

# Context-Free Grammars

November 14, 2012

CS 81: Computability and Logic

# PROVING LANGUAGES **NON-REGULAR**

## ✓ Myhill-Nerode

- ▶ Find an infinite list of prefixes that disagree on what suffixes should be accepted.

## ✓ Pumping Lemma

1. Define  $p$  to be the “pumping length”
  2. Find a string in the language of length  $\geq p$
  3. Show that no  $xyz$  decomposition with  $|xy| \leq p$  and  $y \neq \epsilon$  is pumpable.
- 
- ✓ Show it can't be accepted by a mechanism more powerful than a finite-state machine.

## NOW WHAT?

Since computers are finite-state machines, no computer can:

✓ Detect arbitrary palindromes

✓ Confirm valid additions

$$1+1=2 \quad 223+103005=103228$$

✓ Check whether parentheses are properly nested

$((()))$       $()()(( ))$

✓ Or solve most other problems.

How, then, can computers be useful?

# CONTEXT-FREE GRAMMARS BY EXAMPLE

- ✓ Language (Terminals):  $\{0, 1\}$
- ✓ Variables (Nonterminals):  $\{S, A, B\}$
- ✓ Start symbol:  $S$ .
- ✓ Context-Free Rules

$$S \rightarrow \varepsilon$$

$$S \rightarrow 0B$$

$$S \rightarrow 1A$$

$$A \rightarrow 0S$$

$$A \rightarrow 1AA$$

$$B \rightarrow 1S$$

$$B \rightarrow 0BB$$

The language of the grammar: *strings of terminals that we can derive from the start symbol.*

# GRAMMARS, FORMALIZED

An *unrestricted* grammar consists of

- ✓ A finite set  $\Sigma$  of terminals
- ✓ A disjoint finite set  $V$  of nonterminals (variables)
  - ▶ The book says that  $V$  includes the terminals, and that  $V \setminus \Sigma$  are the variables.
- ✓ A finite set of rules of the form

LEFT  $\rightarrow$  RIGHT

where LEFT  $\in (V \cup \Sigma)^+$  and RIGHT  $\in (V \cup \Sigma)^*$

- ✓ One designated  $S \in V$ , called the start symbol (start variable)

In a *context-free* grammar, every LEFT is a single variable.

## STRING REWRITING

If we have a rule

$$\text{LEFT} \rightarrow \text{RIGHT}$$

this permits us to replace LEFT by RIGHT *inside* any string:

$$\alpha \text{LEFT} \beta \Rightarrow \alpha \text{RIGHT} \beta$$

where  $\alpha, \beta \in (\Sigma \cup V)^*$

# CHOMSKY HIERARCHY

- ✓ Type 0: Unrestricted Grammars
- ✓ Type 1: Context-Sensitive Grammars
- ✓ Type 2: Context-Free Grammars
- ✓ Type 3: Regular Grammars

## CONTEXT-SENSITIVE EXAMPLE

What is the language of the following *context-sensitive (monotone) grammar*?

$$S \rightarrow abc$$
$$S \rightarrow aSQ$$
$$bQc \rightarrow bbcc$$
$$cQ \rightarrow Qc$$

## STRINGS WITH COMPOSITE (NON-PRIME) LENGTHS

 $S \rightarrow HaC$  $C \rightarrow aC$  $C \rightarrow aT$  $T \rightarrow ZU$  $AZ \rightarrow ZA$  $aZ \rightarrow ZAa$  $HZ \rightarrow HY$  $YA \rightarrow AY$  $Ya \rightarrow aY$  $YU \rightarrow T$  $T \rightarrow X$  $aX \rightarrow Xaa$  $AX \rightarrow Xa$  $HX \rightarrow \varepsilon$

## THE LANGUAGE OF A GRAMMAR

Given a grammar  $G$ , we define

$$L(G) := \{ w \in \Sigma^* \mid S \Rightarrow^* w \}$$

Note:  $L(G_1) = L(G_2)$  does not imply  $G_1 = G_2$ .

A language  $L$  is said to be context-free if it is the language of some context-free grammar.

