Automata Theory and Formal Grammars: Lecture 7

Non-Context Free Languages

Non-Context Free Languages

Last Time

- Context-free grammars and languages
- Closure properties of CFLs
- Relating regular languages and CFLs

Today

- An introduction to Chomsky Normal Form
- Eliminating ε-productions from CFGs
- Eliminating unit productions from CFGs
- A Pumping Lemma for CFLs
- Non-closure Properties for CFLs

Simplifying CFGs: Chomsky Normal Form

A question we are ultimately interested in: what can and can't we do with CFGs? I.e. are there langauges that are not context-free?

- For regular languages, we showed how FAs can be simplified (minimized).
- This served as basis for proofs of nonregularity.

We will follow a similar line of development for CFLs, but with a twist.

- We will show how CFGs can be "simplified" into *Chomsky Normal Form*.
- We will use this simplification scheme as a basis for establishing that languages are not CFLs (among other things).

Defining Chomsky Normal Form

Definition A CFG $\langle V, \Sigma, S, P\rangle$ is in *Chomsky Normal Form* (CNF) if every production has one of two forms.

- $A \longrightarrow BC$ for $B, C \in V$
- $A \longrightarrow a$ for $a \in \Sigma$

Examples

1. Is $S \longrightarrow \varepsilon \mid 0S1$ in CNF?
   No; both productions violate the two allowed forms.
2. Is $S \longrightarrow SS \mid 0 \mid 1$ in CNF?
   Yes.

What's the Big Deal about CNF?

In an arbitrary CFG it is hard to say whether applying a production leads to "progress" in generating a word.

Example Consider the following CFG G:

$$S \longrightarrow SS \mid 0 \mid 1 \mid \varepsilon$$

and look at this derivation of 01.

$$S \Rightarrow_G SS \Rightarrow_G SSS \Rightarrow_G SSSS \Rightarrow_G SSS \Rightarrow_G SS \Rightarrow_G 0S \Rightarrow_G 01$$

The "intermediate strings" can grow and shrink!

What's the Big Deal about CNF? (cont.)

Applying a production in a CNF grammar always results in "one step of progress": either the number of nonterminals grows by one, or the number of terminals increases by 1.

Example Consider G' given below.

$$S \longrightarrow SS \mid 0 \mid 1$$

The derivation for 01 is:

$$S \Rightarrow_{G'} SS \Rightarrow_{G'} 0S \Rightarrow_{G'} 01.$$

Converting CFGs into CNF

Can every CFG G be converted into a CNF CFG G' so that $\mathcal{L}(G') = \mathcal{L}(G)$?

No! If G' is in CNF, then $\varepsilon \notin \mathcal{L}(G)$!

However, we can get a CNF G' so that $\mathcal{L}(G') = \mathcal{L}(G) - \{\varepsilon\}$.

1. Eliminate *ε-productions* (i.e. productions of form $A \longrightarrow \varepsilon$).
2. Eliminate *unit productions* (i.e. productions of form $A \longrightarrow B$).
3. Eliminate *terminal+ productions* (i.e. productions of form $A \longrightarrow aC$ or $A \longrightarrow aba$).
4. Eliminate *nonbinary productions* (i.e. productions of form $A \longrightarrow ABA$).

Eliminating ε-Productions

CNF grammars contain no ε-productions, and yet arbitrary CFGs may. To convert a CFG to CNF, we therefore need a way of eliminating them.

Of course, CFGs without ε-productions cannot generate the word ε.

Goal Given CFG G, generate CFG G_1 such that:

- G_1 has no ε-productions; and
- $\mathcal{L}(G_1) = \mathcal{L}(G) - \{\varepsilon\}$.

Eliminating ε-Productions: The Naive Approach

Can we just eliminate the ε-productions?

No! What would language of new grammar be if we eliminate the ε-production in the following?

$$S \longrightarrow \varepsilon \mid 0S1$$

Answer $\emptyset$!

- The new grammar would be $S \longrightarrow 0S1$.
- Every derivation looks like: $S \Rightarrow_G 0S1 \Rightarrow_G 00S11 \Rightarrow_G \cdots$.
- That is, can't get rid of S!

