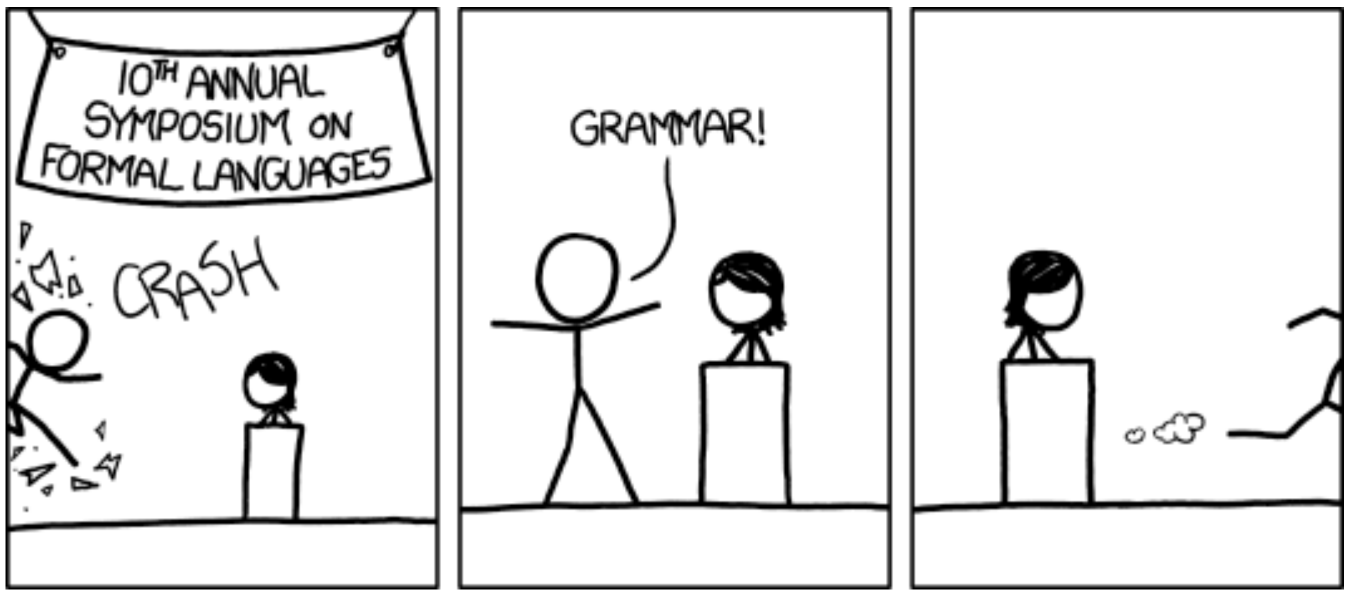
CSE 311: Foundations of Computing

Lecture 19: Regular Expressions & Context-Free Grammars



[Audience looks around]
"What is going on? There must be some context we're missing"

Review: Languages

- Subsets of strings are called *languages*
- Examples:
  - $\Sigma^* =$ All strings over alphabet Σ
  - Palindromes over Σ
  - Binary strings that don't have a 0 after a 1
  - Binary strings with an equal # of 0's and 1's
  - Legal variable names. keywords in Java/C/C++
  - Syntactically correct Java/C/C++ programs
  - English sentences
  - …

Review: each regular expression is a "pattern"

ε matches the **empty string**

a matches the one character string *a*

A $\cup$ **B** matches all strings that either **A** matches or **B** matches (or both)

AB matches all strings that have a first part that **A** matches followed by a second part that **B** matches

A* matches all strings that have any number of strings (even 0) that **A** matches, one after another

– equivalently, $A^* = \varepsilon \cup A \cup AA \cup AAA \cup \ldots$

Examples

$(0 \cup 1)\, 0\, (0 \cup 1)\, 0$

$(0^*1^*)^*$

$(0 \cup 1)^*\, 0110\, (0 \cup 1)^*$

Examples

$(\mathit{0} \cup \mathit{1})\ \mathit{0}\ (\mathit{0} \cup \mathit{1})\ \mathit{0}$

{0000, 0010, 1000, 1010}

$(\mathit{0}^*\mathit{1}^*)^*$

All binary strings

$(\mathit{0} \cup \mathit{1})^*\ \mathit{0110}\ (\mathit{0} \cup \mathit{1})^*$

Binary strings that contain “0110”

Examples

- All binary strings that have an even # of 1's

Examples

- All binary strings that have an even # of 1's

  e.g., 0*(10*10*)*

Examples

- All binary strings that have an even # of 1's

  e.g., 0*(10*10*)*

- All binary strings that *don't* contain 101

Examples

- **All binary strings that have an even # of 1's**

  e.g., $0^*(10^*10^*)^*$

- **All binary strings that *don't* contain 101**

  e.g., $0^*(1 \cup 1000^*)^* (0^* \cup 10^*)$

Limitations of Regular Expressions

- Not all languages can be specified by regular expressions
- Even some easy things like
  - Palindromes
  - Strings with equal number of 0's and 1's
- But also more complicated structures in programming languages
  - Matched parentheses
  - Properly formed arithmetic expressions
  - etc.

Context-Free Grammars

- A Context-Free Grammar (CFG) is given by a finite set of substitution rules involving
  - A finite set **V** of *variables* that can be replaced
  - Alphabet Σ of *terminal symbols* that can't 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 start symbol **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 generates are all strings produced in this way (after a finite number of steps) that have no variables

Example Context-Free Grammars

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

Example Context-Free Grammars

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

The set of all binary palindromes

Example Context-Free Grammars

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

Example Context-Free Grammars

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

0^*1^*

Example Context-Free Grammars

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

(all strings with same # of 0's and 1's with all 0's before 1's)

Example Context-Free Grammars

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

(all strings with same # of 0's and 1's with all 0's before 1's)

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

Example Context-Free Grammars

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

(all strings with same # of 0's and 1's with all 0's before 1's)

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

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

Example Context-Free Grammars

Grammar for $\{0^n1^n : n \geq 0\}$

(all strings with same # of 0's and 1's with all 0's before 1's)

S → 0**S**1 | ε

Grammar for $\{0^n1^{n+1}0 : n \geq 0\}$

S → A 10

A → 0**A**1 | ε

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

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 $(2*x) + y$

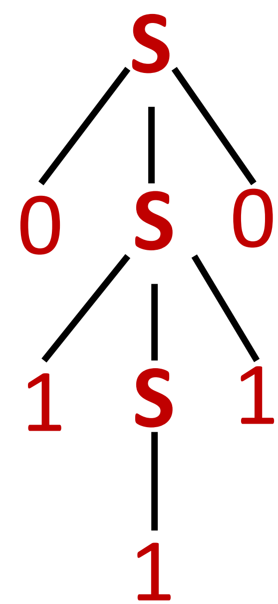
$$E \Rightarrow E+E \Rightarrow (E)+E \Rightarrow (E*E)+E \Rightarrow (2*E)+E \Rightarrow (2*x)+E \Rightarrow (2*x)+y$$

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

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 $x+y*z$ in two ways that give two *different* parse trees

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 x+y∗z in two ways that give two *different* parse trees

$$E \Rightarrow E+E \Rightarrow x+E \Rightarrow x+E*E \Rightarrow x+y*E \Rightarrow x+y*z$$

(multiply y with z and then add to x)

$$E \Rightarrow E*E \Rightarrow E+E*E \Rightarrow x+E*E \Rightarrow x+y*E \Rightarrow x+y*z$$

(add x to y and then multiply by z)