Scheme Coding and Grammars

Kong Jimmy Vang
CS 421 (Spring)
2/28/2022

Grammars

Consider an expression language with two binary operators "$" and "#" and two unary prefix operators "*" and "@". It also includes a single alphabetic symbol "X" along with parenthesis "(" and ")". An EBNF expression grammar is given below where the starting non-terminal is <expr> and terminals are highlighted.

```
<expr> ::= <term> { $ <term> }
<term> ::= <factor> { # <factor> }
<factor> ::= [ @ | * ] ( X | ( <expr> ) )
```

1. Give an equivalent unambiguous BNF grammar for this expression language.

```
<expr>   ::= <term> $ <expr>
           | <term>
<term>   ::= <factor> # <term>
           | <factor>
<factor> ::= @ <symbol>
           | * <symbol>
           | <symbol>
<symbol> ::= X
           | ( <expr> )
```

2. Give the associativity of the two binary operators and a precedence table for all four operators in your BNF grammar.

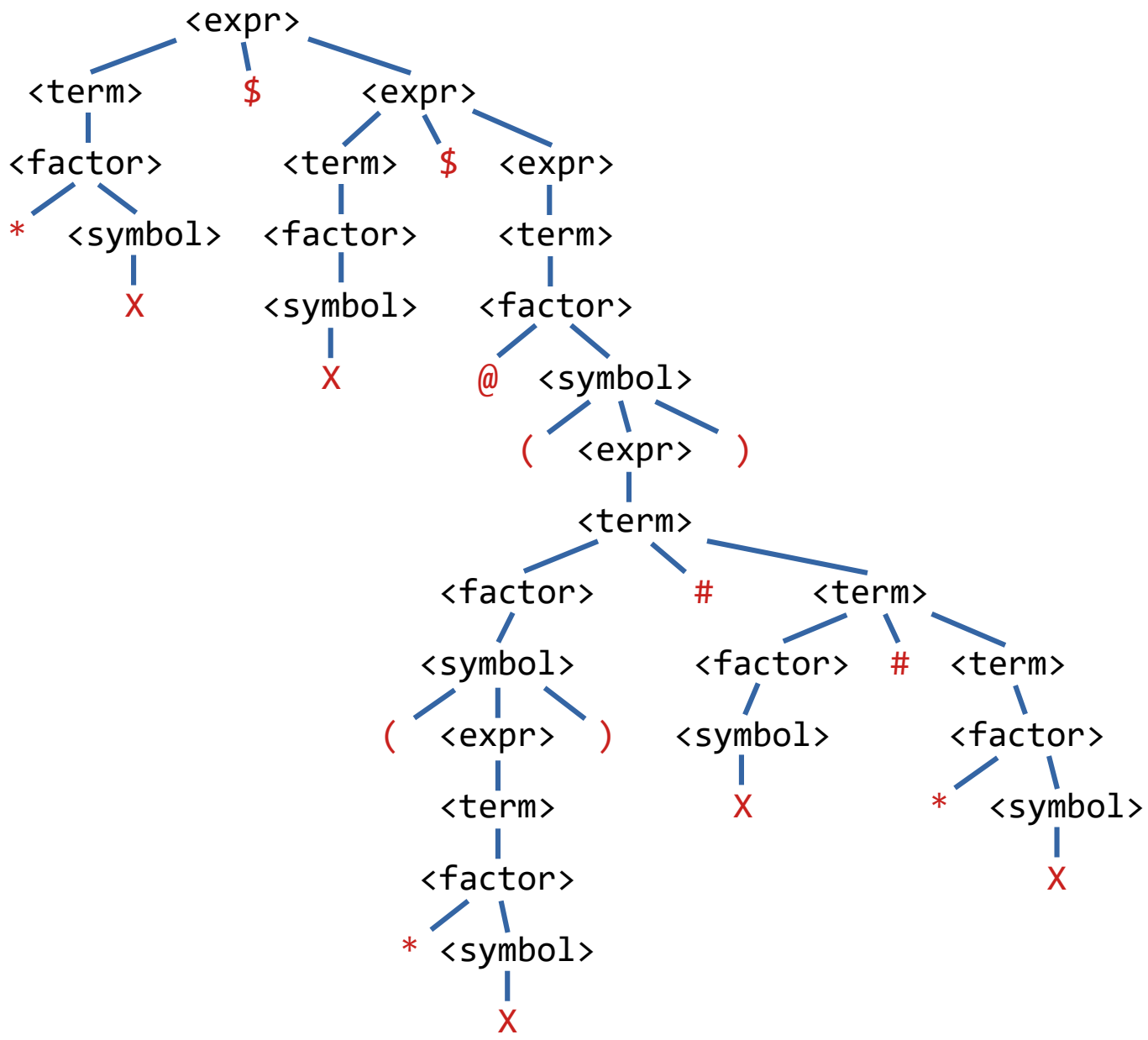
The binary operators $ and # are right associative because productions #1 and #3 are right recursive with themselves.

