

10/9 Discussion Session

Context-Free Grammars
Exam Review

Recall: Context-Free Grammars

- Regular Expressions are great ... but not good enough to capture a programming language!
 - Example: RE for balanced pairs of parentheses
 - $L = \{ "()", "({})", \dots \}$ (same # of "(" and ")")
- CFGs subsume REs:
 - $(a | b)^*$ same as $S \rightarrow aS | bS |$

Generating Strings from CFGs

- $S \rightarrow aS | bS |$
 - Means “we can replace S with aS, bS, or e”
 - Generate (*derive*) string “abba”
 - $S \Rightarrow aS \Rightarrow abS \Rightarrow abbS \Rightarrow abbaS \Rightarrow abba$
- $S \rightarrow (S) |$
 - Matching pairs of ()
 - $S \Rightarrow (S) \Rightarrow ((S)) \Rightarrow (())$

Describing Grammars

- Example 1:
 - $S \rightarrow abS | a$
- Example 2:
 - $S \rightarrow aSb |$
- Example 3:
 - $S \rightarrow aS | T$
 - $T \rightarrow bT | U$
 - $U \rightarrow cU |$

Describing Grammars

- Example 1: $S \rightarrow abS | a$
 - $(ab)^*a$
- Example 2: $S \rightarrow aSb |$
 - Any # of a’s followed by the same number of b’s
 - $a^n b^n$
 - RE?
- Example 3: $S \rightarrow aS | T, T \rightarrow bT | U, U \rightarrow cU |$
 - Any # of a’s, followed by any # of b’s, followed by any # of c’s
 - $a^* b^* c^*$

Deriving Strings

- Example 1: $S \rightarrow abS | a$
- Example 2: $S \rightarrow aSb |$
- Example 3: $S \rightarrow aS | T, T \rightarrow bT | U, U \rightarrow cU |$

Deriving Strings

- Example 1: $S \rightarrow abS \mid a$
 - a
 - ababa
- Example 2: $S \rightarrow aSb \mid ab$
 - ab
 - aaabbb
- Example 3: $S \rightarrow aS \mid T, T \rightarrow bT \mid U, U \rightarrow cU \mid abc$
 - aabbbbcc
 - bbc

Working Toward PLs

- Basic Arithmetic Expressions
- Boolean Expressions

Working Toward PLs

- Basic Arithmetic Expressions
 - $E \rightarrow a \mid b \mid c \mid E + E \mid E - E \mid E * E \mid (E)$
- Boolean Expressions
 - $S \rightarrow S \text{ and } S \mid S \text{ or } S \mid (S) \mid \text{true} \mid \text{false}$

Properties of Grammars - Ambiguity

- Ambiguity
 - Multiple leftmost or rightmost derivations
- Leftmost/Rightmost Derivation
 - When deriving string, always derive ___-most non-terminal first
- Example: $S \rightarrow S \text{ and } S \mid S \text{ or } S \mid (S) \mid \text{true} \mid \text{false}$
 - Derive ((true and false) or (false and true and true))

Leftmost Derivation

- Leftmost:
 - $S \Rightarrow$
 - $(S) \Rightarrow$
 - $(S \text{ or } S) \Rightarrow$
 - $((S) \text{ or } S) \Rightarrow$
 - $((S \text{ and } S) \text{ or } S) \Rightarrow$
 - $((\text{true and } S) \text{ or } S) \Rightarrow$
 - $((\text{true and false}) \text{ or } S) \Rightarrow$
 - $((\text{true and false}) \text{ or } (S)) \Rightarrow$
 - $((\text{true and false}) \text{ or } (S \text{ and } S)) \Rightarrow$
 - $((\text{true and false}) \text{ or } (S \text{ and } S \text{ and } S)) \Rightarrow$
 - $((\text{true and false}) \text{ or } (\text{false and } S \text{ and } S)) \Rightarrow$
 - $((\text{true and false}) \text{ or } (\text{false and true and } S)) \Rightarrow$
 - $((\text{true and false}) \text{ or } (\text{false and true and true}))$

Properties of Grammars - Ambiguity

- Is our grammar for basic boolean expressions ambiguous?
 - If so, what is an example?