## EXAMPLES: CFG CONSTRUCTION

Find CFGs for the following languages ( $\Sigma = \{a, b\}$ ):

1.  $\{w \in \Sigma^* \mid w \text{ has an odd length and } a \text{ in the middle}\}$
2.  $\{w \in \Sigma^* \mid w \text{ has an odd length}\}$
3.  $\{w \in \Sigma^* \mid w \text{ a palindrome}\}$

# NOTATIONAL VARIATIONS

$$\begin{aligned} S \rightarrow & \text{if } E \text{ then } S \text{ else } S \\ & | \text{begin } S \text{ L} \\ & | \text{print } E \end{aligned}$$

`<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`

`<expr> ::= <digit>`  
`| <expr> - <expr>`  
`| ( <expr> - <expr> )`

*pointer:*

*\* type-qualifier-list<sub>opt</sub>*

*\* type-qualifier-list<sub>opt</sub> pointer*

# C99 INTEGER CONSTANTS

(6.4.4.1) *integer-constant*:

*decimal-constant integer-suffix<sub>opt</sub>*  
*octal-constant integer-suffix<sub>opt</sub>*  
*hexadecimal-constant integer-suffix<sub>opt</sub>*

(6.4.4.1) *decimal-constant*:

*nonzero-digit*  
*decimal-constant digit*

(6.4.4.1) *octal-constant*:

**0**  
*octal-constant octal-digit*

(6.4.4.1) *hexadecimal-constant*:

*hexadecimal-prefix hexadecimal-digit*  
*hexadecimal-constant hexadecimal-digit*

(6.4.4.1) *hexadecimal-prefix*: one of

**0x 0X**

(6.4.4.1) *nonzero-digit*: one of

**1 2 3 4 5 6 7 8 9**

(6.4.4.1) *octal-digit*: one of

**0 1 2 3 4 5 6 7**

(6.4.4.1) *hexadecimal-digit*: one of

**0 1 2 3 4 5 6 7 8 9**  
**a b c d e f**  
**A B C D E F**

(6.4.4.1) *integer-suffix*:

*unsigned-suffix long-suffix<sub>opt</sub>*  
*unsigned-suffix long-long-suffix*  
*long-suffix unsigned-suffix<sub>opt</sub>*  
*long-long-suffix unsigned-suffix<sub>opt</sub>*

(6.4.4.1) *unsigned-suffix*: one of

**u U**

(6.4.4.1) *long-suffix*: one of

**l L**

(6.4.4.1) *long-long-suffix*: one of

**ll LL**

# PRODUCTION SEQUENCES

$$E \rightarrow E+E \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Do you agree that  $E \Rightarrow^* 3 + 2 + 1$ ?

$$E \Rightarrow E+E \Rightarrow 3+E \Rightarrow 3+E+E \Rightarrow 3+2+E \Rightarrow 3+2+1$$

$$E \Rightarrow E+E \Rightarrow 3+E \Rightarrow 3+E+E \Rightarrow 3+E+1 \Rightarrow 3+2+1$$

$$E \Rightarrow E+E \Rightarrow E+1 \Rightarrow E+E+1 \Rightarrow 3+E+1 \Rightarrow 3+2+1$$

$$E \Rightarrow E+E \Rightarrow E+1 \Rightarrow E+E+1 \Rightarrow E+2+1 \Rightarrow 3+2+1$$

# PARSE TREES

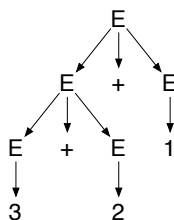
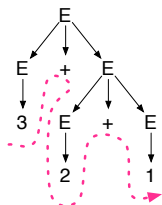
$$E \rightarrow E+E \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

$$E \Rightarrow E+E \Rightarrow 3+E \Rightarrow 3+E+E \Rightarrow 3+2+E \Rightarrow 3+2+1$$

$$E \Rightarrow E+E \Rightarrow 3+E \Rightarrow 3+E+E \Rightarrow 3+E+1 \Rightarrow 3+2+1$$

$$E \Rightarrow E+E \Rightarrow E+1 \Rightarrow E+E+1 \Rightarrow 3+E+1 \Rightarrow 3+2+1$$

$$E \Rightarrow E+E \Rightarrow E+1 \Rightarrow E+E+1 \Rightarrow E+2+1 \Rightarrow 3+2+1$$



# AMBIGUITY

A grammar is *ambiguous* if there is a string with more than one parse tree.

$$E ::= d$$

$$| E - E$$

$$E ::= d$$

$$| d - E$$

$$E ::= d$$

$$| ( E - E )$$

$$E ::= d$$

$$| E E -$$

$$E ::= d$$

$$| E - d$$

## ENCODING PRECEDENCE

$$\begin{aligned} E ::= & d \\ & | E + E \\ & | E - E \\ & | E * E \\ & | E / E \\ & | ( E ) \end{aligned}$$
$$\begin{aligned} E ::= & T \\ & | E + T \\ & | E - T \\ \\ T ::= & F \\ & | T * F \\ & | T / F \end{aligned}$$
$$\begin{aligned} F ::= & d \\ & | ( E ) \end{aligned}$$

## DIGRESSION

There exist *inherently ambiguous* languages that have no unambiguous grammar.

