Automata Theory and Formal Grammars: Lecture 7

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 \( \varepsilon \)-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 languages 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

A CFG \((V, \Sigma, S, P)\) is in Chomsky Normal Form (CNF) if every production has one of two forms.
- \(A \rightarrow BC\) for \(B, C \in V\)
- \(A \rightarrow a\) for \(a \in \Sigma\)

Examples
1. Is \(S \rightarrow \varepsilon \mid 0S1\) in CNF?
   No; both productions violate the two allowed forms.
2. Is \(S \rightarrow SS \mid 0 \mid 1\) 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 \( G \):

\[
S \rightarrow SS \mid 0 \mid 1 \mid \varepsilon
\]

and look at this derivation of \( 01 \).

\[
S \Rightarrow_G SS \Rightarrow_G SSS \Rightarrow_G SSSS \Rightarrow_G SS \Rightarrow_G 0S \Rightarrow_G 01
\]

The “intermediate strings” can grow and shrink!

Converting CFGs into CNF

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

No! If \( G' \) is in CNF, then \( \varepsilon \not\in L(G) \)!

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

1. Eliminate \( \varepsilon \)-productions (i.e. productions of form \( A \rightarrow \varepsilon \)).
2. Eliminate unit productions (i.e. productions of form \( A \rightarrow B \)).
3. Eliminate terminal+ productions (i.e. productions of form \( A \rightarrow aC \) or \( A \rightarrow abA \)).
4. Eliminate nonbinary productions (i.e. productions of form \( A \rightarrow ABA \)).

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.

\[
S \rightarrow SS \mid 0 \mid 1
\]

The derivation for \( 01 \) is:

\[
S \Rightarrow_{G'} SS \Rightarrow_{G'} 0S \Rightarrow_{G'} 01.
\]

Eliminating \( \varepsilon \)-Productions

CNF grammars contain no \( \varepsilon \)-productions, and yet arbitrary CFGs may. To convert a CFG to CNF, we therefore need a way of eliminating them. Of course, CFGs without \( \varepsilon \)-productions cannot generate the word \( \varepsilon \).

Goal Given CFG \( G \), generate CFG \( G_1 \) such that:

- \( G_1 \) has no \( \varepsilon \)-productions; and
- \( L(G_1) = L(G) - \{\varepsilon\} \).
Eliminating $\varepsilon$-Productions: The Naive Approach

Can we just eliminate the $\varepsilon$-productions?

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

$$S \rightarrow \varepsilon \mid 0S1$$

Answer: $\emptyset$!

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

So How Can We Eliminate $\varepsilon$-Productions?

$\varepsilon$-productions add “derivational capability” in CFGs by allowing variables to be “eliminated” in a derivation step.

Example Consider the CFG $G$ given as follows.

$$S \rightarrow \varepsilon \mid 0S1$$

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

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

1. Precisely what “derivational capability” do $\varepsilon$-productions provide?
2. How can we recover this capability without $\varepsilon$-productions?

Nullability

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

E.g. Consider the following CFG.

$$S \rightarrow ABCBC$$
$$A \rightarrow CD$$
$$B \rightarrow Cb$$
$$C \rightarrow a \mid \varepsilon$$
$$D \rightarrow bD \mid \varepsilon$$

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

Why are variables nullable? Because of $\varepsilon$-productions! So nullability is the “derivational capability” that $\varepsilon$-productions add to a CFG.

Generating a $\varepsilon$-Production-Free CFGs

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

If we remove the $\varepsilon$-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

$$S \rightarrow \varepsilon \mid 0S1$$

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

$$S \rightarrow 0S1 \mid 01$$
Constructing $\varepsilon$-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 $G_1 = \langle V_1, \Sigma, S_1, P_1 \rangle$.

\[
V_1 = V \\
S_1 = S \\
P_1 = P - \{ A \rightarrow \varepsilon \mid A \rightarrow \varepsilon \in P \} \\
\cup \{ A \rightarrow \alpha_0 \cdots \alpha_n \mid \alpha_0 \cdots \alpha_n \neq \varepsilon \land \exists A_1, \ldots, A_n \in N, A \rightarrow \alpha_0 A_1 \alpha_1 \cdots \alpha_n A_n \epsilon \in P \}
\]

Huh?

$P_1$ contains:
- the non-$\varepsilon$-productions in $P$, together with
- productions obtained by selectively omitting occurrences of nullable variables.
  - $A \rightarrow \alpha_0 A_1 \alpha_1 \cdots \alpha_n A_n$ is a production in $G$.
  - The $A_i$ are nullable variables.
  - The $\alpha_i$ is the “stuff” in-between the $A_i$.
  - $A \rightarrow \alpha_0 \cdots \alpha_n$ is a modified production with the $A_i$’s omitted.

