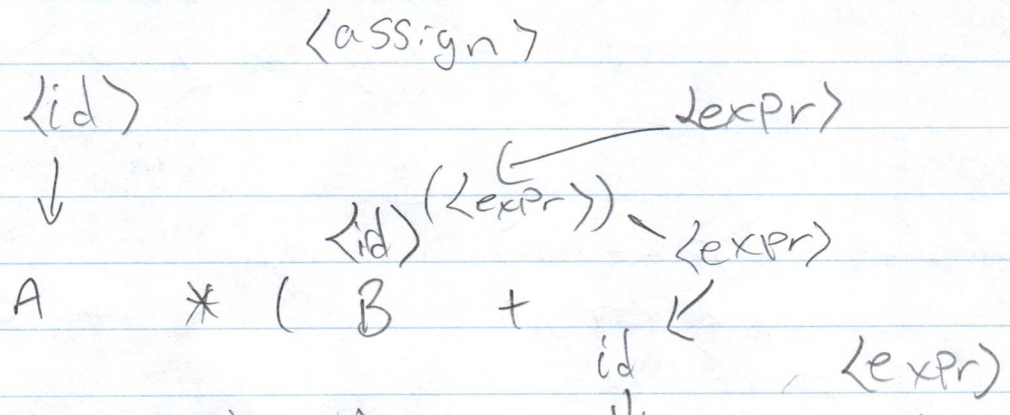
1 Grammar

<assign> ::= <id> = <expr>

<id> ::= A | B | C

<expr> ::= <id> + <expr> | <id> * <expr> | (<expr>) |
<id>

i Parse tree:



<assign> ::= <expr>

::= <id> * <expr>

::= A * <expr>

::= A * (<expr>)

::= A * (<id> + <expr>)

::= A * (B + <expr>)

::= A * (B + (<expr>)) <id>

::= A * (B + (C * <expr>))

::= A * (B + (C * <id>))

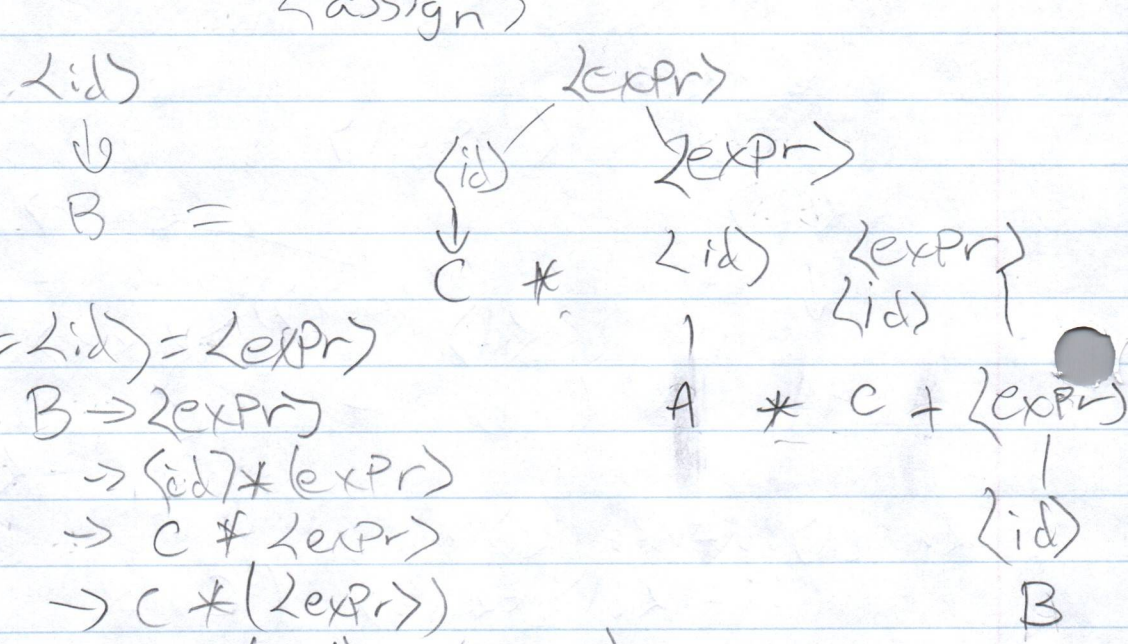
::= A * (B + (C * A))

