Thomas Nguyen
Coen 171
T R 5:40 – 7:20

Homework # 2

3.2a Write EBNF descriptions for the following: A Java class definition header statement

<head> -> {<modifier>} class <id> [extends name]
[implements <interface_name> , {<interface_name>}]
<modifier> -> public | abstract | final

3.2c Write EBNF descriptions for the following: A C switch statement

<switch_stmt> -> switch (<expr>) {case <literal> : <stmt_list>
{case <literal> : <stmt_list> } [default : <stmt_list>] }

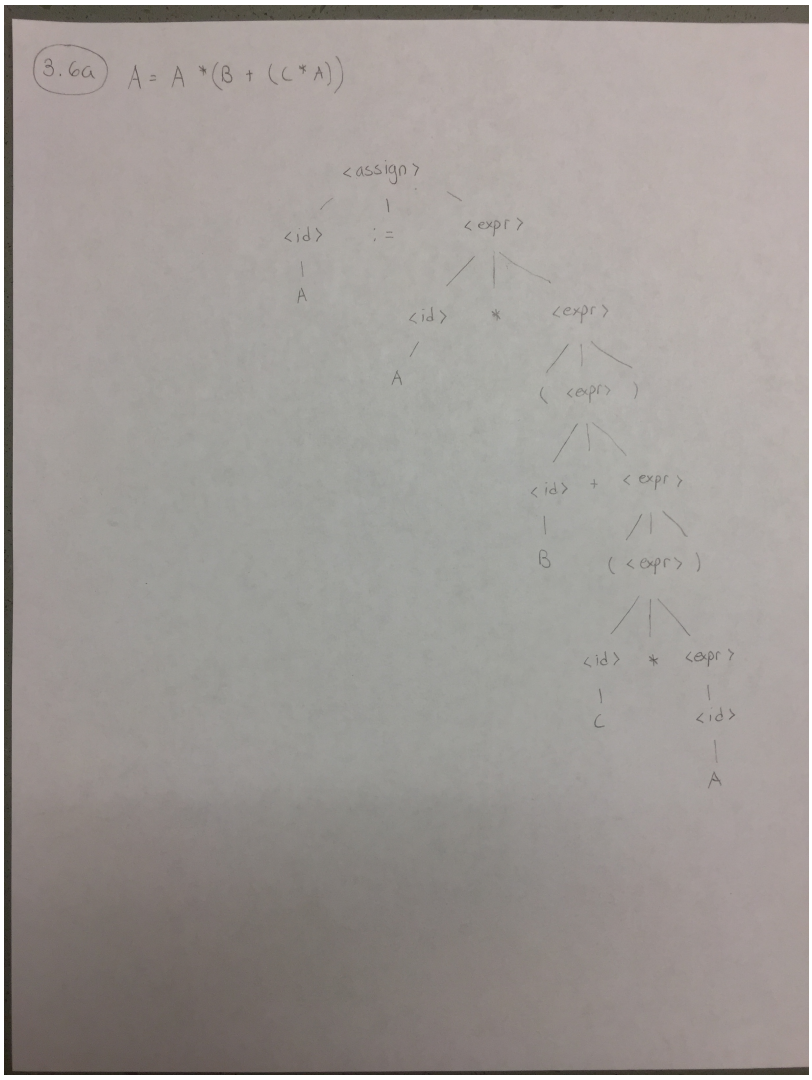
3.3 Rewrite the BNF of Example 3.4 to give + precedence over * and force + to be right associative

<assign> -> <id> = <expr>
<id> -> A | B | C
<expr> -> <expr> * <term>
| <term>
<term> -> <term> + <factor>
| <factor>
<factor> -> (<expr>)
| <id>

3.6 Using the grammar in Example 3.2, show a parse tree and a leftmost derivation for each of the following statements:

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

<assign> => <id> = <expr>
=> A = <expr>
=> A = <id> * <expr>
=> A = A * <expr>
=> A = A * (<expr>)
=> A = A * (<id> + <expr>)
=> A = A * (B + <expr>)
=> A = A * (B + (<expr>))
=> A = A * (B + (<id> * <expr>))
=> A = A * (B + (C * <id>))
=> A = A * (B + (C * A))



