Levi George

02/14/2021

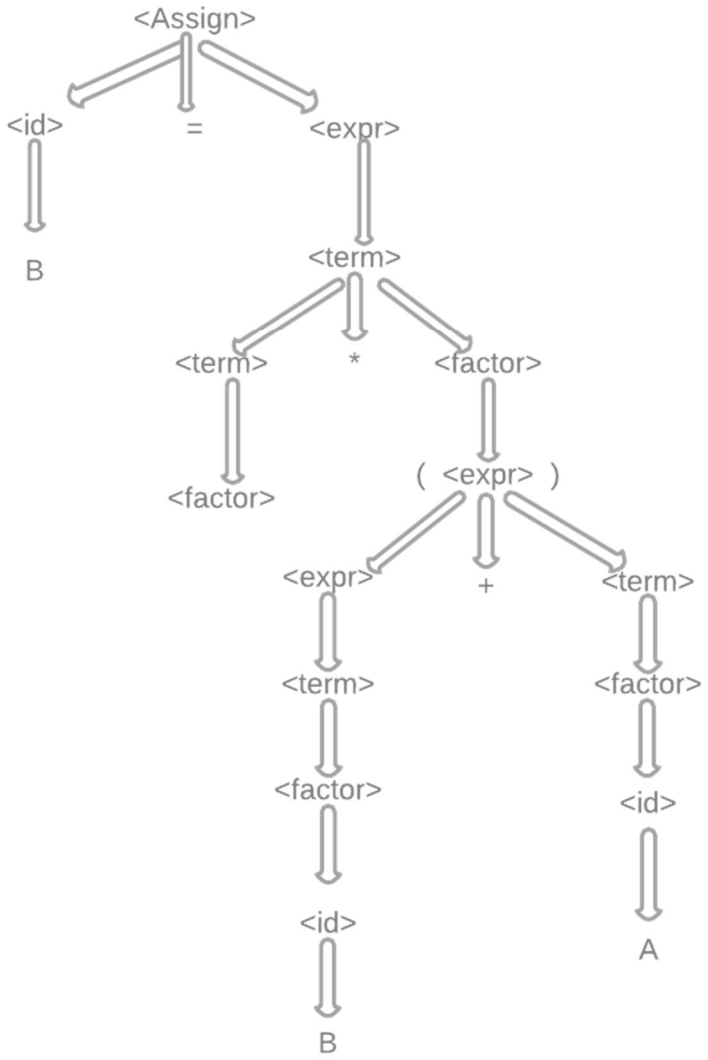
CS 350: Programming Language Design

Prof. Amal Khalifa

Homework 2: Grammars, Parse Trees, and Semantics

Q1.

a. Draw a parse tree for: $(B = A * \big(C * (B + A)\big)$



b. Show the rightmost derivation for the statement in a.

```
<assign> => begin <id> = <expr> end
         => begin <id> = <term> end
         => begin <id> = <term> * <factor> end
         => begin <id> = <term> * ( <expr ) end
         => begin <id> = <term> * ( <term> ) end
         => begin <id> = <term> * ( <term> * <factor> ) end
         => begin <id> = <term> * ( <term> * ( <expr> ) ) end
         => begin <id> = <term> * ( <term> * ( <expr> + <term> ) ) end
         => begin <id> = <term> * ( <term> * ( <expr> + <factor> ) ) end
         => begin <id> = <term> * ( <term> * ( <expr> + <id> ) ) end
         => begin <id> = <term> * ( <term> * ( <expr> + A ) ) end
         => begin <id> = <term> * ( <term> * ( <term> + A ) ) end
         => begin <id> = <term> * ( <term> * ( <factor> + A ) ) end
         => begin <id> = <term> * ( <term> * ( <id> + A ) ) end
         => begin <id> = <term> * ( <term> * ( B + A ) ) end
         => begin <id> = <term> * ( <factor> * ( B + A ) ) end
         => begin <id> = <term> * ( <id> * ( B + A ) ) end
         => begin <id> = <term> * ( C * ( B + A ) ) end
         => begin <id> = <factor> * ( C * ( B + A ) ) end
         => begin <id> = <id> * ( C * ( B + A ) ) end
         => begin <id> = A * ( C * ( B + A ) ) end
         => begin B = A * ( C * ( B + A ) ) end
```

c. Rewrite the grammar to add the ++ and – operators

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

d. Rewrite the grammar in EBNF

```
<assign> => <id> = <expr>
<id>     => A | B | C
<expr>   => <expr> [ ( * | + ) <term> ]
<term>   => (<expr>) | <id>
```

Q2. Consider the Ruby case Statement:

a. Describe the syntax in BNF (**I wasn't sure how far you wanted me to go with the BNF definition, so I just went up to the assignment and identifier definition**)

```
<case_stmt>     => case <term> <case_cond>
<case_cond>     => when <expr> <statement> <case_cond> |
                   when <expr> <statement> <case_end>  |
                   Else <statement> <case_end>
<case_end>      => End

<term>          => <identifier> | <constant>
<statement>     => <expr>
<expr>          => <assign> |
                <expr>, <assign>
<identifier>    => <letter>, <letter_list>
<letter_list>   => A | B | C ...
```

b. Using virtual machine instructions given below, give an operational semantic definition of the statement.

Meaning	Ruby Case Statement
ident = var	case var
if var < ident + 1 goto label	when var < ident + 1 label
if var > ident - 1 goto label	when var > ident -1 label
goto label	else label
	End