The Pumping Lemma for CFLs and Properties of Context-Free Languages

The Pumping Lemma for CFLs

The Pumping Lemma for CFLs

Intuition:

- The **pumping lemma of regular languages** tell us that
  - **If there was a string long enough to cause a cycle in the DFA for the language, then we could "pump" the cycle (a piece of the string) and discover an infinite sequence of strings that had to be in the language.**

- The **pumping lemma of context-free languages** tell us that
  - **If there was a string long enough to cause a cycle (same variable appears more than once in the derivation), then we can always find two pieces of this sufficiently long string to "pump" in tandem and discover an infinite sequence of strings that had to be in the language.**
  - **That is: if we repeat each of the two pieces the same number of times, we get another string in the language.**

The Pumping Lemma for CFLs

The Size of Parse Trees

- The first step in deriving a pumping lemma for CFLs is to examine the shape and size of parse trees.
- One of the uses of Chomsky Normal Form (CNF) is to turn parse trees into binary trees.
  - Remember that every CFG can be converted to an equivalent grammar in CNF, and
  - **A context-free grammar is in *Chomsky Normal Form* if every rule is of the form**
    $A \rightarrow BC$ or $A \rightarrow a$

THEOREM: ***The Size of Parse Trees for Grammars in CNF***

Suppose we have a parse tree according to a Chomsky-Normal-Form grammar G = (V, T, P, S), and suppose that the yield of the tree is a terminal string w. If the length of the longest path is n, then $|w| \leq 2^{n-1}$.

The Pumping Lemma for CFLs

THEOREM: ***(The pumping lemma for context-free languages)***

Let L be a CFL, then there exists a constant n such that if z is any string in L such that |z| is at least n, then we can write z = uvwxy, subject to the following conditions:

1. **$|vwx| \leq n$** That is, the middle portion is not too long.

2. **$vx \neq \varepsilon$** Since v and x are the pieces to be "pumped," this condition says that at least one of the strings we pump must not be empty.

3. **For all $i \geq 0$, uv^iwx^iy is in L.**

   That is, the two strings v and x may be "pumped" any number of times, including 0, and the resulting string will still be a member of L.

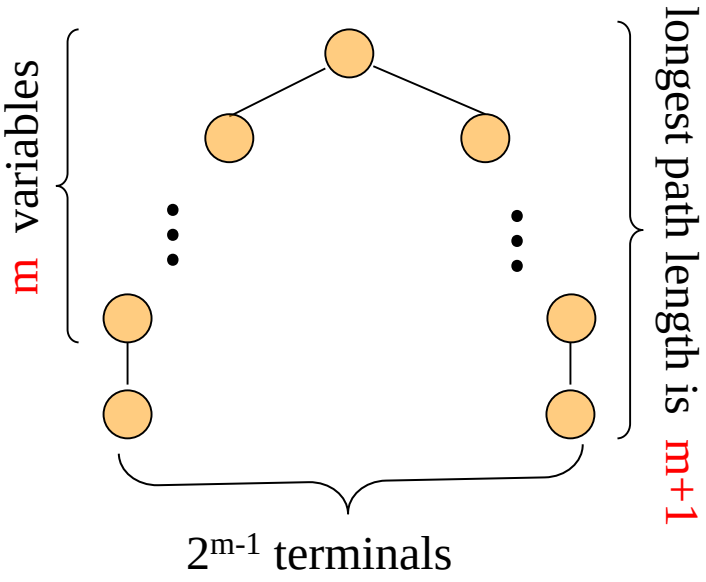
Proof of the Pumping Lemma for CFLs

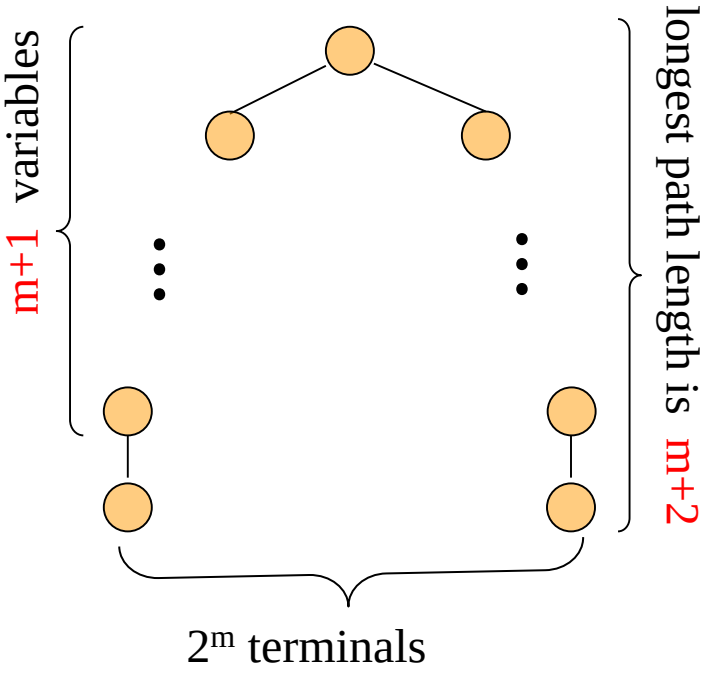
- **The first step is to find a Chomsky Normal Form grammar G for language L.**
- **If L is ϕ, {ε} or it contains ε, this does not cause any problem in the proof.**
  - If L is ϕ or {ε} then the statement of the theorem, which talks about a non-empty string z in L surely cannot be violated, since there is no such z in L in this case.
  - Also, if L contains ε, then it is not also a problem, the selected string z must be non-empty.
- **Now, starting with a CNF grammar G=(V,T,P,S) such that L(G)=L-{ε}.**
  - **Let the grammar have m variables.**
  - **Pick $n = 2^m$.**
  - **Let $|z| \geq n$.**
  - **We claim ("*ShapeOfParseTree Claim*") that a parse tree with yield z must have a path of length m+2 or more.**

Proof of the Pumping Lemma for CFLs

Shape of Parse Tree for CNF Grammars

- If the length of the longest path in the parse tree of a CNF grammar is $m+1$, then the longest yield has length 2^{m-1}, and there are m variables on the longest path.
- If the length of the longest path in the parse tree of a CNF grammar is $m+2$, then the longest yield has length 2^{m}, and there are $m+1$ variables on the longest path.