So How Can We Eliminate ε-Productions?

ε-productions add "derivational capability" in CFGs by allowing variables to be "eliminated" in a derivation step.

Example Consider the CFG G given as follows.

$$S \longrightarrow \varepsilon \mid 0S1$$

The derivation $S \Rightarrow_G 0S1 \Rightarrow_G 01$ uses the ε-production to get rid of S.

If we want to eliminate ε-productions, we need to add new productions that preserve this derivational capability.

1. Precisely what "derivational capability" do ε-productions provide?
2. How can we recover this capability without ε-productions?

Nullability

Definition Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG. Then $A \in V$ is *nullable* if $A \Rightarrow_G^* \varepsilon$.

E.g. Consider the following CFG.

$$\begin{array}{lcl} S & \longrightarrow & ABCBC \\ A & \longrightarrow & CD \\ B & \longrightarrow & Cb \\ C & \longrightarrow & a \mid \varepsilon \\ D & \longrightarrow & bD \mid \varepsilon \end{array}$$

A is nullable since $A \Rightarrow_G CD \Rightarrow_G D \Rightarrow_G \varepsilon$.

Why are variables nullable? Because of ε-productions! So nullability is the "derivational capability" that ε-productions add to a CFG.

Generating a ε-Production-Free CFGs

Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG, and let $N \subseteq V$ be the set of nullable variables.

If we remove the ε-productions from G, we remove the capability of nullifying variables (i.e. "eliminating" them).

To restore this capability, we need to add productions in which nullable variables are explicitly removed.

Example Consider

$$S \longrightarrow \varepsilon \mid 0S1$$

S is nullable; to eliminate ε-production we should add production $S \longrightarrow 01$. The new grammar:

$$S \longrightarrow 0S1 \mid 01$$

Constructing ε-Free CFGs

Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG, and let $N \subseteq V$ be the set of nullable variables. Consider the following definition of $G_1 = \langle V_1, \Sigma, S_1, P_1 \rangle$.

$$V_1 = V$$
$$S_1 = S$$
$$P = P - \{ A \longrightarrow \varepsilon \mid A \longrightarrow \varepsilon \in P \}$$
$$\cup \{ A \longrightarrow \alpha_0 \cdots \alpha_n \mid \alpha_0 \cdots \alpha_n \neq \varepsilon \wedge \exists A_1, ..., A_n \in N. \; A \longrightarrow \alpha_0 A_1 \alpha_1 ... \alpha_{n-1} A_n \alpha_n \in P \}$$

Huh?

P_1 contains:

- the non-ε-productions in P, together with
- productions obtained by selectively omitting occurrences of nullable variables.
  - $A \longrightarrow \alpha_0 A_1 \alpha_1 ... \alpha_{n-1} A_n \alpha_n$ is a production in G.
  - The A_i are nullable variables.
  - The α_i is the "stuff" in-between the A_i.
  - $A \longrightarrow \alpha_0 \cdots \alpha_n$ is a modified production with the A_i's omitted.

The idea is that in the original grammar, $A \Rightarrow^*_G \alpha_0 \cdots \alpha_n$ by "nullifying" the A_i. In G_1, this capability is realized in a single production.

Calculating the Set of Nullable Variables

To generate G_1, we need to calculate the set $N \subseteq V$ of nullable variables. We can do so by giving a recursive characterization of N.

Define $N(G) \subseteq V$ as follows.

- If $A \longrightarrow \varepsilon$ then $A \in N(G)$.
- If $A \longrightarrow B_1 \cdots B_n$ and $B_1, ..., B_n \in N(G)$ then $A \in N(G)$.

Lemma Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG, and let $A \in V$. Then $A \in N(G)$ if and only if A is nullable.

Proof Use induction!