B = C * (A * C + B)

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

=> B = <expr>

=> B = <id> * <expr>

=> B = <id> * (<expr>)

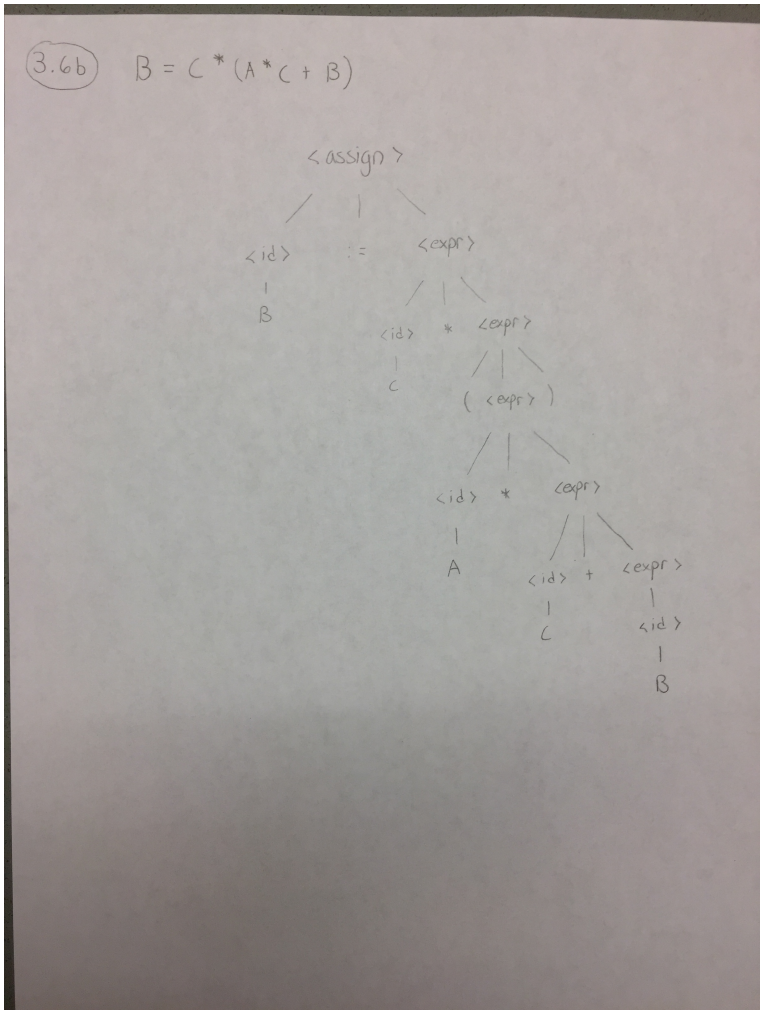
=> B = C * (<id> * <expr>)

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

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

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

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



A = A * (B + (C))

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

=> A = <expr>

=> A = <id> * <expr>

=> A = A * <expr>

=> A = A * (<expr>)

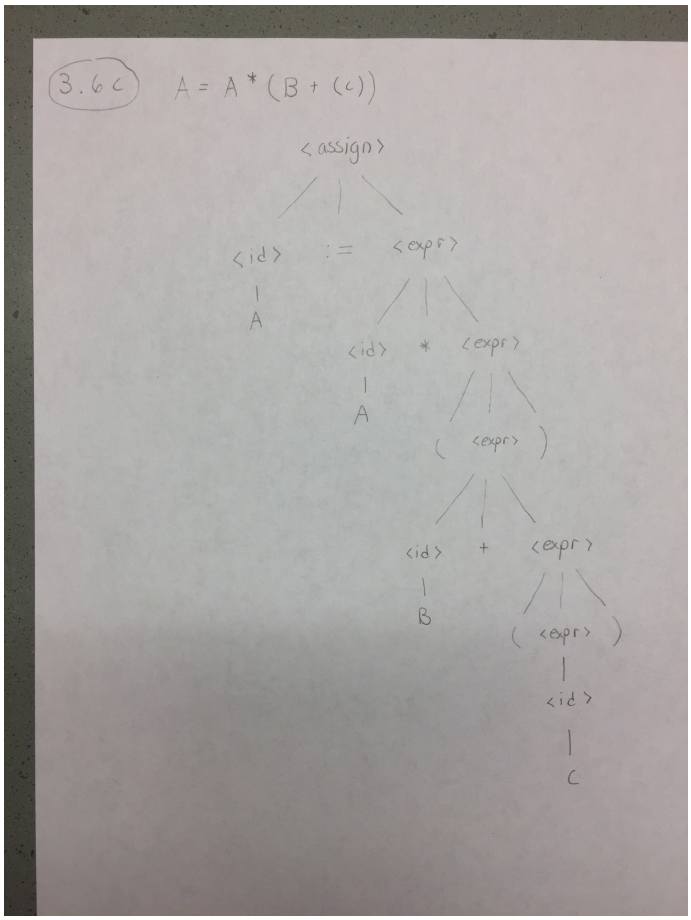
=> A = A * (<id> + <expr>)

