# Ambiguous Grammars

## Techniques for Removing Ambiguity

Thursday, November 2, 2007
Reading: Stoughton 4.6

**CS235 Languages and Automata**

Department of Computer Science
Wellesley College

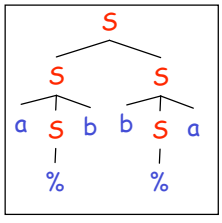
## Some Grammars are Ambiguous

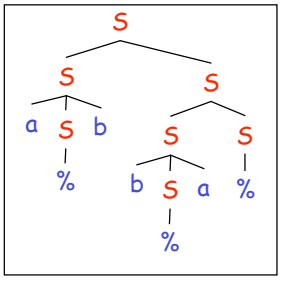
A CFG is **ambiguous** if there is more than one parse tree for a string that it generates.

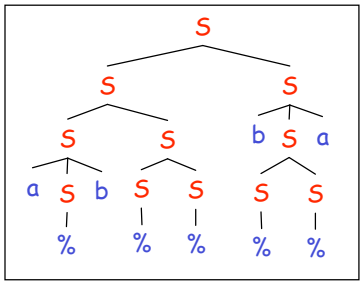
$S \rightarrow \%$
$S \rightarrow SS$
$S \rightarrow aSb$
$S \rightarrow bSa$

This is an example of an ambiguous grammar.
The string abba has an infinite number of parse trees!

Here are a few of them:



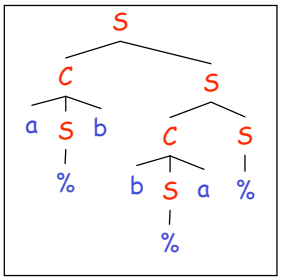



# Removing Ambiguity via Extra Variables

In many cases, ambiguity can be removed by introducing extra variables.

S → %
S → CS
C → aSb
C → bSa

Here, the new variable C stands for a "chunk" of an equal number of as and bs delimited by a…b or by b…a. S stands for a rightward-chained list of such chunks. There is now a unique parse tree:






# Ambiguity Can Affect Meaning

Ambiguity can affect the meaning of a phrase in both natural languages and programming languages.

Here's are some natural language examples:

High school principal

Fruit flies like bananas.

A woman without her man is nothing.

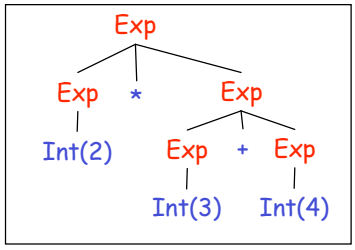
A classic programming languages example is arithmetic expressions:

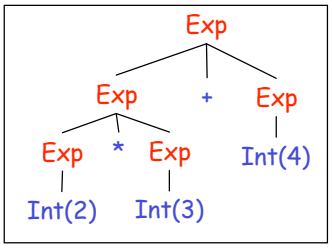
Exp → Exp + Exp
Exp → Exp - Exp
Exp → Exp * Exp
Exp → Exp / Exp
Exp → (Exp)
Exp → INT(i)

# Arithmetic Expressions: Precedence

Exp → Exp + Exp
Exp → Exp - Exp
Exp → Exp * Exp
Exp → Exp / Exp
Exp → (Exp)
Exp → INT(i)

What does 2 * 3 + 4 mean?



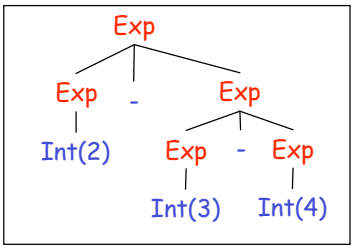



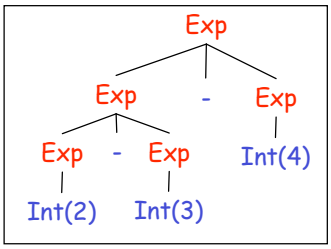



# Arithmetic Expressions: Associativity

Exp → Exp + Exp
Exp → Exp - Exp
Exp → Exp * Exp
Exp → Exp / Exp
Exp → (Exp)
Exp → INT(i)

What does 2 - 3 - 4 mean?



## Precedence Levels

```
Exp → Term | Exp + Exp | Exp - Exp
Term → Factor | Term * Term | Term / Term
Factor → INT(i) | (Exp)
```

Now there is only one parse tree for 2 * 3 + 4. Why? What is it?



## Specifying Left Associativity

```
Exp → Term | Exp + Term | Exp - Term
Term → Factor | Term * Factor | Term / Factor
Factor → INT(i) | (Exp)
```

Now there is only one parse tree for 2 - 3 - 4. Why? What is it?

How would we specify right associativity?

## Another Classic Example: Dangling **Else**

```
Stm → if Exp then Stm else Stm
Stm → if Exp then Stm
Stm → … other rules …
```

There are two parse trees for the following statement.
What are they?

if $Exp_1$ then if $Exp_2$ then $Stm_1$ else $Stm_2$



## Fixing the Dangling **Else**

```
Stm → MaybeElseAfter | NoElseAfter
MaybeElseAfter → if Exp then MaybeElseAfter else MaybeElseAfter
MaybeElseAfter → … other rules …
NoElseAfter → if Exp then Stm
NoElseAfter → if Exp then MaybeElseAfter else NoElseAfter
```

Now there is only one parse tree for the following statement.
What is it?

if $Exp_1$ then if $Exp_2$ then … else …