Example

Consider G given as follows.

$$\begin{array}{lcl} S & \longrightarrow & ABCBC \\ A & \longrightarrow & CD \\ B & \longrightarrow & Cb \\ C & \longrightarrow & a \mid \varepsilon \\ D & \longrightarrow & bD \mid \varepsilon \end{array}$$

First, calculate $N(G)$.

$$\begin{array}{lcl} N(G)_0 & = & \emptyset \\ N(G)_1 & = & \{C, D\} \\ N(G)_2 & = & \{A, C, D\} \\ N(G)_3 & = & N(G)_2 \end{array}$$

Recall G; remember that $N(G) = \{A, C, D\}$.

$$\begin{array}{lll} S & \longrightarrow & ABCBC \\ A & \longrightarrow & CD \\ B & \longrightarrow & Cb \end{array} \qquad \begin{array}{lll} C & \longrightarrow & a \mid \varepsilon \\ D & \longrightarrow & bD \mid \varepsilon \end{array}$$

G_1 is (boxed transitions are new ones):

$$\begin{array}{lll} S & \longrightarrow & ABCBC \mid \boxed{ABCB} \mid \boxed{ABBC} \mid \boxed{ABB} \\ & \mid & \boxed{BCBC} \mid \boxed{BCB} \mid \boxed{BBC} \mid \boxed{BB} \\ A & \longrightarrow & CD \mid \boxed{C} \mid \boxed{D} \\ B & \longrightarrow & Cb \mid \boxed{b} \\ C & \longrightarrow & a \\ D & \longrightarrow & bD \mid \boxed{b} \end{array}$$

Where are we?

So Far

- Simplifying CFGs and Chomsky Normal Form (CNF)
- Eliminating ε-productions from CFGs.

To Do Eliminating:

- unit
- terminal+
- nonbinary productions

from CFGs.

Converting CFGs into Chomsky Normal Form

1. Eliminate *ε-productions* ($A \longrightarrow \varepsilon$).
2. Eliminate *unit productions* ($A \longrightarrow B$).
3. Eliminate *terminal+ productions* ($A \longrightarrow aC$, $A \longrightarrow aba$).
4. Eliminate *nonbinary productions* ($A \longrightarrow ABA$).

Last time we proved the following.

Lemma Let G be a CFG. Then there is a CFG $G1$ containing no ε-productions and such that $\mathcal{L}(G1) = \mathcal{L}(G) - \{\varepsilon\}$.

I.e. we now know how to eliminate ε-productions! What about the others?

Eliminating Unit Productions

Definition A *unit production* has form $A \longrightarrow B$ where $B \in V$.

Like ε productions, they add "derivational capability" to grammars.

Consequently, if we eliminate them we need to "add in" productions that simulate derivations that involved them.

Example Consider G given by:

$$\begin{array}{lll} S & \longrightarrow & A \mid C \\ A & \longrightarrow & aA \mid B \\ B & \longrightarrow & bB \mid b \\ C & \longrightarrow & cC \mid c \end{array}$$

In order to remove $S \longrightarrow A$, need to add e.g. $S \longrightarrow aA$!

But Which Productions Do We Need To Add?

Suppose G is a CFG. Then unit productions allow derivations like this.

$$A \Rightarrow_G A_1 \Rightarrow_G A_2 \Rightarrow_G \cdots \Rightarrow_G A_n \Rightarrow_G \alpha$$

where each $A_i \in V$ is a single variable. If α is not just a single variable, then we should add a production $A \longrightarrow \alpha$. How do we determine these α's?

Definition Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG, with $A \in V$. Then $U(G, A) \subseteq V$ is defined inductively as follows.

- $A \in U(G, A)$.
- If $B \in U(G, A)$ and $B \longrightarrow C \in P$ then $C \in U(G, A)$.

$U(G, A)$ and New Productions

Intuitively, $B \in U(G, A)$ iff $A \Rightarrow_G^* B$ using only unit productions!

Idea In new CFG, we will remove unit productions but add in productions of form $A \longrightarrow \alpha$ for every variable A, where $B \longrightarrow \alpha$ in original CFG and $B \in U(G, A)$!

