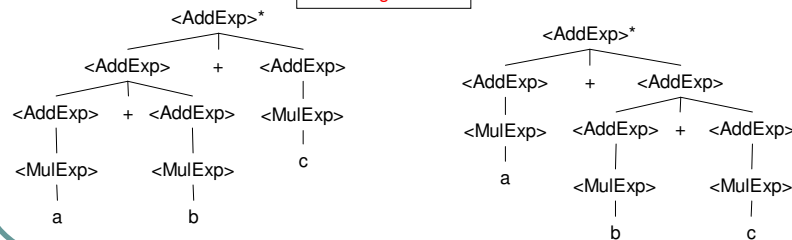


Grammars and Semantics

However, our new grammar still has a problem, consider the sentence $a+b+c$; here we have two possible parse trees:

```
G' : <AddExp> ::= <AddExp> + <AddExp>
      | <MulExp>
    <MulExp> ::= <MulExpr> * <MulExp>
      | a | b | c
```

The grammar G' is
ambiguous



Grammars and Semantics

- Again, our grammar is ambiguous because the computation specified by the sentence $a+b+c$ can be represented by two different parse trees.
- We need more information!
- There is one more algebraic property we have not yet explored – associativity
- Most algebraic operators, including the $+$ operator, are left-associative.
- We can rewrite our grammar to take advantage of this additional information:

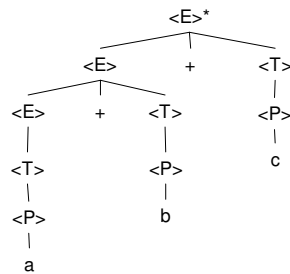
```
G'' : <E>* ::= <E> + <T> | <T>
      <T> ::= <T> * <P> | <P>
      <P> ::= a | b | c
```

Grammars and Semantics

Let's try our sentence $a+b+c$ again with grammar G'' :

$$G'' : \langle E \rangle^* ::= \langle E \rangle + \langle T \rangle \mid \langle T \rangle$$

$$\langle T \rangle ::= \langle T \rangle * \langle P \rangle \mid \langle P \rangle$$

$$\langle P \rangle ::= a \mid b \mid c$$


There is no other way to derive this string from the grammar and thus the grammar is not ambiguous.

Grammars and Semantics

By taking into account associativity and precedence of operators we can construct Grammars that are not ambiguous.

Example: 3.3 d)

$$G : \langle S \rangle^* ::= \langle R \rangle \langle S \rangle \mid \langle O \rangle$$

$$\langle R \rangle ::= (\langle R \rangle) \mid ()$$

$$\langle S \rangle ::= [\langle S \rangle] \mid []$$

$$\langle O \rangle ::= (\langle O \rangle] \mid (\langle I \rangle]$$

$$\langle I \rangle ::=) \langle I \rangle [\mid] [$$

$S = () [] \quad S \in L(G)?$ Is grammar ambiguous?

HW#2:
3.1 a
3.3 b,c
3.4 c
Due Monday 2/6