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

Obviously, we tend to avoid when defining programming languages...

But does every language have at least one context-free grammar?

# ANOTHER PUMPING LEMMA

If  $L$  is context-free, then

there exists a number  $p$  such that

For every  $s \in L$  with  $|s| \geq p$

we can decompose  $s$  into  $uvxyz$  where

1.  $vy \neq \varepsilon$
2.  $|vxy| \leq p$
3.  $uv^i xy^i z \in L$  for every  $i \geq 0$ .

$$L := \{ a^n b^n c^n \mid n \geq 0 \}$$

Suppose  $L$  were context-free

- ✓ Let  $p$  be the pumping length
- ✓ Consider, for example,  $s := a^p b^p c^p$ . (Note that  $|s| \geq p$ .)
- ✓ Consider *all* possible decompositions

$$s = uvxyz \quad \text{with} \quad vy \neq \varepsilon \wedge |vxy| \leq p.$$

- ✓ None of them stay in  $L$  when we pump. (Why?)
- ✓ Contradiction.

So  $L$  is not context-free. QED.

$$L := \{ ww \mid w \in \{0, 1\}^* \}$$

Suppose  $L$  were context-free

- ✓ Let  $p$  be the pumping length
- ✓ Consider, for example,  $s := 0^p 1 0^p 1$ . (Note that  $|s| \geq p$ .)
- ✓ Consider, for example,  $s := 0^p 1^p 0^p 1^p$ . (Note that  $|s| \geq p$ .)
- ✓ Consider *all possible decompositions*

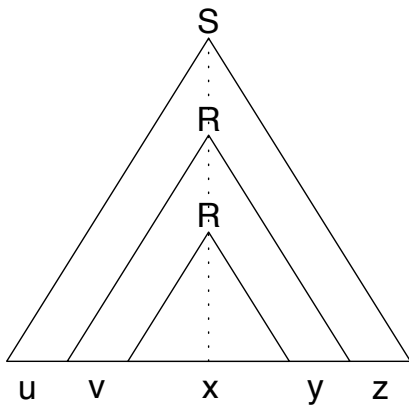
$$s = uvxyz \quad \text{with} \quad vy \neq \varepsilon \wedge |vxy| \leq p.$$

- ✓ Oops...this string *can be pumped!* Tells us nothing.
- ✓ None of them stay in  $L$  when we pump. (Why?)
- ✓ Contradiction.

So  $L$  is not context-free. QED.

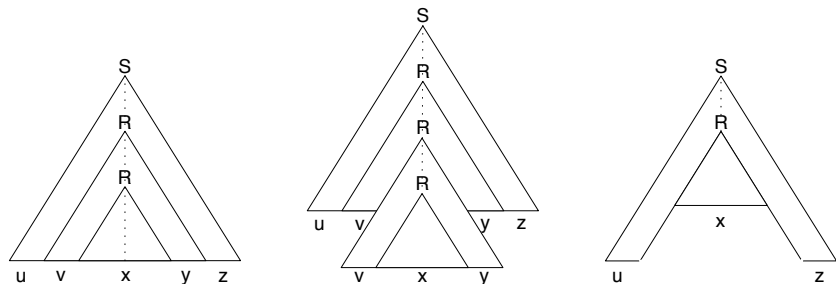
# INTUITION

If our parse tree is tall enough, there must be a path with a repeating nonterminal.



*Consequence?*

## PUMPING



- ✓ How can we be sure that  $v$  and  $y$  aren't empty?
- ✓ How can we ensure  $vxy$  is short enough?

# REGULAR GRAMMARS

A grammar is *regular* if its rules are all of the forms:

$$X \rightarrow a$$

$$X \rightarrow aY$$

$$X \rightarrow \varepsilon$$

# EXAMPLE REGULAR GRAMMAR

$$S \rightarrow 1B$$
$$B \rightarrow 1B$$
$$B \rightarrow 0C$$
$$C \rightarrow 0S$$
$$S \rightarrow 0S$$
$$C \rightarrow 1B$$
$$C \rightarrow$$

How could we turn this into a finite state machine?