Example

Let G be given as follows.

$$\begin{array}{rcl} S & \longrightarrow & A \mid C \\ A & \longrightarrow & aA \mid B \\ B & \longrightarrow & bB \mid b \\ C & \longrightarrow & cC \mid c \end{array}$$

Then $U(G, S)$ can be computed as follows.

$$\begin{array}{rcl} U(G,S)_0 & = & \emptyset \\ U(G,S)_1 & = & \{S\} \\ U(G,S)_2 & = & \{S, A, C\} \\ U(G,S)_3 & = & \{S, A, B, C\} \;=\; U(G,S)_4 \end{array}$$

Example (cont.)

$$\begin{array}{rcl} S & \longrightarrow & A \mid C \\ A & \longrightarrow & aA \mid B \\ B & \longrightarrow & bB \mid b \\ C & \longrightarrow & cC \mid c \end{array}$$

We can similarly show that $U(G, A) = \{A, B\}$, $U(G, B) = \{B\}$, and $U(G, C) = \{C\}$. Then the new grammar should be:

$$\begin{array}{rcl} S & \longrightarrow & \boxed{aA} \mid \boxed{bB} \mid \boxed{b} \mid \boxed{cC} \mid \boxed{c} \\ A & \longrightarrow & aA \mid \boxed{bB} \mid \boxed{b} \\ B & \longrightarrow & bB \mid b \\ C & \longrightarrow & cC \mid c \end{array}$$

Formal Construction

Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG. Then we define $G_2 = \langle V, \Sigma, S, P_2 \rangle$ as follows.

$$P_2 \;=\; \{\, A \longrightarrow \alpha \mid \exists B \in U(G, A), \alpha.\ B \longrightarrow \alpha \in P \land \alpha \notin V \,\}$$

Fact Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG without ε productions, and let G_2 be defined as above. Then the following hold.

1. G_2 contains no ε productions.
2. G_2 contains no unit productions.
3. $\mathcal{L}(G_2) = \mathcal{L}(G) - \{\varepsilon\}$.

Eliminating Terminal+ Productions

Definition A production $A \longrightarrow \alpha$ is *terminal+* if $|\alpha| \geq 2$ and α contains at least one terminal.

Examples

- $A \longrightarrow Ca$
- $A \longrightarrow aba$

Eliminating these is fairly simple:

- Introduce a new variable X_a for each terminal $a \in \Sigma$.
- Add productions $X_a \longrightarrow a$.
- In each terminal+ production, replace terminals a by variables X_a.

Example

Let G be given by:

$$S \;\longrightarrow\; aSb \mid aS \mid Sb \mid a \mid b$$

Then G_3 is:

$$\begin{array}{lcl} S & \longrightarrow & X_aSX_b \mid X_aS \mid SX_b \mid a \mid b \\ X_a & \longrightarrow & a \\ X_b & \longrightarrow & b \end{array}$$

Formal Construction

Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG. Then we define $G_3 = \langle V_3, \Sigma, S, P_3 \rangle$ as follows.

$V_3 = V \cup \{\, X_a \mid a \in \Sigma \,\}$, where $X_a \notin V \cup \Sigma$

$P_3 = \{\, A \longrightarrow \alpha' \mid A \longrightarrow \alpha \in P$

$\land\ \alpha'$ is α with a replaced by X_a if $A \longrightarrow \alpha$ is terminal+$\}$

$\cup \{\, X_a \longrightarrow a \mid a \in \Sigma \,\}$

Lemma Let G be a CFG without ε- or unit-productions, and let G_3 be constructed as above. Then the following are true.

1. G_3 contains no ε or unit productions.
2. G_3 contains no terminal+ productions.
3. $\mathcal{L}(G_3) = \mathcal{L}(G) - \{\varepsilon\}$.

Eliminating Nonbinary Productions

Definition A production $A \longrightarrow \alpha$ is *nonbinary* if $|\alpha| \geq 3$.

Example $A \longrightarrow BAB$

How do we eliminate these?

