Non-Context-Free language

Shakir Ullah Shah

Non-Context-Free language

- There arises a question, whether all languages are CFL? The answer is no.
- Languages which are not Context-Free are called Non-CFL.
- To prove the claim that all languages are not Context-Free, the study of machines of word production from the grammar is needed

Terminology

- **Live production**:

  Nonterminal ✉ Nonterminals Nonterminals

- **Dead production**:

  Nonterminal ✉ terminal

- It may be noted that every CFG in CNF has only these types of productions.

Theorem 32

- **If a CFG is in CNF and if there is restriction to use the live production at most once each, then only the finite many words can be generated.**
- It may be noted that every time a live production is applied during the derivation of a word it increases the number of non-terminals by one.
- Similarly applying dead production decreases the non-terminals by one. Which shows that

Example

- S => XY

  => aY

  => ab

Example

- S => XY

  => aY

  => ab

- Here one live and two dead productions are used to generate a word ab.

Proof of Theorem 32

- Suppose we start in some abstract CFG in CNF with

$$S \Rightarrow AB$$

- Suppose we apply the live production $A \rightarrow XY$, we get

$$\Rightarrow XYB$$

which has 3 nonterminals (i.e., one more nonterminal).

- If we now apply the dead production $X \rightarrow b$, we obtain

$$\Rightarrow bYB$$

which has 2 nonterminals (i.e., one less nonterminal).

Proof of Theorem 32

- Thus, every time we apply a live production, we increase the number of nonterminal by one. Every time we apply a dead production, we decrease the number of nonterminals by one.
- Hence, to start from a nonterminal S and eventually arrive at a string of only terminals (i.e., no nonterminals), we must **apply one more dead production than live production**.
- Suppose the the grammar G has exactly p live productions and q dead productions.
- Then, any derivation that does not re-use a live production must have at most p live productions, and accordingly at most $(p + 1)$ dead productions.

Proof of Theorem 32

- Since each letter in the final word comes from one dead production, all words generated from G without repeating any live productions have at most $(p+1)$ letters in them.
- Since no words can be more than $(p+1)$ letters long, there can only be finitely many words.

Theorem 33

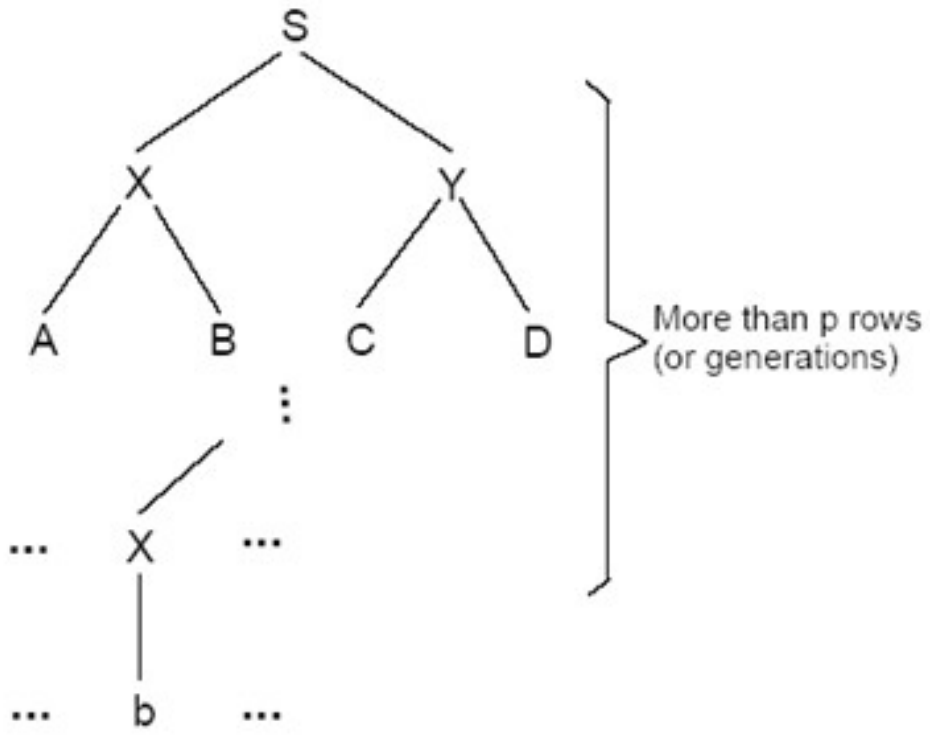
Let G be a CFG in CNF that has p live productions and q dead productions.