Proof of the Pumping Lemma for CFLs

- **We start with a CNF grammar G=(V,T,P,S) such that L(G)=L-{ε}.**
  - **Let the grammar have m variables.**
  - **Pick $n = 2^m$.**
  - **Let $|z| \geq n$.**

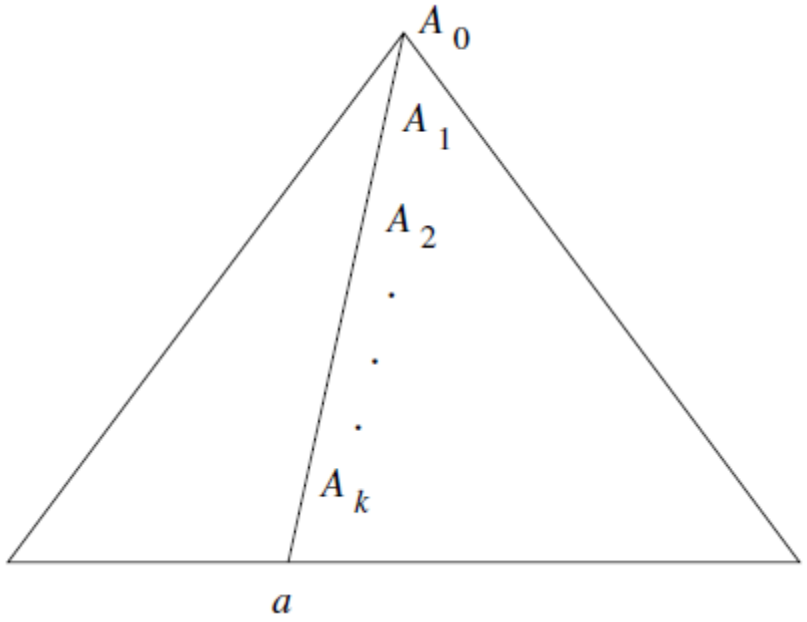
- Now, we know that the parse tree for z has a path with at least m+1 variables beacuse the length of the longest path ≥ m+2.

- **There are only m different variables, so among the lowest m+1 variables on that path, we can find two nodes A_i and A_j with the same label, say A.**