- For each such production $p = A \longrightarrow A_1 A_2 \cdots A_n$ and $n \geq 3$, we will introduce new variables $X_{p,2}, \ldots X_{p,n-1}$.
- Replace $A \longrightarrow A_1 A_2 \cdots A_n$ by a collection of productions:

$$\begin{array}{rcl} A & \longrightarrow & A_1 X_{p,2} \\ X_{p,2} & \longrightarrow & A_2 X_{p,3} \\ & \vdots & \\ X_{p,n-1} & \longrightarrow & A_{n-1} A_n \end{array}$$

Explaining the Idea

Suppose we have a production $A \longrightarrow BCCD$. The construction would replace it with the following.

$$\begin{array}{rcl} A & \longrightarrow & BX_{p,2} \\ X_{p,2} & \longrightarrow & CX_{p,3} \\ X_{p,3} & \longrightarrow & CD \end{array}$$

In the original CFG, $A \Rightarrow_G^* BCCD$ in one step.

In the new CFG it takes three steps:
$A \Rightarrow_{G_4} BX_{p,2} \Rightarrow_{G_4} BCX_{p,3} \Rightarrow_{G_4} BCCD$.

Example

Let G be:

$$\begin{array}{rcl} S & \longrightarrow & X_a S X_b \mid X_a S \mid S X_b \mid a \mid b \\ X_a & \longrightarrow & a \\ X_b & \longrightarrow & b \end{array}$$

Then G_4 is:

$$\begin{array}{rcl} S & \longrightarrow & \boxed{X_a X_{1,2}} \mid X_a S \mid S X_b \mid a \mid b \\ \boxed{X_{1,2}} & \longrightarrow & \boxed{S X_b} \\ X_a & \longrightarrow & a \\ X_b & \longrightarrow & b \end{array}$$

Formal Construction

Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG containing no terminal+ productions.
Then we define $G_4 = \langle V_4, \Sigma, S, P_4 \rangle$ as follows.

$V_4 = V \cup \{ X_{p,i} \mid p = A \longrightarrow \alpha \in P \wedge 2 \leq i < |\alpha| \}$, where $X_{p,i} \notin V \cup \Sigma$

$P_4 = \{ A \longrightarrow \alpha \in P \mid |\alpha| \leq 2 \}$
$\quad \cup \{ A \longrightarrow A_1 X_{p,2} \mid p = A \longrightarrow A_1 \ldots A_n \in P \wedge n > 2 \}$
$\quad \cup \{ X_{p,i} \longrightarrow A_i X_{p,i+1} \mid p = A \longrightarrow A_1 \ldots A_n \in P \wedge n > 2 \wedge 2 \leq i < n-1 \}$
$\quad \cup \{ X_{p,n-1} \longrightarrow A_{n-1} A_n \mid p = A \longrightarrow A_1 \ldots A_n \in P \wedge n > 2 \}$

Correctness of Nonbinary Production Elimination

Lemma Let G be a CFG without ε-, unit- or terminal+ productions, and let G_4 be constructed as above. Then the following hold.

1. G_4 has no ε-, unit- or terminal+ productions.
2. G_4 has no nonbinary productions.
3. $\mathcal{L}(G_4) = \mathcal{L}(G) - \{\varepsilon\}$.

Note Since G_4 contains no ε-, unit-, terminal+, or nonbinary productions, it has to be in Chomsky Normal Form!

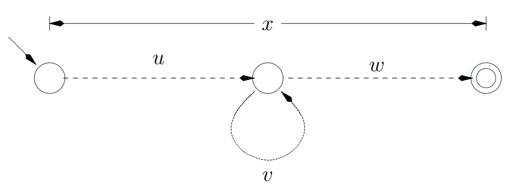
A Pumping Lemma for CFLs

Proving Languages Non-Regular

Recall how we proved languages to be nonregular.

Myhill-Nerode: A language L is regular iff its indistinguishability relation I_L has finitely many equivalence classes.

Pumping Lemma: If L is regular, and $x \in L$ is "long enough", then x can be split into u, v, w so that $uv^iw \in L$ all i.



Proving Languages Non-Context-Free

There's no Myhill-Nerode theorem for CFLs, but there is a Pumping Lemma: if L is a CFL and a word is "long enough" then parts of the word can be replicated.

Questions

- What is "long enough"?
- Which parts can be "replicated"?

To answer these questions we'll:

- introduce the notion of "derivation tree" for CFGs;
- show that CFGs in Chomsky normal form have derivation trees of a specific form.