If w is a word generated by G that has more than 2^p letters in it, then somewhere in every derivation tree for w, there is some nonterminal (call it Z) being used **twice** where the second Z is **descended** from the first Z.

Proof of Theorem 33

- Since at each row in the derivation tree of w, the number of symbols in the working string can at most double the last row, the condition $length(w) > 2^p$ ensures that the derivation tree for w has **more than** p rows (or generations).
- Let consider any terminal in w, say b, that was formed on the bottom row of the derivation tree by a dead production, say $X \rightarrow b$.
- The letter b was formed after **more than** p rows (or generations) of the tree. This means that letter b has more than p direct ancestors up the tree.
- This is illustrated by the following figure.

Proof of Theorem 33



- From letter b, let us trace our way back to the top letter S. We shall encounter one nonterminal after another. Each nonterminal represents a live production.

Proof of Theorem 33

- Since there are more than p rows to trace, there are more than p productions in the ancestor path from b to S.
- But there are only p different live productions, so some live productions must have been used more than once.
- The nonterminal on the left side of this repeated live production must occurs twice (or more) on the descendant path from S to b. This is the nonterminal that proves our theorem.

Self-embedded nonterminal

In a given derivation of a word in a given CFG, a nonterminal is said to be **self-embedded** if it ever occurs as a tree descendant of itself.

Note:

- Then, Theorem 33 says that in any CFG, all sufficiently long words have leftmost derivation that include a self-embedded nonterminal.