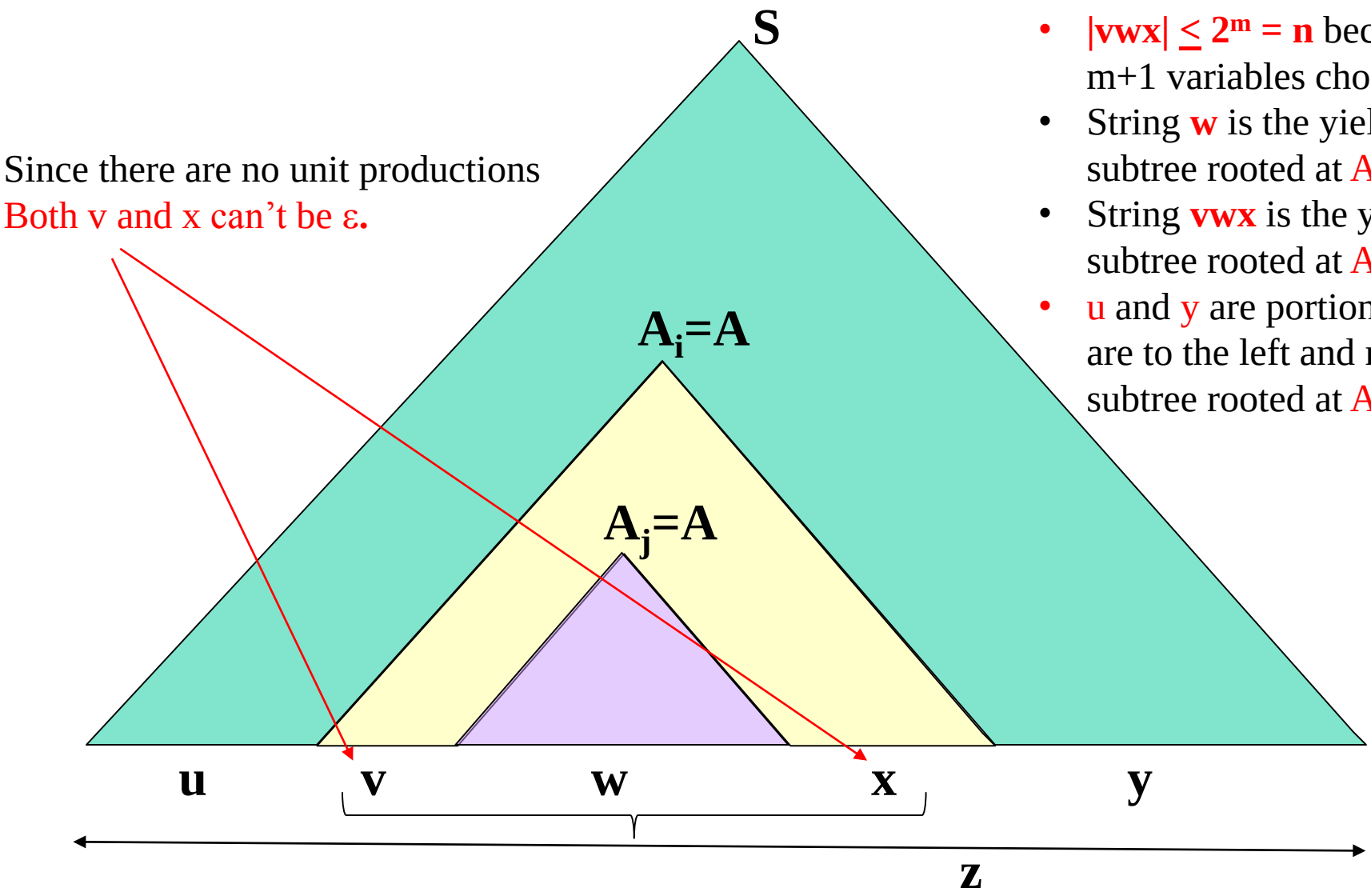
Proof of the Pumping Lemma for CFLs

- **There are only m different variables, so among the lowest m+1 variables on that path, we can find two nodes A_i and A_j with the same label, say A.**

- Since k≥m, there are at least m+1 occurrences of variables A_0, A_1,..., A_k on the path.
- As there are only m different variables in V, at least two of the last m + 1 variables on the path (that is, A_{k-m} through A_k, inclusive) must be the same variable.
- Suppose $A_i=A_j$, where $k-m \leq i < j \leq k$.



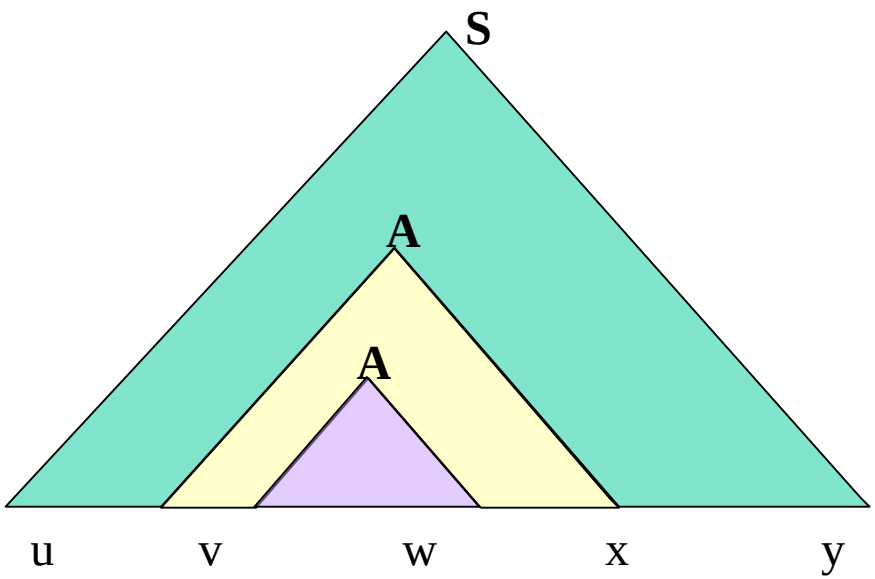
Proof of the Pumping Lemma for CFLs

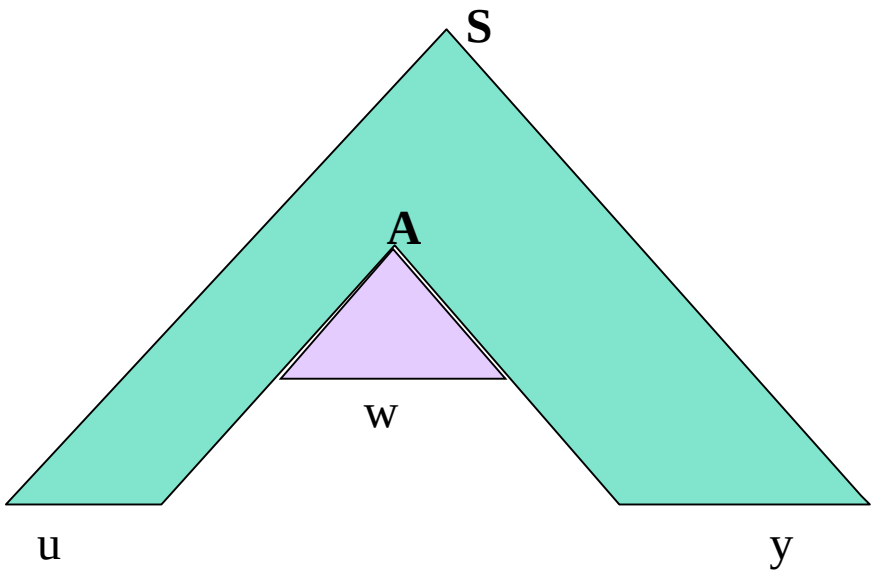


- $|vwx| \leq 2^m = n$ because lowest m+1 variables chosen.
- String **w** is the yield of the subtree rooted at A_j (A).
- String **vwx** is the yield of the subtree rooted at A_i (A).
- u and y are portions of z that are to the left and rightof the subtree rooted at A_i .