Precedence Table

Precedence	Operator	Description
3	* @	Unary operator
2	#	Binary operator
1	$	Binary operator

```
#1.  <expr>   ::= <term> $ <expr>
#2.             | <term>
#3.  <term>   ::= <factor> # <term>
#4.             | <factor>
#5.  <factor> ::= @ <symbol>
#6.             | * <symbol>
#7.             | <symbol>
#8.  <symbol> ::= X
#9.             | ( <expr> )
```

3. Using your BNF specification, provide a parse tree for the expression `*X$X$@((*X)#X#*X)` .




```
<expr>   ::= <term> $ <expr>
           | <term>
<term>   ::= <factor> # <term>
           | <factor>
<factor> ::= @ <symbol>
           | * <symbol>
           | <symbol>
<symbol> ::= X
           | ( <expr> )
```

Syntactic Ambiguity

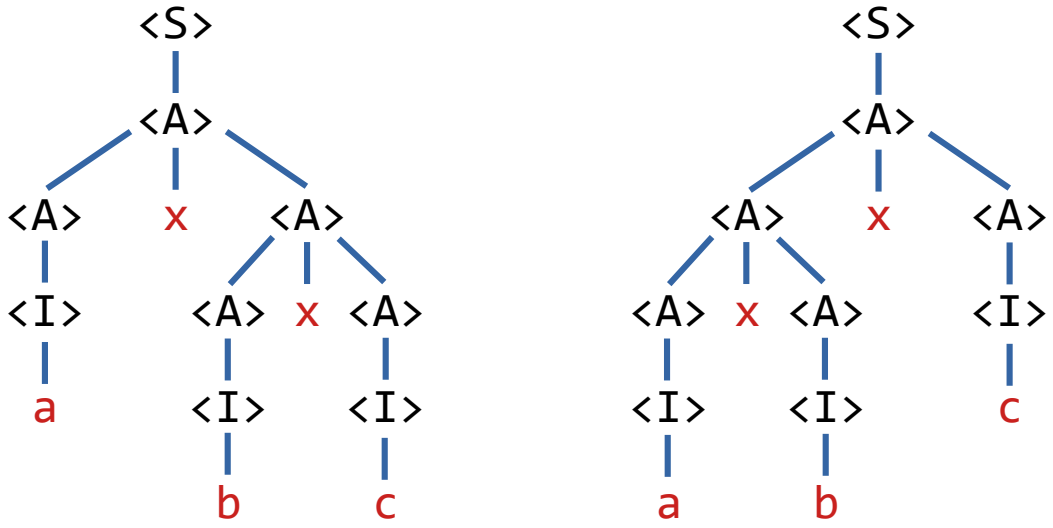
Prove that the following BNF grammar G is ambiguous.

```
N = {<S>, <A>, <I>}
T = {a, b, c, x}
P = {<S> ::= <A>
     <A> ::= <A>x<A>
     <A> ::= <I>
     <I> ::= a
     <I> ::= b
     <I> ::= c}
S = <S>
```

Claim: The BNF grammar G is ambiguous
Proof:

Suppose the BNF grammar **G** is NOT ambiguous, that grammer **G** and its language does NOT contains at least one string with two or more distinct parse trees. {

Let **STR** = "axbxc" {



First, there are two valid parse trees for **STR.**
However, this would mean that there is at least one string with two or more distinct parse trees contained in the BNF grammar **G**.
This contradicts the supposed statement which states that grammar **G** is NOT ambiguous.

}

Therefore, **the BNF grammar G is ambiguous.**

}

□