Context-Free Grammars

November 9, 2011

CS 81: Computability & 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 $\mathbf{p}$ to be the "pumping length"
  2. Find a string in the language of length $\geqslant \mathbf{p}$
  3. Show that no $\mathbf{xyz}$ decomposition with $|\mathbf{xy}| \leqslant \mathbf{p}$ and $\mathbf{y} \neq \varepsilon$ is pumpable.
- ✓ Show it can't be accepted by a mechanism more powerful than a finite-state machine.

Grammars, Formalized

An *unrestricted* grammar consists of

- ✓ A finite set Σ 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

$$\text{LEFT} \to \text{RIGHT}$$

  where $\text{LEFT} \in (V \cup \Sigma)^+$ and $\text{RIGHT} \in (V \cup \Sigma)^*$
- ✓ One designated $S \in V$, called the start symbol (start variable)

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

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

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.

Context-Free Grammars by Example

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

$$
\begin{aligned}
S &\to \varepsilon \\
S &\to 0B \\
S &\to 1A \\
A &\to 0S \\
A &\to 1AA \\
B &\to 1S \\
B &\to 0BB
\end{aligned}
$$

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

Examples: CFG Construction

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

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

Notational Variations

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

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

pointer:

* *type-qualifier-list*$_{opt}$
* *type-qualifier-list*$_{opt}$ *pointer*

C99 Integer Constants

(6.4.4.1) *integer-constant:*
decimal-constant *integer-suffix*$_{opt}$
octal-constant *integer-suffix*$_{opt}$
hexadecimal-constant *integer-suffix*$_{opt}$

(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*$_{opt}$
unsigned-suffix *long-long-suffix*
long-suffix *unsigned-suffix*$_{opt}$
long-long-suffix *unsigned-suffix*$_{opt}$

(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

$$E \to E{+}E \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

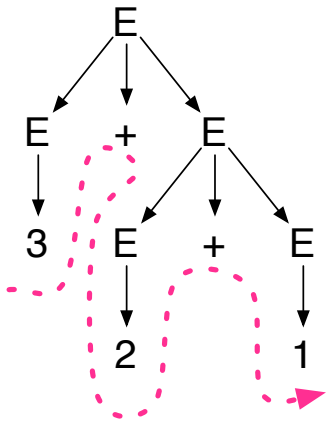
Do you agree that $E \Rightarrow^* 3 + 2 + 1$?

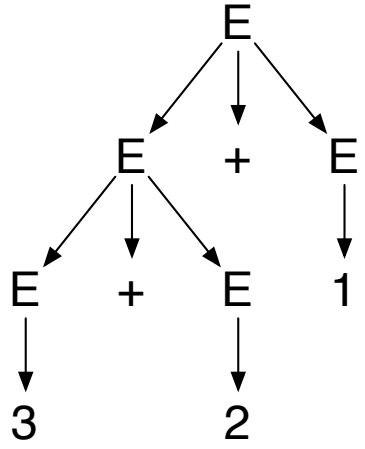
$$E \Rightarrow E{+}E \Rightarrow 3{+}E \Rightarrow 3{+}E{+}E \Rightarrow 3{+}2{+}E \Rightarrow 3{+}2{+}1$$
$$E \Rightarrow E{+}E \Rightarrow 3{+}E \Rightarrow 3{+}E{+}E \Rightarrow 3{+}E{+}1 \Rightarrow 3{+}2{+}1$$
$$E \Rightarrow E{+}E \Rightarrow E{+}1 \Rightarrow E{+}E{+}1 \Rightarrow 3{+}E{+}1 \Rightarrow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

Parse Trees

$$E \to E{+}E \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

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





Ambiguity

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

```
E ::= d
    | E - E
```

```
E ::= d
    | ( E - E )
```

```
E ::= d
    | E - d
```

```
E ::= d
    | d - E
```

```
E ::= d
    | E E -
```

Encoding Precedence

```
E ::= d
    | E + E
    | E - E
    | E * E
    | E / E
    | ( E )
```

```
E ::= T
    | E + T
    | E - T

T ::= F
    | T * F
    | T / F

F ::= d
    | ( E )
```

Digression

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

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

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 $|s| \geqslant p$

we can decompose s into $uvxyz$ where

1. $vy \neq \varepsilon$
2. $|vxy| \leqslant p$
3. $uv^ixy^iz \in L$ for every $i \geqslant 0$.

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

Suppose L were context-free

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

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

- ✓ 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 $|s| \geqslant p$.)
- ✓ Consider, for example, $s := 0^p 1^p 0^p 1^p$. (Note that $|s| \geqslant p$.)
- ✓ Consider *all* possible decompositions