Proof of the Pumping Lemma for CFLs

pump zero times

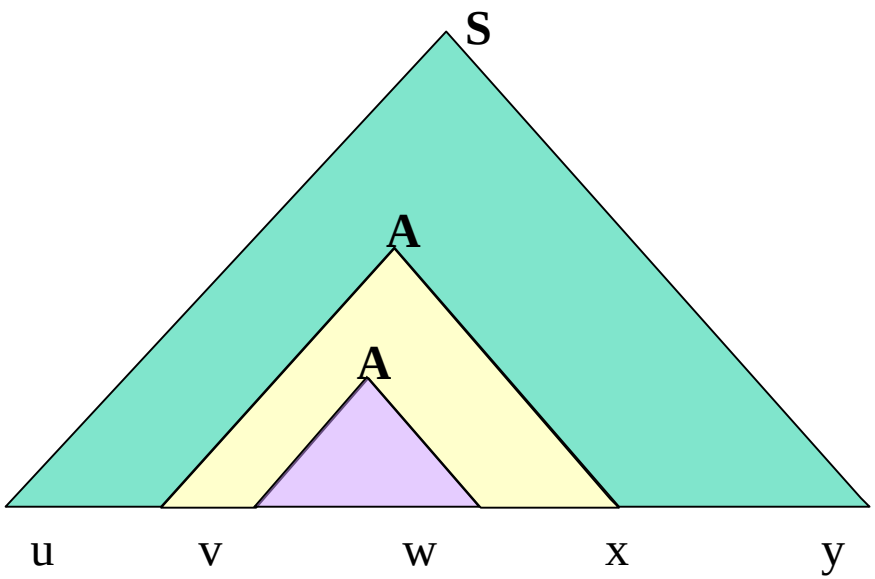


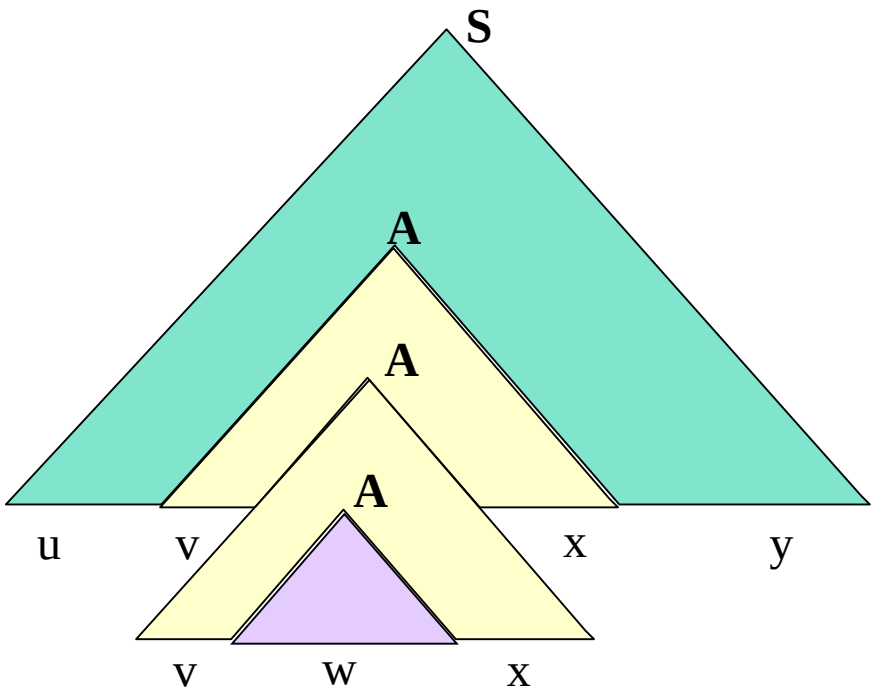


$uwy = uv^0wx^0y$ must be in the language.

