * <term>))
18. A = A * (B + (C * <factor>)) → A = A * (B + (C * A))

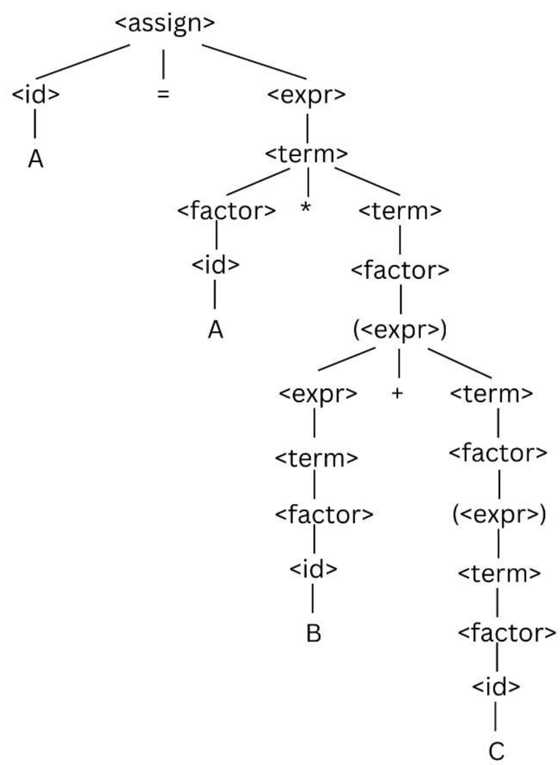
- **B = C * (A * C + B)**



Leftmost derivation:

1. <assign> → <id> = <expr>
2. <id> = <expr> → B = <expr>
3. B = <expr> → B = <term>
4. B = <term> → B = <factor> * <term>
5. B = <factor> * <term> → B = <factor> * <term>
6. B = <factor> * <term> → B = C * <factor>
7. B = C * <factor> → B = C * (<expr>)
8. B = C * (<expr>) → B = C * (<expr> + <term>)
9. B = C * (<expr> + <term>) → B = C * (<term> + <expr>)
10. B = C * (<term> + <expr>) → B = C * (<term> * <factor> + <term>)
11. B = C * (<term> * <factor> + <term>) → B = C * (<factor> * <factor> + <term>)
12. B = C * (<factor> * <factor> + <term>) → B = C * (A * <factor> + <term>)
13. B = C * (A * <factor> + <term>) → B = C * (A * C + <term>)
14. B = C * (A * C + <term>) → B = C * (A * C + <factor>)
15. B = C * (A * C + <factor>) → B = C * (A * C + B)

   - **A = A * (B + (C))**



Leftmost derivation:

1. <assign> → <id> = <expr>
2. <id> = <expr> → A = <expr>
3. A = <expr> → A = <term>
4. A = <term> → A = <factor> * <term>
5. A = <factor> * <term> → A = <factor> * <term>
6. A = <factor> * <term> → A = A * <factor>
7. A = A * <factor> → A = A * (<expr>)
8. A = A * (<expr>) → A = A * (<expr> + <term>)
9. A = A * (<expr> + <term>) → A = A * (<term> + <term>)
10. A = A * (<term> + <term>) → A = A * (<factor> + <term>)
11. A = A * (<factor> + <term>) → A = A * (B + <term>)
12. A = A * (B + <term>) → A = A * (B + <factor>)
13. A = A * (B + <factor>) → A = A * (B + (<expr>))

14. A = A * (B + (<expr>)) → A = A * (B + (<term>))

15. A = A * (B + (<term>)) → A = A * (B + (<factor>))

16. A = A * (B + (<factor>)) → A = A * (B + (C))

3. Consider the following grammar:

- **<S>→<A> a <B>b**
- **<A> →<A> b | b**
- **<B>→ b**

Which of the following sentences are in the language generated by this grammar?

1. babb

Start with `<S>`

`<S>` → `<A> a <B> b`

`<A> a <B> b` → `b a <B> b` (using `<A>` → `b`) `b`
`a <B> b` → `b a b b` (using `<B>` → `b`)

So babb is generated by this grammar.

2. bbbabb

Start with `<S>`

`<S>` → `<A> a <B> b`

`<A> a <B> b` → `<A> b a <B> b` (using `<A>` → `<A> b`)

`<A> b a <B> b` → `<A> b b a <B> b` (using `<A>` → `<A> b`)
`<A> b b a <B> b` → `b b b a <B> b` (using `<A>` → `b`) `b b`
`b a <B> b` → `b b b a b b` (using `<B>` → `b`)

So bbbabb is generated by this grammar.

3. bbaaaaabc

Start with `<S>`

`<S>` → `<A> a <B> b`

`<A> a <B> b` → `<A> b a <B> b` (using `<A>` → `<A> b`)

`<A> b a <B> b` → `b b a <B> b` (using `<A>` → `b`)
This string doesn’t match the pattern required by the grammar. This grammar requires exactly one “a” following by one “b” form <B> and ending with one “b” this string has multiple “a” ‘s and ends with “c”.

4. aaaaaa

aaaaaa This string consists only of 'a's but the grammar requires at least one 'b' from <A>, then 'a', then 'b' from <B>, and ending with 'b'.