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

<id> ::= <expr>

<id> ::= A | B | C

<expr> ::= <id> | (<expr>) | <id> * <expr> | <id> + <expr>



<assign> ::= <id> = <expr>

B → <expr>

→ <id> * <expr>

→ C * <expr>

→ C * (<expr>)

→ C * (<id> * <expr>)

→ C * (A * <id> + <expr>)

→ C * (A * C + <id>)

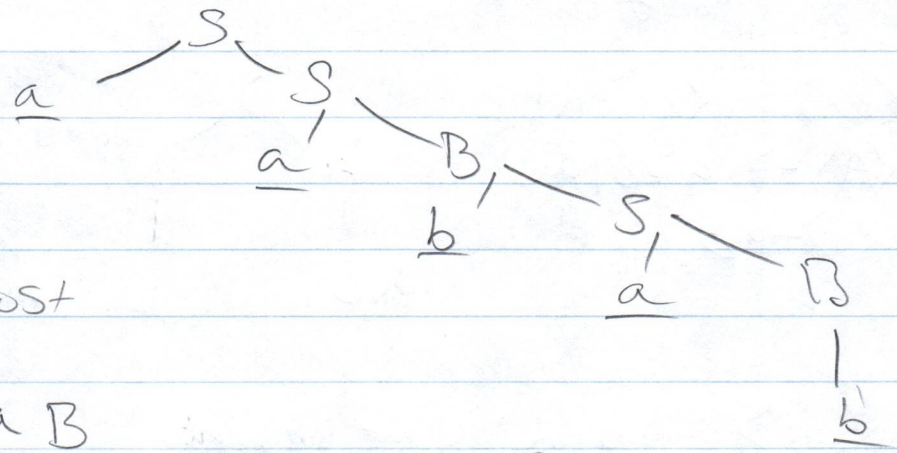
→ C * (A * C + B)

$w = aababab$

$S \to aB \mid bA$

$S \to aS \mid bAA \mid a$

$B \to bS \mid aBB \mid b$



Right most

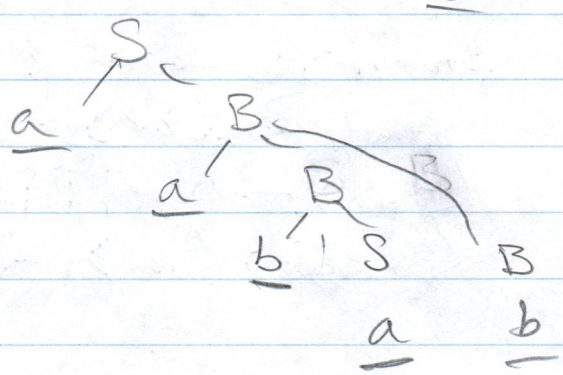
$S \to aS$

$\to aaB$

$\to aababS$

$\to aababaB$

$\to aababab$



left most

$S \to aB$

$\to aaBB$

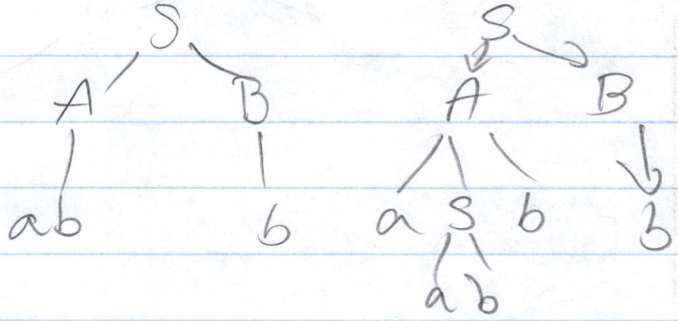
$\to aabSB$

$\to aababB$

$\to aababab$

3 Contains strings with char a & b
≥1 char each with char 'c' being the terminating char

4 <S> → <A><B>
<A> → aSb | ab
<B> → b



5 <Prog> → begin <Stmt arr> end
<Stmt arr> → Stmt {; <Stmt arr>}
<Stmt> → <Var> = <expr>
<Var> → A | B | C
<expr> → <Var> | <Var> (+|−) <Var>

6 S → AB
B → bAB | bA
A → aA | abA