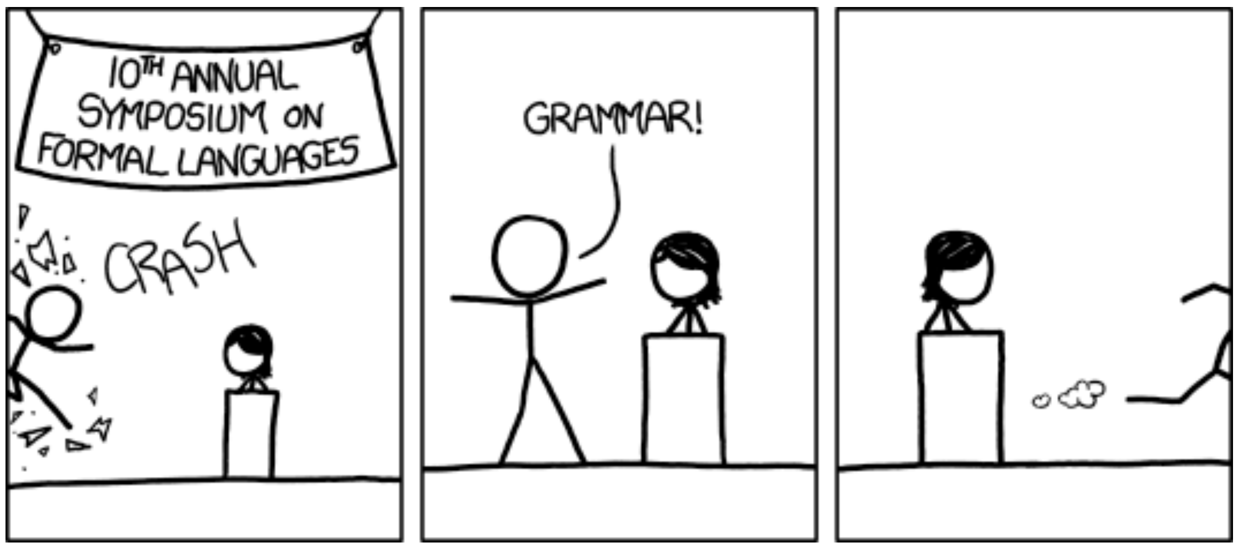
CSE 311: Foundations of Computing

Lecture 21: Context-Free Grammars



[Audience looks around]
“What is going on? There must be some context we’re missing”

Context-Free Grammars

- A Context-Free Grammar (CFG) is given by a finite set of substitution rules involving
  - Alphabet Σ of *terminal symbols* that can't be replaced
  - A finite set **V** of *variables* that can be replaced
  - One variable, usually **S**, is called the *start symbol*

- The substitution rules involving a variable **A**, written as

$$\mathbf{A} \rightarrow w_1 \mid w_2 \mid \cdots \mid w_k$$

  where each w_i is a string of variables and terminals
  - that is $w_i \in (\mathbf{V} \cup \Sigma)^*$

How CFGs generate strings

- Begin with "**S**"

- If there is some variable **A** in the current string, you can replace it by one of the w's in the rules for **A**
  - $\mathbf{A} \rightarrow w_1 \mid w_2 \mid \cdots \mid w_k$
  - Write this as $x\mathbf{A}y \Rightarrow xwy$
  - Repeat until no variables left

- The set of strings the CFG describes are all strings, containing no variables, that can be *generated* in this manner after a finite number of steps

Example Context-Free Grammars

Example: $\mathbf{S} \rightarrow 0\mathbf{S} \mid \mathbf{S}1 \mid \varepsilon$

Example Context-Free Grammars

Example: $\mathbf{S} \rightarrow 0\mathbf{S} \mid \mathbf{S}1 \mid \varepsilon$

0*1*

Example Context-Free Grammars

Example: $\mathbf{S} \rightarrow 0\mathbf{S} \mid \mathbf{S}1 \mid \varepsilon$

0*1*

Example: $\mathbf{S} \rightarrow 0\mathbf{S}0 \mid 1\mathbf{S}1 \mid 0 \mid 1 \mid \varepsilon$

Example Context-Free Grammars

Example: $\mathbf{S} \rightarrow 0\mathbf{S} \mid \mathbf{S}1 \mid \varepsilon$

0^*1^*

Example: $\mathbf{S} \rightarrow 0\mathbf{S}0 \mid 1\mathbf{S}1 \mid 0 \mid 1 \mid \varepsilon$

The set of all binary palindromes

Example Context-Free Grammars

Grammar for $\{0^n 1^n : n \geq 0\}$

(i.e., matching `0*1*` but with same number of 0's and 1's)