Proof of the Pumping Lemma for CFLs

pump tvice

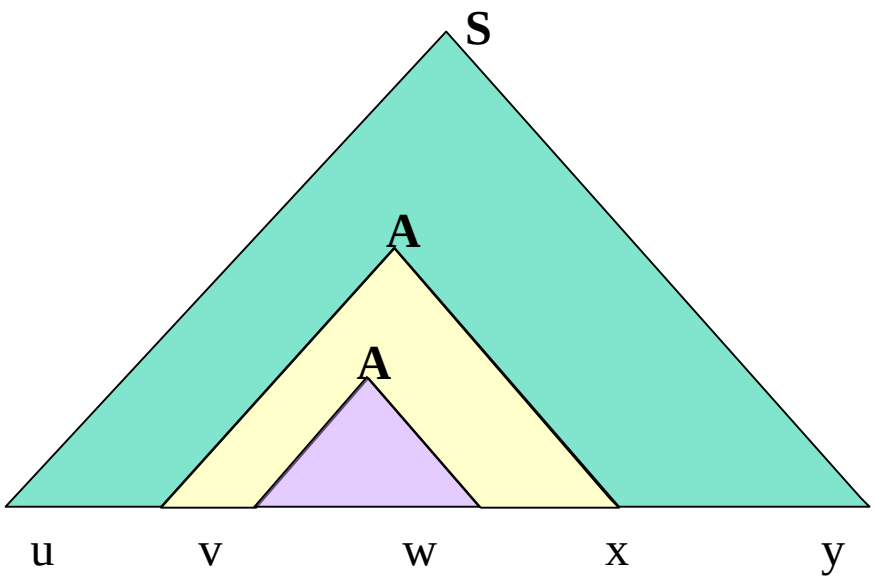


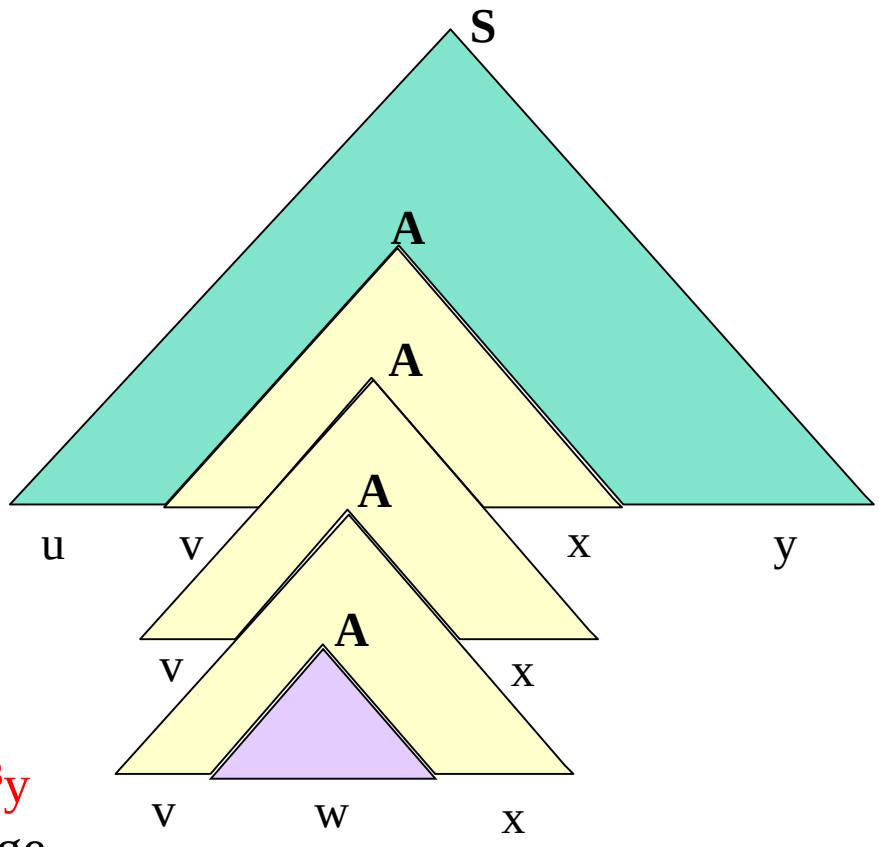


$uvvwxxy = uv^2wx^2y$
must be in the language.

