**Ambiguity of a Grammar**

Ex. $a + a^*a$

$$
\begin{array}{c}
E \\
E \ A \ E \\
E
\end{array}
$$

Leaves the same: derive same string

Correspond to $a + (a^*a)$ and $(a + a) * a$

May yield different answers if evaluated!

Def. G is ambiguous if there is some string w in $\Sigma^*$ with two different parse trees.

---

**Ambiguity of a Grammar?**

Ex. $S \rightarrow (S) \mid SS \mid \varepsilon$ where $\Sigma$ is $\{\varepsilon, (, )\}$

What is $L(G)$?

Is $G$ ambiguous?

Ex. $E \rightarrow E+E \mid (E) \mid a$

Is $G$ ambiguous?

Ex. $S \rightarrow \text{if } E \text{ then } S \mid \text{if } E \text{ then } S \text{ else } S \mid \text{other}$ where $\Sigma$ is $\{\text{if, then, else, other}\}$

Is $G$ ambiguous?
Disambiguation of CFG

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E → E+E</td>
<td>E × E</td>
</tr>
<tr>
<td>Rewrite:</td>
<td>E → E+T</td>
</tr>
<tr>
<td></td>
<td>T → TxF</td>
</tr>
<tr>
<td></td>
<td>F → (E)</td>
</tr>
</tbody>
</table>

Expression is a list of terms (T) separated by +
Term is a list of factors (F) separated by ×

Ex. S → if E then S | if E then S else S | other where Σ is { if, then, else, other}

Want to match else to closest previous unmatched else (C)
Add 2 new variables, MS (matched) and US (unmatched)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → MS</td>
<td>US</td>
</tr>
<tr>
<td>MS → if E then MS else MS</td>
<td></td>
</tr>
<tr>
<td>US → if E then S</td>
<td>if E then MS else US</td>
</tr>
</tbody>
</table>

Ex. S → (S) | SS | ε | What is the problem? |

Ex. B → B and B | a | b |

Problem: How associate to left or right?
Make right associate:
Regular Languages and CFG’s

Given NFA, can construct CFG with same language

State of NFA \rightarrow Variable

\[ \Sigma \rightarrow \Sigma \]

Edge \quad q_1 \xrightarrow{a} q_2 \rightarrow Q_1 \rightarrow a Q_2 \quad \text{Rule}
q_1 \xrightarrow{\varepsilon} q_2 \rightarrow Q_1 \rightarrow Q_2 \quad \text{Rule}
q_0 \text{ (start)} \rightarrow Q_0 \text{ (start)}
q_2 \text{ is final} \rightarrow Q_2 \rightarrow \varepsilon \quad \text{Rule}

Ex. \quad \begin{array}{c}
\circ\quad q_0 \\
\quad a, b \\
\quad b \\
\quad q_1 \\
\quad b \\
\quad q_2 \\
\quad b \\
\circ\quad q_3 \\
\quad a, b
\end{array}

Chomsky Normal Form

Def. A CFG is in CNF if all rules are of the form

\[ A \rightarrow BC \quad B, C \text{ variables, not } S \]
\[ A \rightarrow a \quad \text{a terminal} \]
\[ S \rightarrow \varepsilon \]

Th. For any CFG G, there is an equivalent CFG G’ in Chomsky normal form.

Proof Idea: Assume no \( \varepsilon \) rules.
1. Add new start symbol \( S_0 \) and new rule \( S_0 \rightarrow S \).
   (So \( S_0 \) not on any RHS)
2. Eliminate unit rules \( A \rightarrow B \).
3. Replace other rules \( A \rightarrow B_1 \quad B_2 \quad \ldots \quad B_m \)
   First, get all variables on RHS, then get in form
Example, Chomsky Normal Form

V \rightarrow bA \quad V \rightarrow aB
A \rightarrow a \quad B \rightarrow b
A \rightarrow aV \quad B \rightarrow bV
A \rightarrow bAA \quad B \rightarrow aBB

1. Add S \rightarrow V, S new start variable.

2. Eliminate unit rule S \rightarrow V:

V appears on LHS of rules V \rightarrow bA, V \rightarrow aB
Add the rules S \rightarrow bA, S \rightarrow aB.
(Only add non-unit rules.)

3. RHS all variables:  For any rule with non-terms on the RHS, replace with variables:

A \rightarrow bAA  replace with  A \rightarrow QAa,  Q \rightarrow b

Example, Chomsky Normal Form

S \rightarrow B1A \quad B1 \rightarrow b \quad S \rightarrow A1B \quad A1 \rightarrow b
A \rightarrow a \quad B \rightarrow b
A \rightarrow A2V \quad A2 \rightarrow a \quad B \rightarrow B2S \quad B2 \rightarrow b
A \rightarrow B3AA \quad B3 \rightarrow b \quad B \rightarrow A3BB \quad A3 \rightarrow a

4. Rewrite long rules:

A \rightarrow B3AA  to  A \rightarrow B3Z  \quad Z \rightarrow AA
B \rightarrow A3BB  to  B \rightarrow A3W  \quad W \rightarrow BB

In step 2: eliminate all unit rules. Determine for which variables you have A \rightarrow B  \quad (how?).  For each rule

B \rightarrow u  \quad |u| > 1

add rule  \quad A \rightarrow u
Why Chomsky Normal Form?

How big can grammar grow?

Step 1: Add new variable, rule

Step 2: One new rule with original start V on LHS

Step 3: Without ε rules, at most 2 extra rules for every occurrence of a nonterminal

Step 4: At most n new rules, for each RHS of n-1 symbols

So polynomial in size of original grammar

Have simple and easily bounded algorithm to determine if string is generated by CNF form

How?

Pushdown Automata

NFA with single, last-in-first-out push down stack

unlimited memory

input from alphabet Σ

from stack alphabet Γ

Stack operations:
Read top
Remove top (Pop)
Write top (Push)

a, x → y Reading input a with x on top, replace x with y

(a, x = ε)
Example PDA

PDA to recognize \( \{ 0^n 1^n | n \geq 0 \} \) (non-regular)

\[
\begin{array}{c}
\text{NFA} \\
\uparrow \\
0 \rightarrow 1 \\
\uparrow \\
\text{Push } $ \text{ on stack.}
\end{array}
\]

\( \Sigma = \{0, 1\} \)

\( \Gamma = \{0, \$\} \)

Start reading input. As read 0's, push them on stack.
When reach 1, for each 1, pop a 0 off stack. If no 0 to pop, reject.
If reach another 0, after you've read a 1, reject.
Accept if finish input, and stack has $ on top.
Finish by popping $ off stack. (Deterministic)