CMSC 330: Organization of Programming Languages

Context-Free Grammars

Review

- Why should we study CFGs?
- What are the four parts of a CFG?
- How do we tell if a string is accepted by a CFG?
- What's a parse tree?

Review

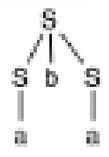
A *sentential form* is a string of terminals and non-terminals produced from the start symbol

Inductively:

- The start symbol
- If αAδ is a sentential form for a grammar, where (α and δ ∈ (N|Σ)*), and A → γ is a production, then αγδ is a sentential form for the grammar
  - In this case, we say that αAδ *derives* αγδ in one step, which is written as αAδ ⇒ αγδ

Leftmost and Rightmost Derivation

- Example: S → a | SbS String: aba



Leftmost Derivation

S ⇒ SbS ⇒ abS ⇒ aba

At every step, apply production to leftmost non-terminal

Rightmost Derivation

S ⇒ SbS ⇒ Sba ⇒ aba

At every step, apply production to rightmost non-terminal

- Both derivations happen to have the same parse tree
- A parse tree has a unique leftmost and a unique rightmost derivation
- Not every string has a unique parse tree
- Parse trees don't show the order productions are applied

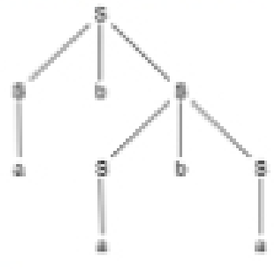
Another Example (cont'd)

S → a | SbS

- Is ababa in this language?

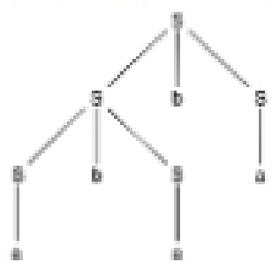
A leftmost derivation

S ⇒ SbS ⇒ abS ⇒
abSbS ⇒ ababS ⇒ ababa



Another leftmost derivation

S ⇒ SbS ⇒ SbSbS ⇒
abSbS ⇒ ababS ⇒ ababa



Ambiguity

- A string is *ambiguous* for a grammar if it has more than one parse tree
  - Equivalent to more than one leftmost (or more than one rightmost) derivation
- A grammar is *ambiguous* if it generates an ambiguous string
  - It's can be hard to see this with manual inspection
- Exercise: can you create an unambiguous grammar for S → a | SbS ?

Are these Grammars Ambiguous?

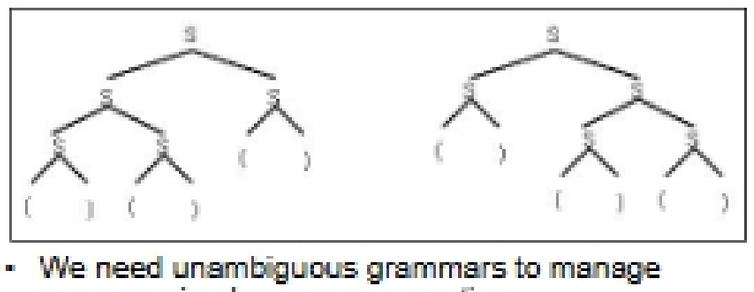
(1) $S \rightarrow aS \mid T$
$T \rightarrow bT \mid U$
$U \rightarrow cU \mid \varepsilon$

(2) $S \rightarrow T \mid T$
$T \rightarrow Tx \mid Tx \mid x \mid x$

(3) $S \rightarrow SS \mid () \mid (S)$

Ambiguity of Grammar (Example 3)

- 2 different parse trees for the same string: ()()()
- 2 distinct leftmost derivations :
  $S \Rightarrow SS \Rightarrow SSS \Rightarrow ()SS \Rightarrow ()()S \Rightarrow ()()()$
  $S \Rightarrow SS \Rightarrow ()S \Rightarrow ()SS \Rightarrow ()()S \Rightarrow ()()()$



- We need unambiguous grammars to manage programming language semantics

More on Leftmost/Rightmost Derivations

- Is the following derivation leftmost or rightmost?
  $S \Rightarrow aS \Rightarrow aT \Rightarrow aU \Rightarrow acU \Rightarrow ac$
  - There's at most one non-terminal in each sentential form, so there's no choice between left or right non-terminals to expand
- How about the following derivation?
  - S › SbS › SbSbS › SbabS › ababS › ababa