Example Context-Free Grammars

Grammar for $\{0^n 1^n : n \geq 0\}$

(i.e., matching `0*1*` but with same number of 0's and 1's)

$$\mathbf{S} \rightarrow 0\mathbf{S}1 \mid \varepsilon$$

Example Context-Free Grammars

Grammar for $\{0^n 1^n : n \geq 0\}$

(i.e., matching 0*1* but with same number of 0's and 1's)

$$\mathbf{S} \rightarrow 0\mathbf{S}1 \mid \varepsilon$$

Grammar for $\{0^n 1^{2n} : n \geq 0\}$

Example Context-Free Grammars

Grammar for $\{0^n 1^n : n \geq 0\}$

(i.e., matching 0*1* but with same number of 0's and 1's)

S $\rightarrow$ 0**S**1 | ε

Grammar for $\{0^n 1^{2n} : n \geq 0\}$

S $\rightarrow$ 0**S**11 | ε

Example Context-Free Grammars

Grammar for $\{0^n 1^n : n \geq 0\}$

(i.e., matching 0*1* but with same number of 0's and 1's)

S → 0**S**1 | ε

Grammar for $\{0^n 1^{n+1} 0 : n \geq 0\}$

Example Context-Free Grammars

Grammar for $\{0^n 1^n : n \geq 0\}$

(i.e., matching 0*1* but with same number of 0's and 1's)

$$\mathbf{S} \rightarrow 0\mathbf{S}1 \mid \varepsilon$$

Grammar for $\{0^n 1^{n+1} 0 : n \geq 0\}$

$$\mathbf{S} \rightarrow \mathrm{A}10$$

$$\mathbf{A} \rightarrow 0\mathbf{A}1 \mid \varepsilon$$

Example Context-Free Grammars

Example: $\mathbf{S} \rightarrow \mathbf{(S)} \mid \mathbf{SS} \mid \varepsilon$

Example Context-Free Grammars

Example: $S \rightarrow (S) \mid SS \mid \varepsilon$

The set of all strings of matched parentheses

Example Context-Free Grammars

Binary strings with equal numbers of 0s and 1s

(not just 0^n1^n, also 0101, 0110, etc.)

$$S \rightarrow SS \mid 0S1 \mid 1S0 \mid \varepsilon$$

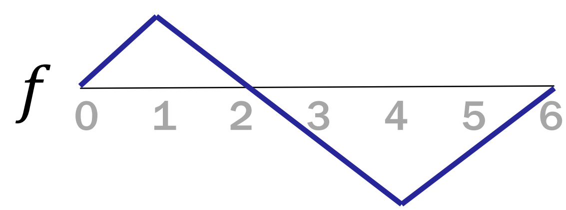
Example Context-Free Grammars

Binary strings with equal numbers of 0s and 1s
(not just 0^n1^n, also 0101, 0110, etc.)

$$S \rightarrow SS \mid 0S1 \mid 1S0 \mid \varepsilon$$

Let $x \in \{0,1\}^*$. Define $f_x(k)$ to be #0s – #1s in the first k characters of x.

E.g., for x = 011100



Example Context-Free Grammars

Binary strings with equal numbers of 0s and 1s
(not just 0^n1^n, also 0101, 0110, etc.)

S → SS | 0S1 | 1S0 | ε

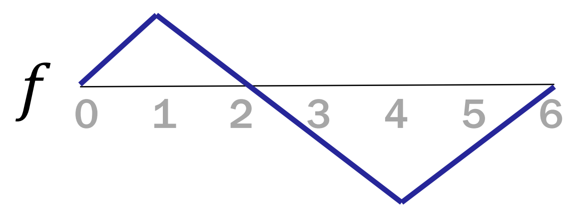
Let $x \in \{0,1\}^*$. Define $f_x(k)$ to be #0s – #1s in the first k characters of x.

If k-th character is 0, then $f_x(k) = f_x(k-1) + 1$
If k-th character is 1, then $f_x(k) = f_x(k-1) - 1$

Example Context-Free Grammars

Let $x \in (0 \cup 1)^*$. Define $f_x(k)$ to be the number 0s minus the number of 1s in the k characters of x.

E.g., for x = 011100



$f_x(k) = 0$ when first k characters have #0s = #1s

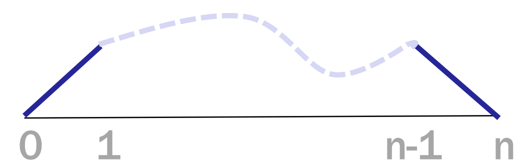
– starts out at 0 $\qquad f_x(0) = 0$

– ends at 0 $\qquad f_x(n) = 0$

Example Context-Free Grammars

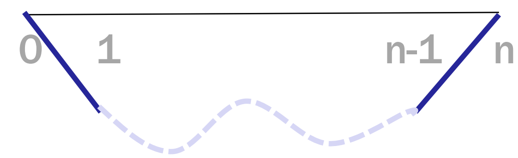
Three possibilities for $f_x(\mathrm{k})$ for $k \in \{1, \dots, n-1\}$