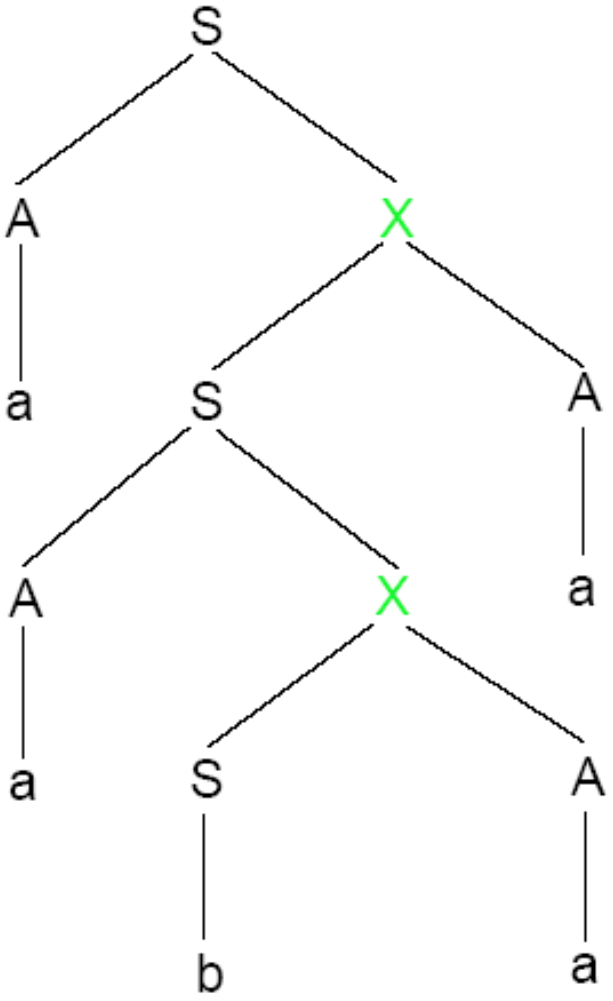
Example

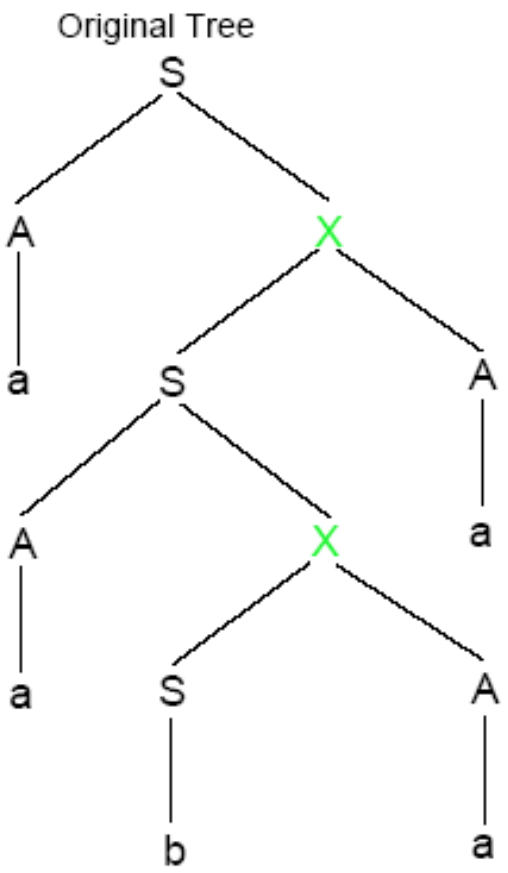
- Consider the following CFG for $NONNULLPALINDROME$ in CNF:

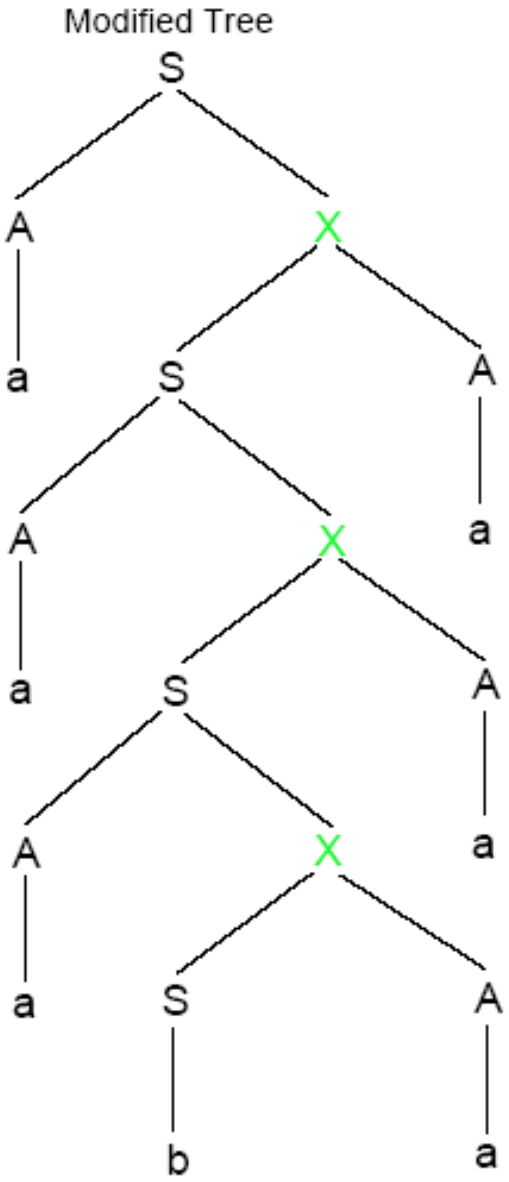
$$S \rightarrow AX|BY|AA|BB|a|b$$
$$X \rightarrow SA$$
$$Y \rightarrow SB$$
$$A \rightarrow a$$
$$B \rightarrow b$$

- There are 6 live productions, so according to Theorem 33, it would require a word of more than $2^6 = 64$ letters to **guarantee** that each derivation has a self-embedded nonterminal.
- However, if we are just looking for one example of a self-embedded nonterminal, we can find such a tree more easily, as shown in the next slide.