Tips for Designing Grammars

1. Use recursive productions to generate an arbitrary number of symbols
   $A \rightarrow xA \mid \varepsilon$ Zero or more x's
   $A \rightarrow yA \mid y$ One or more y's
2. Use separate non-terminals to generate disjoint parts of a language, and then combine in a production
   $G = S \rightarrow AB$
   $A \rightarrow aA \mid \varepsilon$
   $B \rightarrow bB \mid \varepsilon$
   $L(G) = a^*b^*$

Tips for Designing Grammars (cont'd)

3. To generate languages with matching, balanced, or related numbers of symbols, write productions which generate strings from the middle

   $\{a^n b^n \mid n \geq 0\}$ (not a regular language!)
   $S \rightarrow aSb \mid \varepsilon$
   Example: $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$

   $\{a^n b^{2n} \mid n \geq 0\}$
   $S \rightarrow aSbb \mid \varepsilon$

Tips for Designing Grammars (cont'd)

$\{a^n b^m \mid m \geq 2n, n \geq 0\}$
$S \rightarrow aSbb \mid B \mid \varepsilon$
$B \rightarrow bB \mid b$

The following grammar also works:
$S \rightarrow aSbb \mid B$
$B \rightarrow bB \mid \varepsilon$

How about the following?
$S \rightarrow aSbb \mid bS \mid \varepsilon$

Tips for Designing Grammars (cont'd)

$\{a^n b^m a^{n+m} \mid n \geq 0, m \geq 0\}$

Rewrite as $a^n b^m a^m a^n$, which now has matching superscripts (two pairs)

Would this grammar work?

$S \rightarrow aSa \mid B$ Doesn't allow m = 0

$B \rightarrow bBa \mid ba$

Corrected:

$S \rightarrow aSa \mid B$ The outer $a^n a^n$ are generated first, then the inner $b^m a^m$

$B \rightarrow bBa \mid \varepsilon$

Tips for Designing Grammars (cont'd)

4. For a language that's the union of other languages, use separate nonterminals for each part of the union and then combine

$\{ a^n(b^m|c^m) \mid m > n \geq 0\}$

Can be rewritten as

$\{ a^n b^m \mid m > n \geq 0\} \cup$

$\{ a^n c^m \mid m > n \geq 0\}$

Tips for Designing Grammars (cont'd)

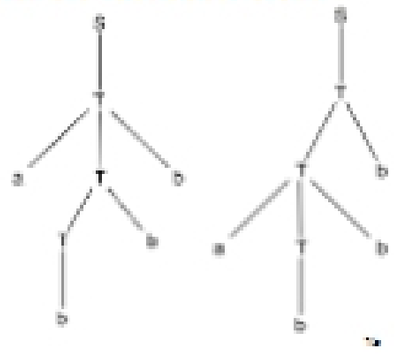
$\{ a^n b^m \mid m > n \geq 0\} \cup \{ a^n c^m \mid m > n \geq 0\}$

$S \rightarrow T \mid U$

$T \rightarrow aTb \mid Tb \mid b$ T generates the first set

$U \rightarrow aUc \mid Uc \mid c$ U generates the second set

- What's the parse tree for string abbb?
  - Ambiguous!



Tips for Designing Grammars (cont'd)

$\{ a^n b^m \mid m > n \geq 0\} \cup \{ a^n c^m \mid m > n \geq 0\}$

Will this fix the ambiguity?

$S \rightarrow T \mid U$

$T \rightarrow aTb \mid bT \mid b$

$U \rightarrow aUc \mid cU \mid c$

- It's not amgiguous, but it can generate invalid strings such as babb

Tips for Designing Grammars (cont'd)

$\{ a^n b^m \mid m > n \geq 0\} \cup \{ a^n c^m \mid m > n \geq 0\}$

Unambiguous version

$S \rightarrow T \mid V$

$T \rightarrow aTb \mid U$

$U \rightarrow Ub \mid b$

$V \rightarrow aVc \mid W$

$W \rightarrow Wc \mid c$

CFGs for Languages

- Recall that our goal is to describe programming languages with CFGs
- We had the following example which describes limited arithmetic expressions

  $E \rightarrow a \mid b \mid c \mid E+E \mid E-E \mid E*E \mid (E)$

- What's wrong with using this grammar?
  - It's ambiguous!