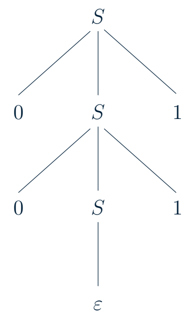
Derivation Trees

“Derivation sequences” show how CFGs generate words.

Example Let G be $S \longrightarrow \varepsilon \mid 0S1$. Then to show that G generates 0011:

$$S \Rightarrow_G 0S1 \Rightarrow_G 00S11 \Rightarrow_G 00 \cdot \varepsilon \cdot 11 = 0011$$

A *derivation tree* is a tree-like representation of a derivation sequence.



Formally Defining Derivation Trees

Definition Let $G = \langle V, \Sigma, S, P \rangle$ be a CFG, and let $w \in \Sigma^*$. Then a *derivation tree* for w in G is a labeled ordered tree satisfying the following.

- The root is labeled by S.
- Internal nodes are labeled by elements of V.singset
- Leaves are labeld by elements of $\Sigma \cup \{\varepsilon\}$.
- If A is label of an internal node and $X_1, ..., X_n$ are labels of its children from left to right then $A \longrightarrow X_1 \cdots X_n$ is a production in P.
- Concatenating the leaves from left to right forms w.

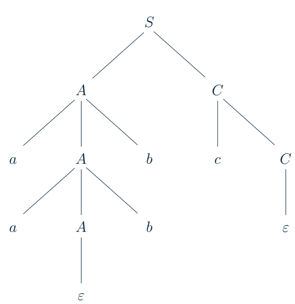
One can show that $w \in \mathcal{L}(G)$ if and only if there is a derivation tree for w in G.

Another Example Derivation Tree

Let G be:

$$\begin{array}{lcl} S & \longrightarrow & AC \\ A & \longrightarrow & aAb \mid \varepsilon \\ C & \longrightarrow & cC \mid \varepsilon \end{array}$$

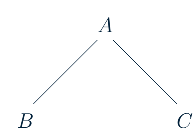
Then a derivation tree for $aabbc$ is:



Derivation Trees and Chomsky Normal Form

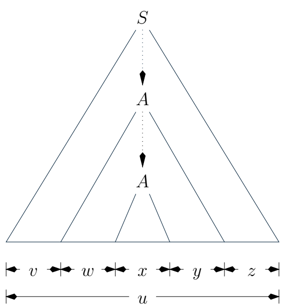
Suppose G is in CNF; what property do the derivation trees for words have?

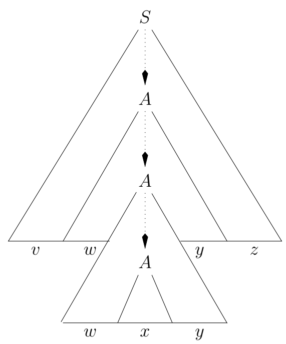
- No leaves are labeled by ε.
- Every internal node has either one child, which must be a leaf, or two children, which must both be internal.



When Are Words "Long Enough"?

If derivation tree for u is... ... then the following derivation tree also exists!





If the CFG is in CNF, one can characterize when words are "long enough" to have such trees!

CFGs, CNF and "Long Enough" Words

Suppose $G = \langle V, \Sigma, S, P\rangle$ is a CFG *in CNF*. We want to know how long a word $w \in \mathcal{L}(G)$ has to be in order to ensure the existence of a derivation like the following.

$$S \Rightarrow_G^* vAz \Rightarrow_G^* vwAyx \Rightarrow_G^* vwxyz$$

Note This holds when derivation tree contains a path of length $|V| + 1$!

- Such a path contains $|V| + 2$ nodes.
- All nodes except last one are labeled by variables.
- So some variable appears twice!

Since derivation trees in G must be binary (G is in CNF), the longest a word $w \in \mathcal{L}(G)$ can be and have a derivation tree of height $|V|$ is $2^{|V|-1}$.

So if $|w| \geq 2^{|V|-1} + 1$, then the "right kind" of derivation must exist!