Properties of Grammars - Ambiguity

- Is our grammar for basic boolean expressions ambiguous?
 - Yes; Consider the leftmost derivation of “true and true and true”:
 - $S \Rightarrow S$ and $S \Rightarrow \text{true}$ and $S \Rightarrow \text{true}$ and $S \Rightarrow \text{true}$ and $S \Rightarrow \text{true}$ and $S \Rightarrow \text{true}$ and $S \Rightarrow \text{true}$
 - $S \Rightarrow S$ and $S \Rightarrow S \text{ and } S$ and $S \Rightarrow \text{true}$ and $S \Rightarrow \text{true}$ and $S \Rightarrow \text{true}$ and $S \Rightarrow \text{true}$ and $S \Rightarrow \text{true}$

Designing Grammars

- Tip 1: Use recursive productions to generate an arbitrary number of symbols:
 - $A \rightarrow aA$ | (zero or more As)
 - $B \rightarrow bB$ | b (one or more Bs)
- Tip 2: Use separate nonterminals to consider disjoint parts of a language, then combine with a production:
 - $G \rightarrow AB$
 - $A \rightarrow aA$ | (grammar for a^*bb^*)
 - $B \rightarrow bB$ | b

Designing Grammars

- Tip 3: To generate languages with matching, balanced, or related numbers of symbols, write productions which generate strings from the middle:
 - $S \rightarrow aSb$ ($a^n b^n$)
 - What about $a^n b^{2n}$?

Designing Grammars

- Tip 3: To generate languages with matching, balanced, or related numbers of symbols, write productions which generate strings from the middle:
 - $S \rightarrow aSb$ ($a^n b^n$)
 - What about $a^n b^{2n}$?
 - $S \rightarrow aSbb$
- Try to rewrite “balanced” strings in terms of their power, e.g. $a^n b^n$

Designing Grammars

- Tip 4: For a language that’s a union of other languages, use separate non-terminals for each part of the union, then combine:
 - $\{a^n (b^m | c^m), m > n \geq 0\}$
 - **EQUIVALENT TO**
 - $\{a^n b^m, m > n \geq 0\} \cup \{a^n c^m, m > n \geq 0\}$
- What is the grammar for this expression?

Designing Grammars

- What is the grammar for this expression?
 - $S \rightarrow T | U$
 - $T \rightarrow aTb | Tb | b$ (but it’s **AMBIGUOUS!**)
 - $U \rightarrow aUc | Uc | c$ (consider abbb)
- Fixing ambiguity is a later topic ... any thoughts?

Practice Designing Grammars

1. $a^x b^y$, where $x = y$
2. $a^x b^y$, where $x > y$
3. $a^x b^y$, where $x = 2y$
4. $a^x b^y a^z$, where $z = x+y$
5. All strings of a and b that are palindromes.
6. All strings of a and b that include substring "baa".
7. All strings of a and b with an odd number of a's and an odd number of b's.

Practice Designing Grammars

1. $a^x b^y$, where $x = y$
 1. $S \rightarrow aSb \mid$
2. $a^x b^y$, where $x > y$
 1. $S \rightarrow aL$
 2. $L \rightarrow aL \mid aLb \mid$
3. $a^x b^y$, where $x = 2y$
 1. $S \rightarrow aaSb \mid$

Practice Designing Grammars

4. $a^x b^y a^z$, where $z = x+y$
 1. $S \rightarrow aSa \mid L$
 2. $L \rightarrow bLa \mid$
5. All strings of a and b that are palindromes.
 1. $S \rightarrow aSa \mid bSb \mid L$
 2. $L \rightarrow a \mid b \mid$

Practice Designing Grammars

6. All strings of a and b that include substring "baa".
 1. $S \rightarrow LbaaL$
 2. $L \rightarrow aL \mid bL \mid$ (all strings over a,b)
7. All strings of a and b with an odd number of a's and an odd number of b's.
 1. $S \rightarrow EaEbE \mid EbEaE$
 2. $E \rightarrow EaEaE \mid EbEbE \mid SS \mid$ (even # of a's & b's)