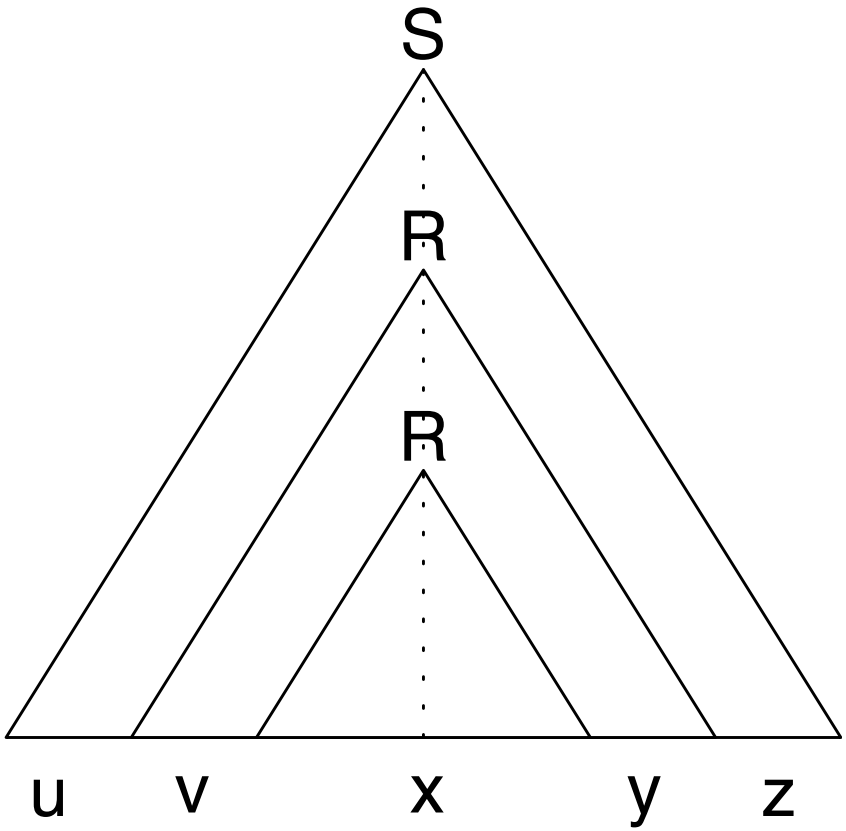
$$s = uvxyz \quad \text{with} \quad vy \neq \varepsilon \ \wedge \ |vxy| \leqslant p.$$

- ✓ 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.

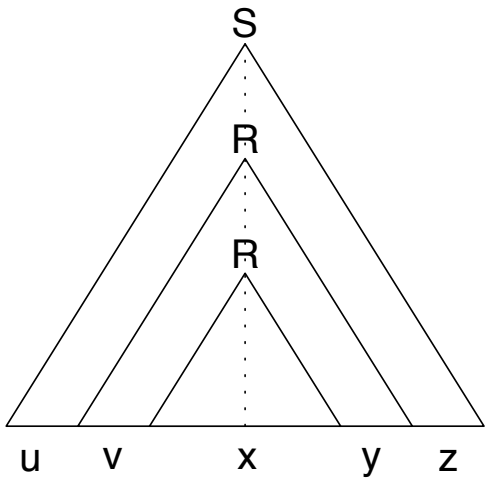
Intutition

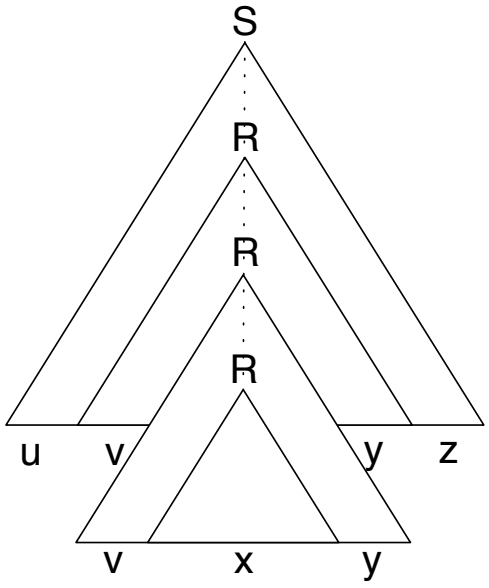
If our parse tree is tall enough, there must be a path with a repeating nonterminal.

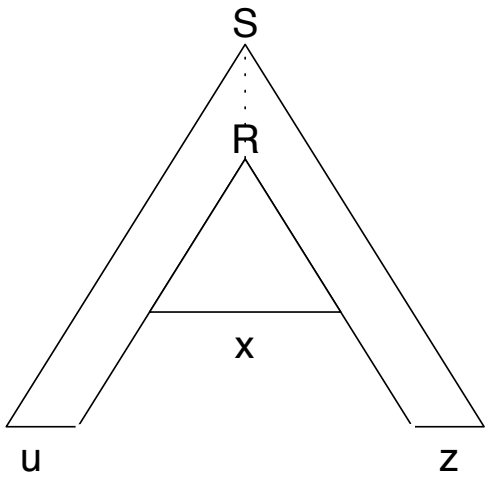


Consequence?

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:

$$
\begin{aligned}
X &\rightarrow a \\
X &\rightarrow aY \\
X &\rightarrow \varepsilon
\end{aligned}
$$

Example Regular Grammar

$$
\begin{aligned}
S &\rightarrow 1B \\
B &\rightarrow 1B \\
B &\rightarrow 0C \\
C &\rightarrow 0S \\
S &\rightarrow 0S \\
C &\rightarrow 1B \\
C &\rightarrow
\end{aligned}
$$

How could we turn this into a finite state machine?