- Here the Non-terminal **X** is self-embedded
- This tree proceeds from S down to 1st X and to 2nd X
- But once reached to 2nd X then it could be repeated, arriving at 3rd X e.g




Original Tree
S
A
X
a
S
A
A
X
a
a
S
A
b
a


Modified Tree
S
A
X
a
S
A
A
X
a
a
S
A
A
X
a
S
A
b
a

Notation

Let us introduce the notion $\Rightarrow^*$ to stand for **"can eventually produce"**. It is used in the following context:

Suppose in a certain CFG, the working string S_1 can produce the working string S_2, which in turn can produce the working string S_3, ..., which in turn can produce the working string S_n:

$$S_1 \Rightarrow S_2 \Rightarrow S_3 \Rightarrow ... \Rightarrow S_n$$

Then, we can write

$$S_1 \Rightarrow^* S_n$$

- Using this notation, the following are true in the above CFG:

$$S \Rightarrow^* aX, \quad X \Rightarrow^* aXa, \quad X \Rightarrow^* ba$$

- Since $X \Rightarrow^* aXa$, we can write

$$X \Rightarrow^* aaXaa, \quad X \Rightarrow^* aaaXaaa, \quad \text{and so on}$$

- In general,

$$X \Rightarrow^* a^n X a^n$$

- We can produce words in this CFG, starting with $S \Rightarrow aX$ and finishing with $X \Rightarrow ba$, using the iterations $X \Rightarrow^* a^n X a^n$ in the middle:

$$S \Rightarrow^* aX \Rightarrow^* aaXa \Rightarrow^* aabaa$$
$$S \Rightarrow^* aX \Rightarrow^* aaaXaa \Rightarrow^* aaabaaa$$
$$S \Rightarrow^* aX \Rightarrow^* aaaaXaaa \Rightarrow^* aaaabaaaa$$
$$\ldots$$
$$S \Rightarrow^* aX \Rightarrow^* aa^nXa^n \Rightarrow^* aa^nbaa^n$$

- Hence, given any derivation tree in any CFG with a self-embedded nonterminal, we can use this iteration trick to produce an **infinite** family of other words in the language.

The Pumping Lemma for CFLs: Theorem 34

If G is any CFG in CNF with p live productions, and w is any word generated by G with length greater than 2^p, then we can break up w into five substrings:

$$w = uvxyz$$

such that x is not $\mathbf{\Lambda}$, and v and y are not both $\mathbf{\Lambda}$, and such that all the words

$$uvxyz, \quad uvvxyyz, \quad uvvvxyyyz, \quad \ldots$$

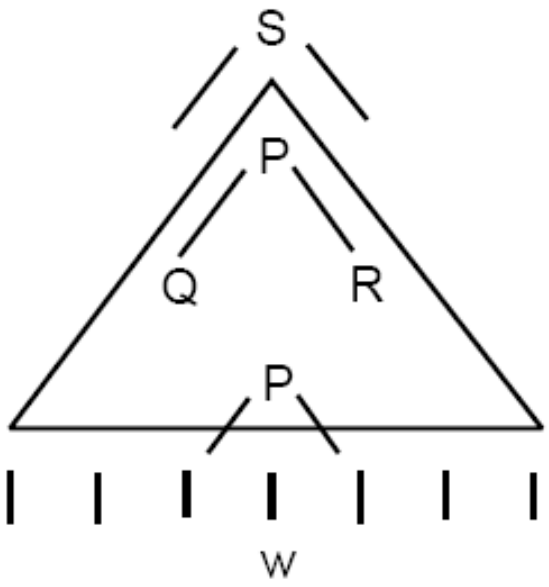
or in general,

$$uv^n xy^n z \quad \text{for } n = 1, 2, 3, \ldots$$

can also be generated by G.