The idea is that in the original grammar, $A \Rightarrow \alpha_0 \cdots \alpha_n$ 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 $A \rightarrow \varepsilon$ then $A \in N(G)$.
- If $A \rightarrow B_1 \cdots B_n$ and $B_1, \ldots, B_n \in N(G)$ 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.

\[
S \rightarrow ABCBC \\
A \rightarrow CD \\
B \rightarrow Cb \\
C \rightarrow a \mid \varepsilon \\
D \rightarrow bD \mid \varepsilon 
\]

First, calculate $N(G)$.

\[
N(G)_0 = \emptyset \\
N(G)_1 = \{ C, D \} \\
N(G)_2 = \{ A, C, D \} \\
N(G)_3 = N(G)_2
\]
Recall \( G \); remember that \( N(G) = \{A, C, D\} \).

\[
\begin{align*}
S & \rightarrow ABCBC & C & \rightarrow a \mid \varepsilon \\
A & \rightarrow CD & D & \rightarrow bD \mid \varepsilon \\
B & \rightarrowCb
\end{align*}
\]

\( G_1 \) is boxed transitions are new ones:

\[
\begin{align*}
S & \rightarrow ABCBC \mid ABCB \mid ABBC \mid ABB \\
& \quad \mid BCBC \mid BCB \mid BBC \mid BB \\
A & \rightarrow CD \mid C \mid D \\
B & \rightarrowCb \mid D \\
C & \rightarrow a \\
D & \rightarrow bD \mid b
\end{align*}
\]

Where are we?

So Far

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

To Do

- Eliminating:
  - unit
  - terminal+
  - nonbinary productions

from CFGs.

Converting CFGs into Chomsky Normal Form

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

Last time we proved the following.

Lemma

Let \( G \) be a CFG. Then there is a CFG \( G_1 \) containing no \( \varepsilon \)-productions and such that \( \mathcal{L}(G_1) = \mathcal{L}(G) - \{\varepsilon\} \).

I.e. we now know how to eliminate \( \varepsilon \)-productions! What about the others?

Eliminating Unit Productions

Definition

A unit production has form \( A \rightarrow B \) where \( B \in V \).

Like \( \varepsilon \) 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:

\[
\begin{align*}
S & \rightarrow A \mid C \\
A & \rightarrow aA \mid B \\
B & \rightarrow bB \mid b \\
C & \rightarrow eC \mid c
\end{align*}
\]

In order to remove \( S \rightarrow A \), need to add e.g. \( S \rightarrow aA! \)
But Which Productions Do We Need To Add?

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

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

where each $A_i \in V$ is a single variable. If $\alpha$ is not just a single variable, then we should add a production $A \rightarrow \alpha$. How do we determine these $\alpha$’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 $B \rightarrow C \in P$ then $C \in U(G, A)$.

**Example**

Let $G$ be given as follows.

- $S \rightarrow A \mid C$
- $A \rightarrow aA \mid B$
- $B \rightarrow bB \mid b$
- $C \rightarrow cC \mid c$

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

- $U(G, S)_0 = \emptyset$
- $U(G, S)_1 = \{S\}$
- $U(G, S)_2 = \{S, A, C\}$
- $U(G, S)_3 = \{S, A, B, C\} = U(G, S)_4$

**Example (cont.)**

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:

- $S \rightarrow aA \mid bB \mid b \mid cC \mid c$
- $A \rightarrow aA \mid bB \mid b$
- $B \rightarrow bB \mid b$
- $C \rightarrow cC \mid c$

$U(G, A)$ 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 $A \rightarrow \alpha$ for every variable $A$, where $B \rightarrow \alpha$ in original CFG and $B \in U(G, A)$!
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.

\[
P_2 = \{ A \rightarrow \alpha \mid \exists B \in U(G, A), \alpha. \, B \rightarrow \alpha \in P \land \alpha \not\in V \}
\]

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

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

Example

Let \( G \) be given by:

\[
S \rightarrow aSb \mid aS \mid Sb \mid a \mid b
\]

Then \( G_3 \) is:

\[
S \rightarrow X_aSX_b \mid X_aS \mid SX_b \mid a \mid b \\
X_a \rightarrow a \\
X_b \rightarrow b
\]

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.

\[
V_3 = V \cup \{ X_a \mid a \in \Sigma \}, \text{ where } X_a \not\in V \cup \Sigma
\]

\[
P_3 = \{ A \rightarrow \alpha' \mid A \rightarrow \alpha \in P \land \alpha' \text{ is } \alpha \text{ with } a \text{ replaced by } X_a \text{ if } A \rightarrow \alpha \text{ is terminal+} \}
\]

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

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

Eliminating Terminal+ Productions

Definition A production \( A \rightarrow \alpha \) is terminal+ if \( |\alpha| \geq 2 \) and \( \alpha \) contains at least one terminal.