Proof of the Pumping Lemma for CFLs

pump three times

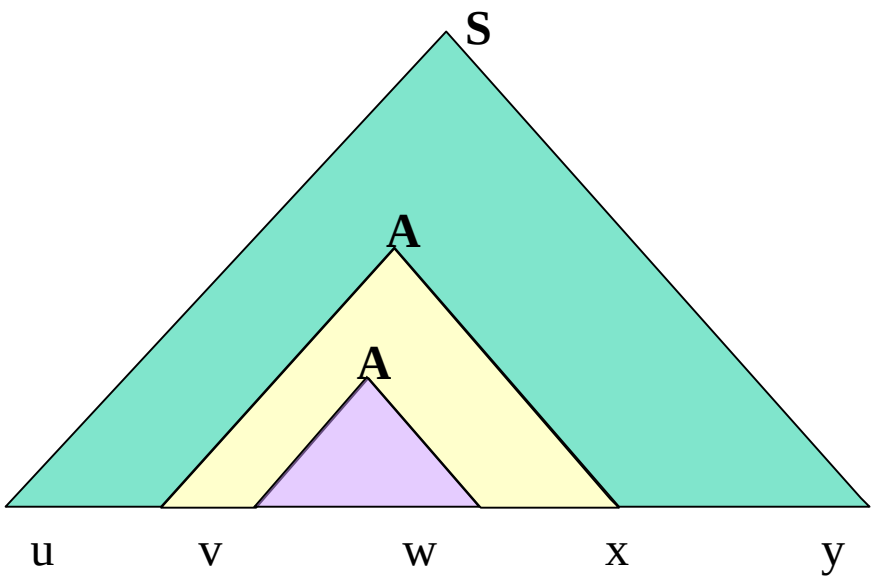


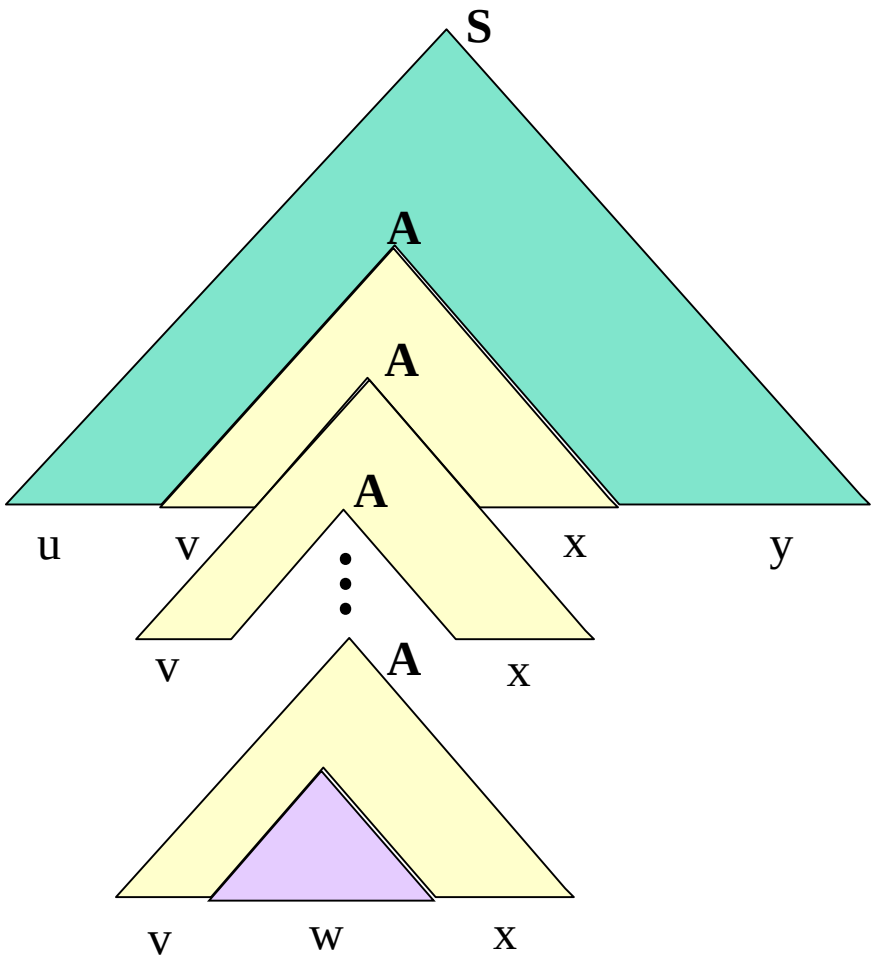


$uvvvwxvxy = uv^3wx^3y$
must be in the language.

Proof of the Pumping Lemma for CFLs

pump i times





Thus, uv^iwx^iy where $i \geq 0$ must be in the language.

Using the Pumping Lemma

- **In order to show that a language L is NOT a CFL using the Pumping Lemma:**
  1. **Suppose L were a CFL.**
  2. **Then there is an integer n given us by the pumping lemma, which we do not know, we must plan for any possible n.**
  3. **Pick a string z which must be in L, it must be defined using n and $|z| \geq n$.**
     - **Tricky Part 1: You should find a string z so that you can create a contradiction in step 5. YOU CANNOT SELECT A SPECIFIC STRING.**
  4. **Break z into uvwxy, subject only to the constraints that $|vwx| \leq n$ and $vx \neq \varepsilon$.**
  5. **Pick i and show that uv^iwx^iy is NOT L in order to create a contradiction.**
     - **Tricky Part 2: You have to show that uv^iwx^iy is NOT in L using only the constraints that $|vwx| \leq n$ and $vx \neq \varepsilon$. You may need to look at more than one cases. YOU CANNOT GIVE A SPECIFIC EXAMPLE.**
  6. **Conclude that L is NOT a CFL.**

Using the Pumping Lemma – Example

Example: Let L be the language $\{0^k1^k2^k \mid k \geq 1\}$. Show that this language is NOT a CFL using the Pumping Lemma:

1. Suppose L were a CFL.
2. Then there is an integer n given us by the pumping lemma.
3. Let us pick a string $z = 0^n1^n2^n$ and $0^n1^n2^n$ is in L.
4. Break z into uvwxy, where $|vwx| \leq n$ and $vx \neq \varepsilon$.
5. Pick 0 for i and we have show that uwy is NOT L in order to create a contradiction.
   - Since $|vwx| \leq n$, we know that vwx cannot involve both 0's and 2's, since the last 0 and the first 2 are separated by n+1 positions. So, there are two cases.

Using the Pumping Lemma – Example

- Case 1: vwx has no 2's:
  - Then vx consists of only 0's and 1's.
  - Since , v or/and x has at least one of these symbols.
  - Then uwy has n 2's but it has fewer than n 0's or fewer than n 1's or both.
  - Therefore, uwy does not belong to L and creates a contradiction with our assumption that L were a CFL.
  - We conclude that L is NOT a CFL in case 1.
- Case 2: vwx has no 0's:
  - Then vx consists of only 1's and 2's.
  - Since , v or/and x has at least one of these symbols.
  - Then uwy has n 0's but it has fewer than n 1's or fewer than n 2's or both.
  - Therefore, uwy does not belong to L and creates a contradiction with our assumption that L were a CFL.
  - We also conclude that L is NOT a CFL in case 2.
- Whichever case holds, we conclude that L has a string we know NOT to be in L.

6. This contradiction allows us to conclude that our assumption was wrong; and L is not a CFL.

Closure Properties of Context-Free Languages

Closure Properties of Context-Free Languages

- **CFLs are closed under**
  - **Substitution**
  - **Union**
  - **Concatenation**
  - **Kleene Closure**
- **CFLs are also closed under**
  - **Reversal**
  - **Homomorphisms and Inverse Homomorphisms.**

- **CFLs are NOT closed under**
  - **Intersection**
  - **Difference**

Closure Properties of Context-Free Languages

Substitution

Substitutions:

- **Σ** is an alphabet. For every symbol **a** in **Σ,** we choose a language L_a.
  - Chosen languages can be over any alphabets.
- The choice of languages defines a function **s (substitution)** on Σ and we refer to $\mathbf{L_a}$ as **s(a)** for each symbol **a**.
- If $\mathbf{w=a_1 \ldots a_n}$ is a string **Σ***, the **substitution of w s(w)** is the language of all strings $\mathbf{x_1 \ldots x_n}$ such that $\mathbf{x_i}$ is in the language of $\mathbf{s(a_i)}$.
- **Substitution of a language s(L) is the union of s(w) for all strings w in L.**

THEOREM:

If L is a context-free language over alphabet Σ and s is a substitution on Σ such that s(a) is a CFL for each a in Σ then s(L) is a CFL.