- $f_x(k) > 0$ for all such k



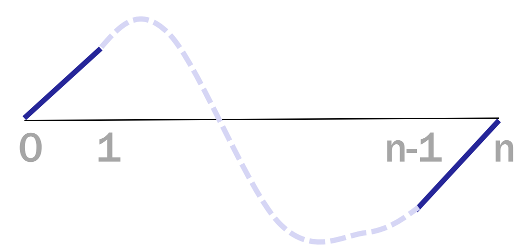
S → 0S1

- $f_x(k) < 0$ for all such k



S → 1S0

- $f_x(k) = 0$ for some such k



S → SS

Simple Arithmetic Expressions

$$E \rightarrow E+E \mid E*E \mid (E) \mid x \mid y \mid z \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Generate $(2*x) + y$

Simple Arithmetic Expressions

$E \rightarrow E{+}E \mid E{*}E \mid (E) \mid x \mid y \mid z \mid 0 \mid 1 \mid 2 \mid 3 \mid 4$
$\mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Generate $(2*x) + y$

$E \Rightarrow E{+}E \Rightarrow (E){+}E \Rightarrow (E*E){+}E \Rightarrow (2*E){+}E \Rightarrow (2*x){+}E \Rightarrow (2*x){+}y$

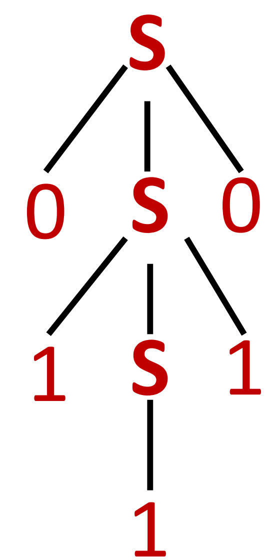
Parse Trees

Suppose that grammar G generates a string x

- A *parse tree* of x for G has
  - Root labeled S (start symbol of G)
  - The children of any node labeled A are labeled by symbols of w left-to-right for some rule $A \rightarrow w$
  - The symbols of x label the leaves ordered left-to-right

$$S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon$$

Parse tree of 01110



Two ways to Define Binary Palindromes

Recursively-Defined Set

Basis:

ε is a palindrome

any $a \in \Sigma$ is a palindrome

Recursive step:

If p is a palindrome,

then apa is a palindrome for every $a \in \Sigma$

Grammar

$\mathbf{S} \to 0\mathbf{S}0 \mid 1\mathbf{S}1 \mid 0 \mid 1 \mid \varepsilon$

CFGs and recursively-defined sets of strings

- A CFG with the start symbol **S** as its *only* variable recursively defines the set of strings of terminals that **S** can generate

- A CFG with more than one variable is a simultaneous recursive definition of the sets of strings generated by *each* of its variables
  - sometimes necessary to use more than one

CFGs and Regular Expressions

Theorem: For any set of strings (language) A described by a regular expression, there is a CFG that recognizes A.

Proof idea:

P(A) is “A is recognized by some CFG”

Structural induction based on the recursive definition of regular expressions…

Regular Expressions over Σ

- **Basis:**
  - ε is a regular expression
  - ***a*** is a regular expression for any $a \in \Sigma$
- **Recursive step:**
  - If **A** and **B** are regular expressions then so are:

    A $\cup$ **B**

    AB

    A*

CFGs are more general than REs

- CFG to match RE **ε**

  S $\rightarrow \varepsilon$

- CFG to match RE ***a*** (for any $a \in \Sigma$)

  S $\rightarrow$ a

CFGs are more general than REs

Suppose CFG with start symbol $\mathbf{S_1}$ matches RE **A**
CFG with start symbol $\mathbf{S_2}$ matches RE **B**

- CFG to match RE $\mathbf{A} \cup \mathbf{B}$

$\mathbf{S} \rightarrow \mathbf{S_1} \mid \mathbf{S_2}$ + rules from original CFGs

- CFG to match RE **AB**

$\mathbf{S} \rightarrow \mathbf{S}_1\, \mathbf{S}_2$ + rules from original CFGs

CFGs are more general than REs

Suppose CFG with start symbol $\mathbf{S_1}$ matches RE $\mathbf{A}$

- CFG to match RE $\mathbf{A}^*$ (= $\varepsilon \cup \mathbf{A} \cup \mathbf{AA} \cup \mathbf{AAA} \cup \ldots$)

  $\mathbf{S} \rightarrow \mathbf{S_1}\,\mathbf{S} \mid \varepsilon$ + rules from CFG with $\mathbf{S_1}$