The Pumping Lemma for CFLs

Theorem

If $L \subseteq \Sigma^*$ is a CFL
then there exists $N > 0$ such that for all $u \in L$,
if $|u| \geq N$
then there exist $v, w, x, y, z \in \Sigma^*$ such that:
$u = vwxyz$ and
$|wy| > 0$ and
$|wxy| \leq N$ and
for all $m \geq 0$, $vw^m xy^m z \in L$.

What is N? If n_L is the smallest number of variables needed to give a CFG G in CNF with $\mathcal{L}(G) = L - \{\varepsilon\}$, then $N = 2^{n_L - 1} + 1$.

Proving Languages Non-Context-Free Using the Pumping Lemma

As was the case with regular languages, we can use the contrapositive of the Pumping Lemma to prove languages to be non-CFLs

Lemma (Pumping Lemma for CFLs) L is a CFL $\implies P(L)$, where $P(L)$ is:

$$\exists N > 0. \forall u \in \boxed{L}. |u| \geq N \implies \exists v, w, x, y, z \in \Sigma^*.$$
$$(u = vwxyz \wedge |wy| > 0 \wedge |wxy| \leq N \wedge \forall m \geq 0. vw^m xy^m z \in \boxed{L})$$

Contrapositive $(\neg P(L)) \implies L$ is *not* a CFL.

So to prove L is not a CFL, it suffices to prove $\neg P(L)$, which can be simplified to:

$$\forall N > 0. \exists u \in L. |u| \geq N \wedge \forall v, w, x, y, z \in \Sigma^*.$$
$$(u = vwxyz \wedge |wy| > 0 \wedge |wxy| \leq N) \implies \exists m \geq 0. vw^m xy^m z \notin L)$$

Example: Proof that $L = \{ a^n b^n c^n \mid n \geq 0 \}$ Is Not a CFL

On the basis of the Pumping Lemma it suffices to prove the following.

$$\forall N > 0.\, \exists u \in L.\, |u| \geq N \wedge \forall v, w, x, y, z \in \Sigma^*.$$
$$(u = vwxyz \wedge |wy| > 0 \wedge |wxy| \leq N) \implies \exists m \geq 0.\, vw^m xy^m z \notin L)$$

So fix $N > 0$ and consider $\boxed{u = a^N b^N c^N}$; clearly $u \in L$ and $|u| \geq N$. Now fix $v, w, x, y, z \in \Sigma^*$ so that the following hold.

- $u = vwxyz$
- $|wy| > 0$
- $|wxy| \leq N$

Proof (cont.)

We wish to show that there is an m such that $vw^m xy^m z \notin L$. There are two cases to consider.

1. $wxy \in \{a, b\}^*$ (i.e. contains no c's).
2. $wxy = w'c^i$ some $i > 0$, $w' \in \{a, b\}^*$ (i.e. does contain c's).

For both cases, consider $m = 0$. In case 1, $vw^0 xy^0 z \notin L$, since $vw^0 xy^0 z$ contains n c's but $< n$ of either a's or b's. In case 2, $w' \in \{b\}^*$ since $|wxy| \leq N$. Consequently, $vw^0 xy^0 z$ contains n a's but $< n$ b's or c's. So we have demonstrated the existence of m with $vw^m xy^m z \notin L$, and L is not context-free.

Ramifications

- Non-context-free languages exist! Other examples:
  - $\{ ww \mid w \in \{a, b\}^* \}$
  - $\{ a^m b^n c^m d^n \mid m, n \geq 0 \}$

  However, $\{ a^m b^n c^n d^m \mid m, n \geq 0 \}$ is a CFL.
  Moral: In CFLs can count pairwise and "outside in".
- CFLs are not closed with respect to $\cap$! Let $L = \{ a^n b^n c^n \mid n \geq 0 \}$. Then $L = L_1 \cap L_2$ where:

$$L_1 = \{ a^n b^n c^m \mid m, n \geq 0 \}$$
$$L_2 = \{ a^m b^n c^n \mid m, n \geq 0 \}$$

  Both L_1 and L_2 are CFLs.

Ramifications (cont.)

- CFLs are not closed with respect to complementation!
  - CFLs are closed with respect to $\cup$.
  - $L_1 \cap L_2 = (L_1' \cup L_2')'$