Closure Properties of Context-Free Languages

Substitution

L: $S \rightarrow 0S0 \mid 1S1 \mid \varepsilon$

Substition s(0): L_0: $A \rightarrow aAb \mid \varepsilon$

Substition s(1): L_1: $B \rightarrow 01B \mid 01$

Substition s(L) is:

$S \rightarrow ASA \mid BSB \mid \varepsilon$

$A \rightarrow aAb \mid \varepsilon$

$B \rightarrow 01B \mid 01$

- **Since L, L_0, L_1 are CFLs then s(L) is also a CFL.**

Closure of CFLs Under Union

- **CFLs are closed under Union.**
- Let L and M be CFL's with grammars G and H, respectively.
- Assume G and H have no variables in common.
  - Names of variables do not affect the language.
- Let S_1 and S_2 be the start symbols of G and H.
- Form a new grammar for $L \cup M$ by combining all the symbols and productions of G and H.
- Then, add a new start symbol S.
- Add productions $S \rightarrow S_1 \mid S_2$.
- In the new grammar, all derivations start with S.
- The first step replaces S by either S_1 or S_2.
- In the first case, the result must be a string in L(G) = L, and in the second case a string in L(H) = M.

Closure of CFLs Under Concatenation

- **CFLs are closed under Concatenation.**

- Let L and M be CFL's with grammars G and H, respectively.
- Assume G and H have no variables in common.
- Let S_1 and S_2 be the start symbols of G and H.
- Form a new grammar for LM by starting with all symbols and productions of G and H.
- Add a new start symbol S.
- Add production $S \rightarrow S_1S_2$.
- Every derivation from S results in a string in L followed by one in M.

Closure Under Star

- **CFLs are closed under Star.**

- Let L have grammar G, with start symbol S_1.
- Form a new grammar for L* by introducing to G a new start symbol S and the productions $S \rightarrow S_1S \mid \epsilon$.
- A rightmost derivation from S generates a sequence of zero or more S_1's, each of which generates some string in L.

Closure of CFLs Under Reversal

- **CFLs are closed under Reversal.**

- If L is a CFL with grammar G, form a grammar for L^R by reversing the right side of every production.

Example:

- Let G have $S \rightarrow 0S1 \mid 01$.
- The reversal of L(G) has grammar $S \rightarrow 1S0 \mid 10$.

Closure of CFLs Under Homomorphism

- **CFLs are closed under Homomorphism.**

- Let L be a CFL with grammar G.
- Let h be a homomorphism on the terminal symbols of G.
- Construct a grammar for h(L) by replacing each terminal symbol **a** by **h(a).**

Example:

- G has productions S → 0S1 | 01.
- h is defined by h(0) = ab, h(1) = ϵ.
- h(L(G)) has the grammar with productions S → abS | ab.

Nonclosure Under Intersection

- **Unlike the regular languages, CFLs are NOT closed under Intersection.**

- We know that $L_1 = \{0^n1^n2^n \mid n \geq 1\}$ is not a CFL (use the pumping lemma).
- However, $L_2 = \{0^n1^n2^i \mid n \geq 1, i \geq 1\}$ is a CFL, and its CFG is:

  $S \rightarrow AB$ $\quad A \rightarrow 0A1 \mid 01$ $\quad B \rightarrow 2B \mid 2$

- So is $L_3 = \{0^i1^n2^n \mid n \geq 1, i \geq 1\}$ is a CFL, and its CFG is:

  $S \rightarrow AB$ $\quad A \rightarrow 0A \mid 0$ $\quad B \rightarrow 1B2 \mid 12$

- But $L_1 = L_2 \cap L_3$ is **NOT** a CFL.

Nonclosure Under Difference

- **CFLs are NOT closed under Difference.**

- We can prove something more general:
  - Any class of languages that is closed under difference is closed under intersection.

Proof:

- $L \cap M = L - (L - M)$.
- Thus, if CFL's were closed under difference, they would be closed under intersection, but they are not.

Nonclosure Under Complement

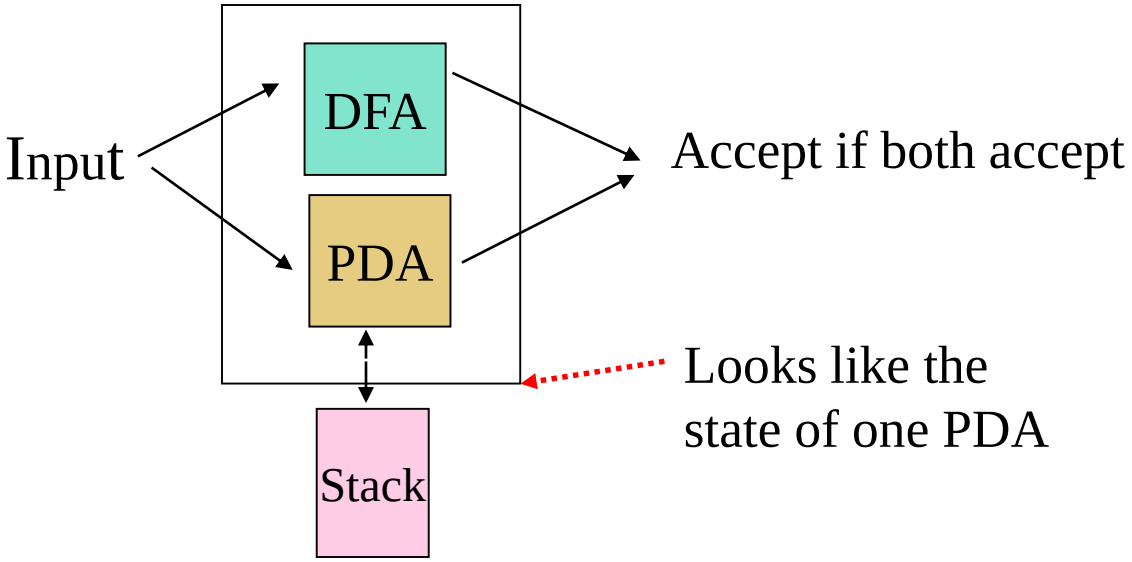
- **CFLs are NOT closed under Complement.**
- **If L is a CFL, its complement $\bar{L}$ is NOT necessarily a CFL.**