Proof of Theorem 34

- By Theorem 33, if $length(w) > 2^p$ then there are self-embedded nonterminals in any derivation for w.
- Consider one specific derivation of w in G. Let's call one self-embedded nonterminal P, whose first production is $P \rightarrow QR$. Suppose the tree looks like this:

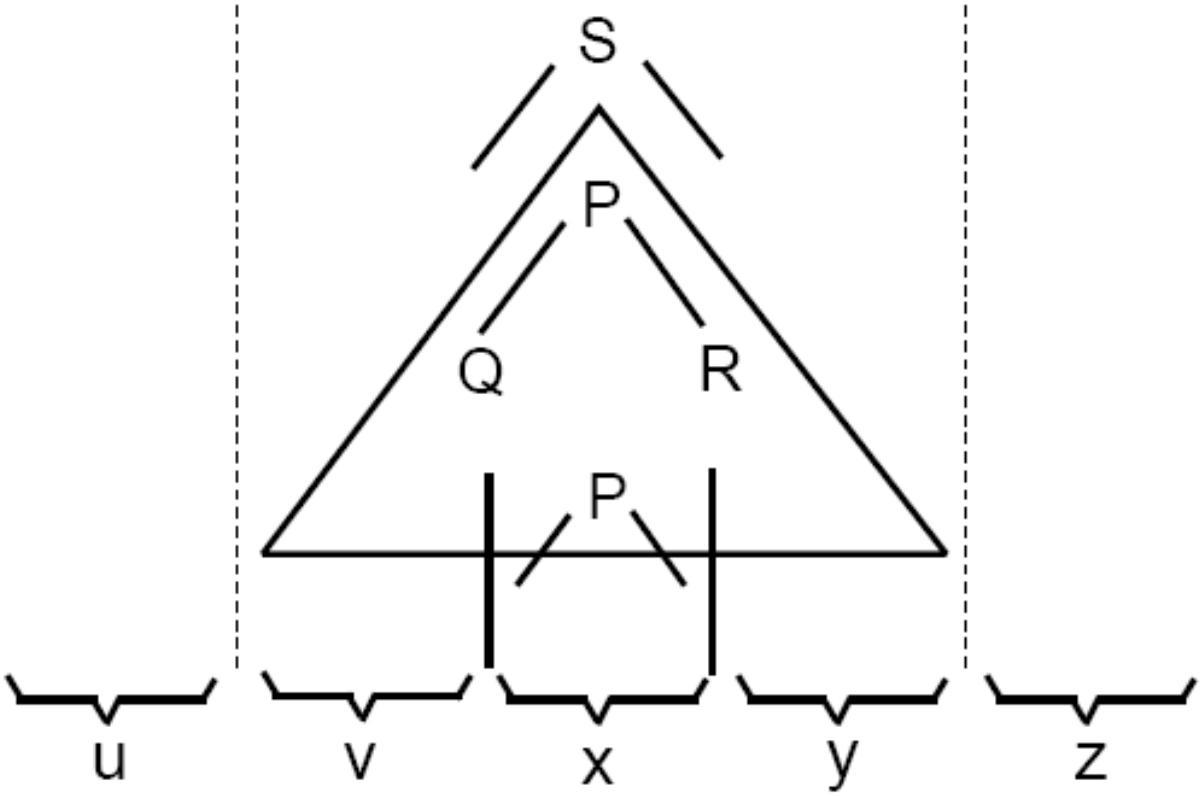


Proof of Theorem 34 (cont.)

