Lecture 7  9/27/18
- Wrap up CFGs, CFLs
  - Ambiguity
  - $0^*1^*2^*$ not a CFL
  - Pumping lemma for CFLs

Next Thurs Oct 4th 7pm-9pm B&H 166
- One handwritten sheet of paper notes allowed
- Review in George + Matt's hours
- Material

HWs
2-4 { DFA, NFA, Regexes
   Properties of regular languages
   Pumping lemma for regular languages
   CFLs, CFGs

Q: Practice exam?
A: Maybe not - But look at exercises in Sipser
   for rough idea of difficulty level

Ambiguity

"A panda eats shoots and leaves"
- This sentence is ambiguous, even though generated
  by correct English grammar

S → <noun phrase> <verb phrase>
<verb> → eats | shoots | leaves | sleeps
Why ambiguous?

Parse Tree

\[
S \\
\text{\langle n.p. \rangle} \quad \text{\langle v.p. \rangle} \\
\quad \text{\langle verb \rangle} \quad \text{\langle n.p. \rangle} \quad \text{\langle n.p. \rangle} \\
\quad \text{\langle verb \rangle} \quad \text{\langle n.p. \rangle} \quad \text{\langle verb \rangle} \\
\quad \text{\langle verb \rangle} \quad \text{\langle verb \rangle} \\
\quad \text{\langle verb \rangle} \\
\quad \text{\langle verb \rangle} \\
\quad \text{\langle verb \rangle} \\
\quad \text{\langle verb \rangle} \\
\text{\langle n.p. \rangle} \quad \text{\langle n.p. \rangle} \quad \text{\langle n.p. \rangle} \\
\text{a panda} \quad \text{eats} \quad \text{shoots} \quad \text{leaves} \\
\text{(PROBABLY CORRECT BASED ON CONTEXT)}
\]

\[\text{I}\]

\[\text{II}\]

\[
S \\
\text{\langle n.p. \rangle} \quad \text{\langle v.p. \rangle} \\
\quad \text{\langle verb \rangle} \quad \text{\langle v.p. \rangle} \\
\quad \text{\langle verb \rangle} \quad \text{\langle v.p. \rangle} \\
\quad \text{\langle verb \rangle} \\
\quad \text{\langle verb \rangle} \\
\quad \text{\langle verb \rangle} \\
\quad \text{\langle verb \rangle} \\
\text{\langle n.p. \rangle} \quad \text{\langle n.p. \rangle} \\
\text{a panda} \quad \text{eats} \quad \text{shoots} \quad \text{leaves} \\
\text{(SILLY BUT VALID PARSE TREE)}
\]

Concept: same grammars inherently ambiguous
Let's Review Def of CFGs

A CFG is a 4-tuple \((V, \Sigma, R, S)\) where

- \(V\): finite set of variables
- \(\Sigma\): finite set of terminals
- \(R\): set of rules

\[ A \rightarrow a_1 \ldots a_k \quad A \in V, \ a_i \in V \cup \Sigma \]

For \(w \in \Sigma^*\), \(w \in L(G) \iff S \xrightarrow{*} w\)

Ambiguity

From Sipser: \(G_4 = (V, \Sigma, R, <EXPR>)\)

- \(V = \{ <EXPR>, <TERM>, <FACTOR> \}\)
- \(\Sigma = \{ a, +, \times, (, ) \}\)

\(R:\)

\[
\begin{align*}
<EXPR> & \rightarrow <EXPR> + <TERM> | <TERM> \\
<TERM> & \rightarrow <TERM> \times <FACTOR> | <FACTOR> \\
<FACTOR> & \rightarrow ( <EXPR> ) | a
\end{align*}
\]

From Sipser: \(G_5 = (V, \Sigma, R, <EXPR>)\)

- \(V = \{ <EXPR> \}\)
- \(\Sigma = \{ a, +, \times, (, ) \}\)

\(R:\)

\[
<EXPR> \rightarrow <EXPR> + <EXPR> | <EXPR> \times <EXPR> | <EXPR> | a
\]
In G5:

```
In G5:

\[
\begin{array}{c}
\text{EXPR} \\
\text{EXPR} + \text{EXPR} \\
\text{EXPR} \times \text{EXPR} \\
2 \\
3 \\
7 \\
\end{array}
\]
```

\[2 + 3 \times 7 = 35\]

Can't tell which parse tree right in G5; (ambiguous)

G4 tries to take care of this?

- Get however many +'s you need, then do **bottom x's**

Claim: G4 unambiguous

**Proof:** Wish to show each variable is unambiguous (over length of string in G4) \(\sim\) strong induction

Base: a, unique parse tree for each var yielding a

**Induction:** Let \( w \) be such that \( \text{EXPR} \Rightarrow w \)

2 cases: \( w \) has + outside parens

- If yes, must have followed **bottom** rule 1
  
  \[ w = w_1 + w_2 \]

- Else either \( w = w_1 \times w_2 \) or followed **bottom** rule 2
  
  and we're good by the Ind Hyp & base case

If 2:

\[ \text{TERM} \]

\[ \text{TERM} \times \text{FACTOR} \]

Summary

- Induction on \(|w| = n\)
  
  \( \text{EXPR} \): unambiguous for length \( n \) by strong induction \( \odot \)

  or \( b/c \) \( \text{TERM} \) is unambiguous for \( n \)

- \( \text{TERM} \): ...

- \( \text{FACTOR} \): ✓
Switching gears: Languages that are not CFLs

In general, parse trees look like this:

\[ S \]

\[ V_1 \rightarrow V_2 \rightarrow V_3 \]

\[ V_S \]

Suppose some variable repeats in path down to a leaf.

Partition based on subtrees of repeated variable:

- \( u \): things to left, not descendants of repeated var
- \( v \): descendants of first \( V_S \) but not second, to left
- \( x \): descendants of both
- \( y \): descendants of first \( V_S \) but not second, to right
- \( z \): things to right, not descendant of repeated var

Treesurgery (pumping tree):

\[ w = uvxyz \]

\[ w' = uvvyyz \]
Pumping lemma for CFLs

For any CFL $A$, there is a pumping length $p$ such that

1. $w \in A$, if $|w| \geq p$, then $w = uvxyz$ such that
2. $|vxy| < p$
3. $|vxy| > 0$
4. $uv^ixyz \in A$ for $i = 0, 1, 2, ...$

Proof:

Let $b$ be the max # of symbols on the right hand side of
a $G$ for $A$
At distance $h$ from root, have $\leq b^h$ nodes
Suppose $w \in A$ such that $|w| \geq b^{h+1}$. Then the parse tree
must have a leaf at depth $\geq h+1$.
If $h = |v|$, must have repeating variable

E.g. let pumping length $p = b^{|v|} + 1$

- Conditions hold $\checkmark$

$A = \{w | w = 0^n 1^n 2^n \}$

Example:

$0^p 1^p 2^p u v x y z$

- If $v$ has not just $0$'s or $1$'s or $2$'s,
pumping gets us outside $A$