Proof:

- Suppose that $\bar{L}$ is always context free when L is a CFL.
- Then since $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$
- and CFLs are closed under union, it would follow that the CFLs are closed under intersection. However we know they are not closed under intersection.

Intersection with a Regular Language

- **Intersection of two CFL's need not be context free.**
- **But the intersection of a CFL with a regular language is always a CFL.**
  - Proof involves running a DFA in parallel with a PDA, and noting that the combination is a PDA.
  - PDAs accept by final state.



Intersection with a Regular Language

Formal Construction

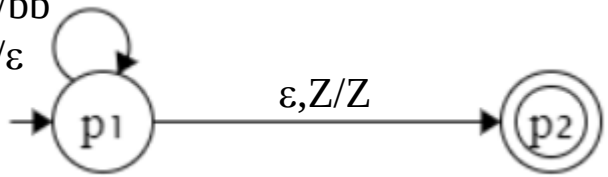
- **The intersection of a CFL with a regular language is always a CFL**

- Let the DFA A have transition function $\boldsymbol{\delta_A}$.
- Let the PDA P have transition function $\boldsymbol{\delta_P}$.
- States of combined PDA are **[q,p]**, where **q** is a state of A and **p** a state of P.
- $\boldsymbol{\delta([q,p], a, X)}$ contains $\boldsymbol{([\delta_A(q,a),r], \alpha)}$ if $\boldsymbol{\delta_P(p, a, X)}$ contains $\boldsymbol{(r, \alpha)}$.
  - Note **a** could be **ε**, in which case $\boldsymbol{\delta_A(q,a) = q}$.
- Accepting states of combined PDA are those **[q,p]** such that **q** is an accepting state of A and **p** is an accepting state of P.
- Initial state of combined PDA is [q,p] such that q is the initial state of A and p is the initial state of P.

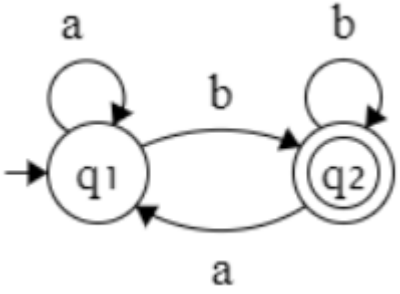
Intersection of a CFL with a Regular Language

a,Z/aZ
a,a/aa
a,b/ε
b,Z/bZ
b,b/bb
b,a/ε

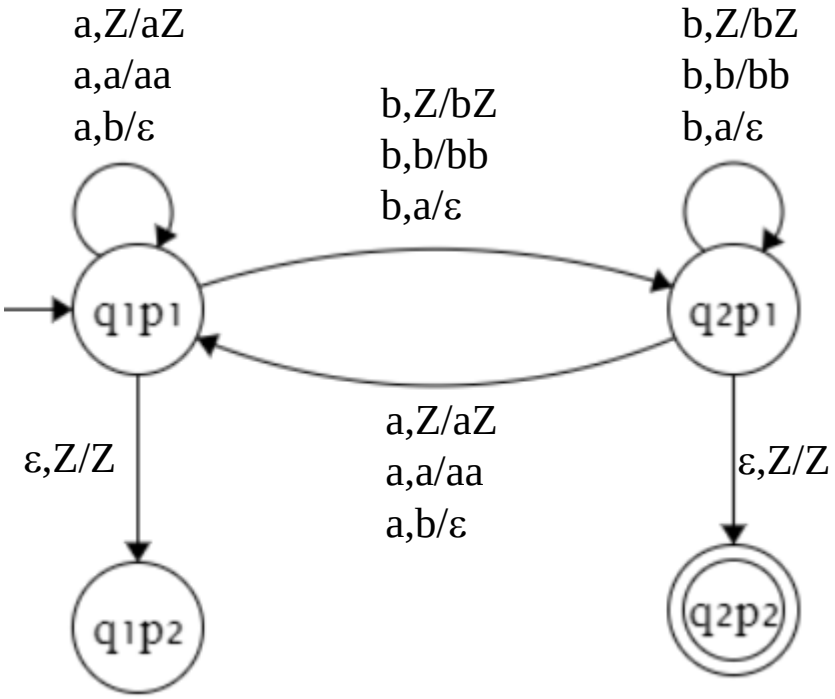
L_1: The number of a's is equal to the number of b's



L_2: Ends with b.



$L_1 \cap L_2$: The number of a's is equal to the number of b's and ends with b.



(CFL – RegularLanguage) is a CFL

- **If L is a CFL and R is a regular language, then L-R is a CFL.**

- Note that $L - R = L \cap \overline{R}$.
- If R is regular, $\overline{R}$ is also regular.
- Then L-R is a CFL.