- The triangle encloses the part of the tree generated from the first P down to where the second P is produced.
- Let's divide w into 5 parts:
  1. u = the substring of w generated to the left of the triangle (this may be $\mathbf{\Lambda}$).
  2. v = the substring of all the letters of w descended from the first P but to the left of the letters generated by the second P (this may be $\mathbf{\Lambda}$).
  3. x = the substring of w descended from the second P (this cannot be $\mathbf{\Lambda}$ because this nonterminal must turn into some terminals).
  4. y = the substring of w generated by the first P but to the right of the letters descended from the second P (this may be $\mathbf{\Lambda}$, but not if $v = \mathbf{\Lambda}$, as we shall see).
  5. z = the substring of w generated to the right of the triangle (this may be $\mathbf{\Lambda}$).

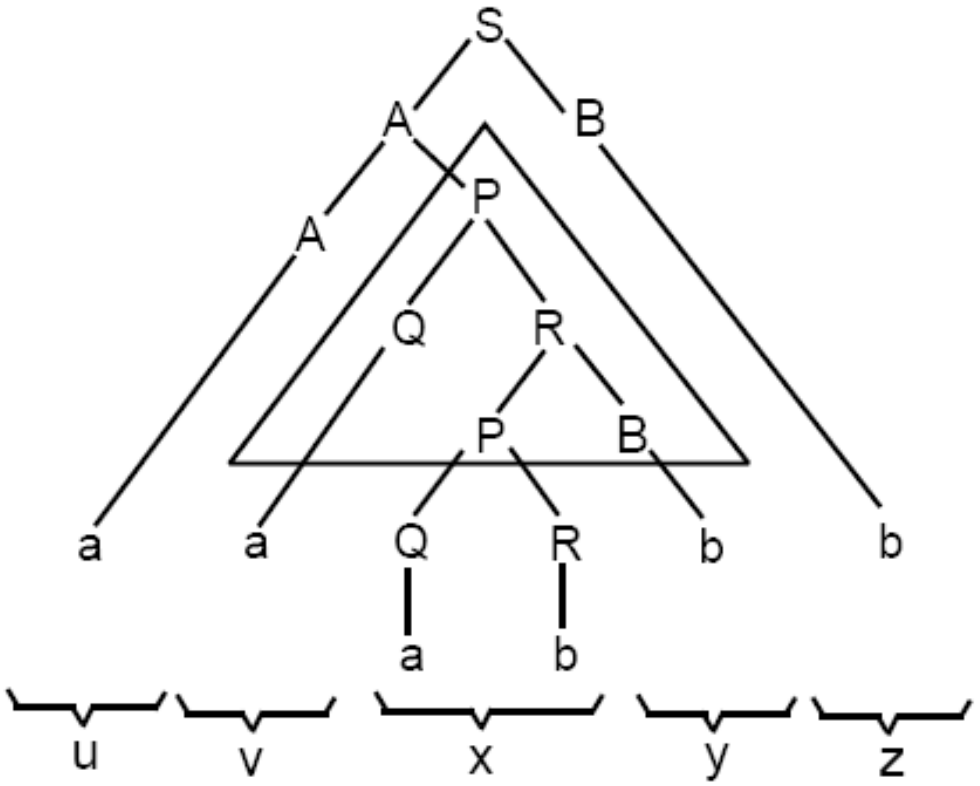
Proof of Theorem 34 (cont.)

- The following figure illustrate the 5 parts of w:



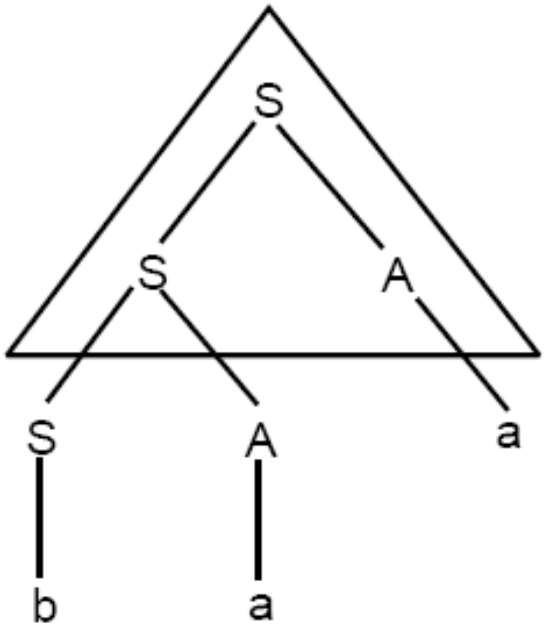
Proof of Theorem 34 (cont.)

- For example, this is a complete tree in an unspecified grammar:



Proof of Theorem 34 (cont.)

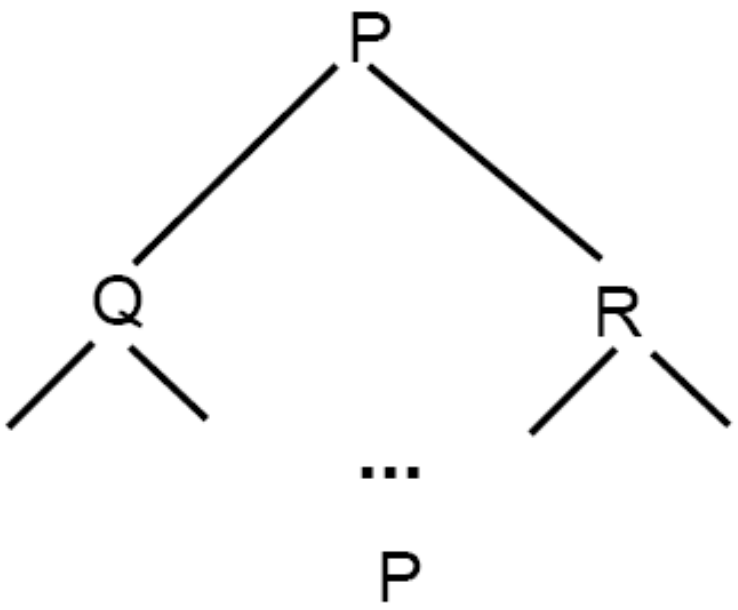
- It is possible that either u or z or both may be Λ, as in the following figure where S is the self-embedded nonterminal, and all the letters of w are generated inside the triangle:



u = Λ v = Λ x = ba y = a z = Λ

Proof of Theorem 34 (cont.)

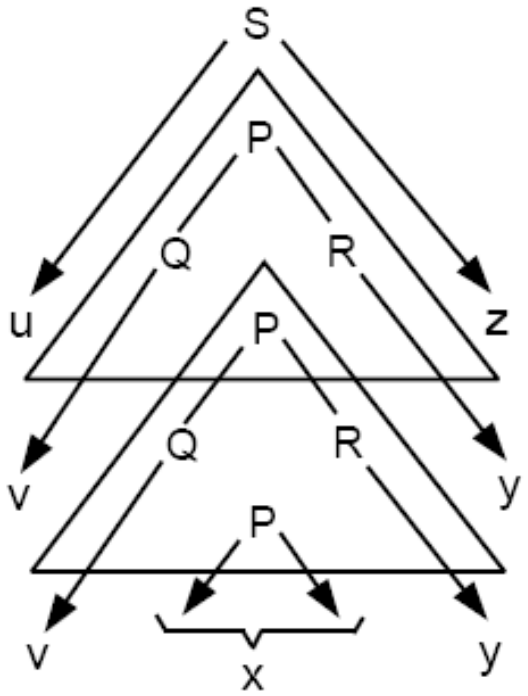
- However, either v is not Λ, y is not Λ, or both are not Λ.
- This is because in the figure



although the lower P can come from the upper Q or from the upper R, there must still be some other letters in w that come from the other branch (i.e., the branch that does not produce the lower P).