Examples

- \( A \rightarrow Ca \)
- \( A \rightarrow aba \)

Eliminating these is fairly simple:

- Introduce a new variable \( X_a \) for each terminal \( a \in \Sigma \).
- Add productions \( X_a \rightarrow a \).
- In each terminal+ production, replace terminals \( a \) by variables \( X_a \).
Eliminating Nonbinary Productions

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

**Example**
$A \rightarrow BAB$

How do we eliminate these?
- For each such production $p = A \rightarrow A_1A_2...A_n$ and $n \geq 3$, we will introduce new variables $X_{p,2}, \ldots, X_{p,n-1}$.
- Replace $A \rightarrow A_1A_2...A_n$ by a collection of productions:
  - $A \rightarrow A_1X_{p,2}$
  - $X_{p,2} \rightarrow A_2X_{p,3}$
  - $\vdots$
  - $X_{p,n-1} \rightarrow A_{n-1}A_n$

Explaining the Idea

Suppose we have a production $A \rightarrow BCCD$. The construction would replace it with the following.
- $A \rightarrow BX_{p,2}$
- $X_{p,2} \rightarrow CX_{p,3}$
- $X_{p,3} \rightarrow CD$

In the original CFG, $A \Rightarrow^* G BCCD$ in one step.
In the new CFG it takes three steps:
- $A \Rightarrow G BX_{p,2} \Rightarrow G BCX_{p,3} \Rightarrow G BCCD$.

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.
- $V_4 = V \cup \{X_{p,i} \mid p = A \rightarrow \alpha \in P \land 2 \leq i < |\alpha|\}$, where $X_{p,i} \notin V \cup \Sigma$
- $P_4 = \{A \rightarrow \alpha \in P \mid |\alpha| \leq 2\}
  \cup \{A \rightarrow A_1X_{p,2} \mid p = A \rightarrow A_1...A_n \in P \land n > 2\}
  \cup \{X_{p,i} \rightarrow A_iX_{p,i+1} \mid p = A \rightarrow A_1...A_n \in P \land n > 2 \land 2 \leq i < n - 1\}
  \cup \{X_{p,n-1} \rightarrow A_{n-1}A_n \mid p = A \rightarrow A_1...A_n \in P \land n > 2\}$

Example

Let $G$ be:
- $S \rightarrow X_aSX_b \mid X_aS \mid SX_b \mid a \mid b$
- $X_a \rightarrow a$
- $X_b \rightarrow b$

Then $G_4$ is:
- $S \rightarrow X_1X_{1,2} \mid X_aS \mid SX_b \mid a \mid b$
- $X_{1,2} \rightarrow SX_b$
- $X_a \rightarrow a$
- $X_b \rightarrow b$
**Correctness of Nonbinary Production Elimination**

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

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

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

**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 $uv^i w \in L$ all $i$.

---

**A Pumping Lemma for CFLs**

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 $S \rightarrow \varepsilon \mid 0S1$. 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 $V$.
- Leaves are labeled by elements of $\Sigma \cup \{\varepsilon\}$.
- If $A$ is label of an internal node and $X_1, \ldots, X_n$ are labels of its children from left to right then $A \rightarrow X_1 \ldots X_n$ is a production in $P$.
- Concatenating the leaves from left to right forms $w$.

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

Another Example Derivation Tree

Let $G$ be:

- $S \rightarrow AC$
- $A \rightarrow aAb \mid \varepsilon$
- $C \rightarrow eC \mid \varepsilon$

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 $\varepsilon$.
- 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!

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 $m \geq 0$, $vw^mxy^mz \in L$.

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

CFGs, CNF and “Long Enough” Words

Suppose $G = (V, \Sigma, S, P)$ is a CFG in CNF. We want to know how long a word $w \in L(G)$ 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 $w \in L(G)$ 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!

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:

$\exists N > 0. \forall u \in L. |u| \geq N \implies \exists v, w, x, y, z \in \Sigma^*$.

$(u = vwxyz \land |wy| > 0 \land |wxy| \leq N \land \forall m \geq 0. vw^mxy^mz \in L)$

**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:

$\forall N > 0. \exists u \in L. |u| \geq N \land \forall v, w, x, y, z \in \Sigma^*$.

$(u = vwxyz \land |wy| > 0 \land |wxy| \leq N) \implies \exists m \geq 0. vw^mxy^mz \notin L)$
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.

\[
\forall N > 0, \exists u \in L, |u| \geq N \land \forall v, w, x, y, z \in \Sigma^*, (u = vxwyz \land |wy| > 0 \land |wxy| \leq N) \implies \exists m \geq 0, vw^mxy^mz \notin L
\]

So fix \( N > 0 \) and consider \( u = a^N b^N c^N \). Clearly \( u \in L \) and \( |u| \geq N \).

Now fix \( v, w, x, y, z \in \Sigma^* \) so that the following hold.

- \( u = vxwyz \)
- \(|wy| > 0\)
- \(|wxy| \leq N\)

Proof (cont.)

We wish to show that there is an \( m \) such that \( vw^mxy^mz \notin L \). 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 \( m = 0 \).

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

Ramifications

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

However, \( \{ a^m b^n c^n d^n \mid m, n \geq 0 \} \) 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:

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

Both \( L_1 \) and \( L_2 \) are CFLs.

Ramifications (cont.)

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