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

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

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

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

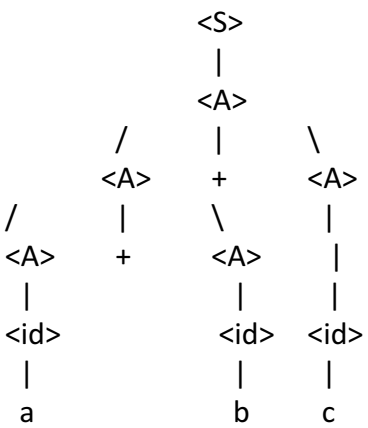


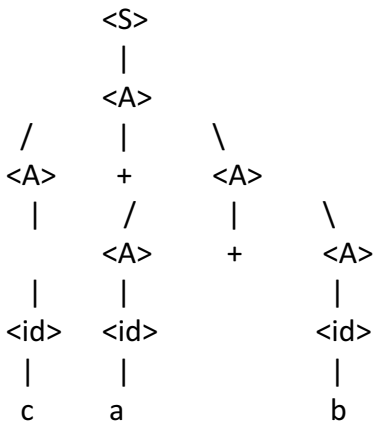
3.8 Prove that the following grammar is ambiguous:

<S> -> <A>

<A> -> <A> + <A> | <id>

<id> -> a | b | c





3.12 Consider the following grammar:
<S> -> a <S> c <B> | <A> | b
<A> -> c <A> | c
<B> -> d | <A>
Which of the following sentences are in the language generated by this grammar
The following two sentences can be generated using the grammar above

A. abcd
E. accc

3.16 Convert the BNF of Example 3.3 to EBNF
<assign> -> <id> = <expr>
<id> -> A | B | C
<expr> -> <expr> (+ | *) <expr>
| (<expr>
| <id>

3.21b Use the virtual machine instructions given in Section 3.5.1.1, give an operational semantic definition of the following: Ada for
i = first
loop: if i < last goto out
…
i = i + 1;

```
        goto loop
out: …
```

3.21c Use the virtual machine instructions given in Section 3.5.1.1, give an operational semantic definition of the following: C++ if-then-else

```
i = expr1
loop: if expr2 == 0 goto out
        …
        expr3
        goto loop
out: …
```

3.22a Write a denotational semantics mapping function for the following statements: Ada for

M_{for}(for var i in init_expr .. final_expr loop L end loop, s)
    if VARMAP(i, s) = undef for var
        then error
        else if M_e(init_expr, s) < M_e(final_expr, s)
            then s
            else M_l(while init_expr - 1 <= final_expr do L)

3.22b Write a denotational semantics mapping function for the following statements: Java do-while

M_{do}(L,s)
M_{while}(repeat L until B)
    if M_b(B, s) = undef
        then error
    else M_{while}(repeat L until B)

3.24 Compute the weakest precondition for each of the following sequences of assignment statements and their post conditions:

A. a < 3 & b < 1
B. a > 3 & b > (1 – a)/2

3.25 Compute the weakest precondition for each of the following selection constructs and their post conditions:

A. a > 0
B. x < 0
C. x > 2