Proof of Theorem 34 (cont.)

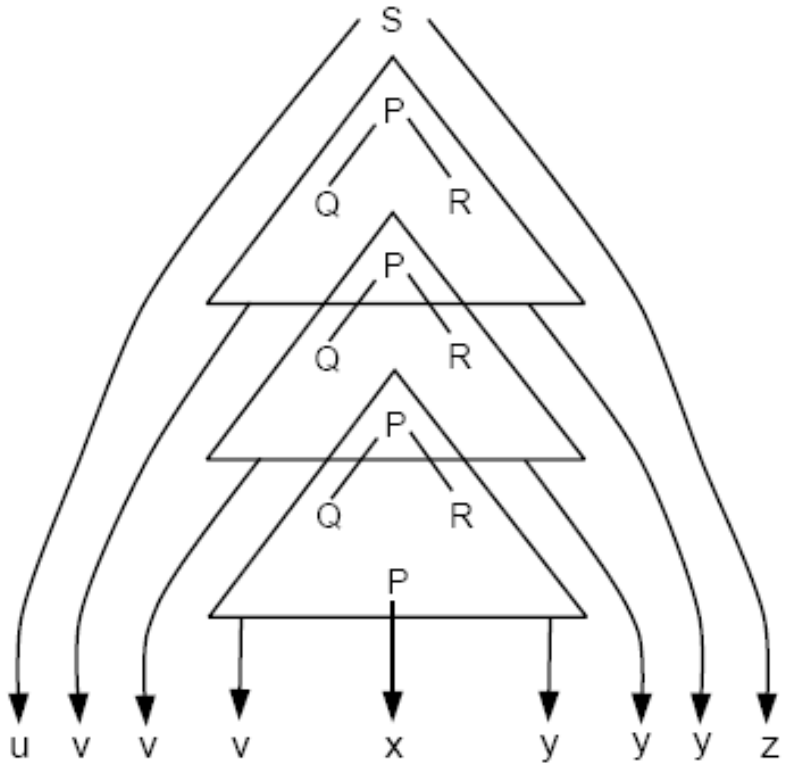
- If we are now iterating the triangle, what would be the end word? In particular, consider this doubled tree:



- As we can see from the figure, we shall generate the word **uvvxyyz**, which must be in the language defined by G.

Proof of Theorem 34 (cont.)

- If we triple the triangle



- We would have a derivation tree for the word uvvvxyyyz, which must be in the language generated by G.

Proof of Theorem 34 (cont.)

- In general, if we repeat the triangle n times, we get a derivation tree for the word

$$uv^nxy^nz$$

which must be in the language generated by G.

- The proof is therefore complete.

Definition (Second)

In a particular CFG, a nonterminal N is called **self-embedded** in the derivation of a word w if there are strings of terminals v and y **not both null**, such that

$$N \Rightarrow^* vNy$$

Proof 2 of Theorem 34

- If P is a self-embedded nonterminal in the derivation of w then

$$S \Rightarrow^* uPz$$

for some u and z, both substrings of w.

- Also, since P is self-embedded, we have

$$P \Rightarrow^* vPy$$

for some substrings v and y of w, not both null.

- Finally,

$$P \Rightarrow^* x$$

where x is another substring of w.

Proof 2 of Theorem 34 (cont.)

- But we can also write

$$
\begin{aligned}
S &\Rightarrow^* uPz \\
&\Rightarrow^* uvPyz \\
&\Rightarrow^* uvvPyyz \\
&\Rightarrow^* uvvvPyyyz \\
&\Rightarrow^* uv^nPy^nz \quad \text{for any } n \\
&\Rightarrow^* uv^nxy^nz
\end{aligned}
$$

- Therefore, the last set of strings are all words derivable in the original CFG.

Example

Show that the language

$$\{a^n b^n a^n \quad \text{for } n = 1, 2, 3, ...\}$$

is not a context-free language.

Solution:

- Assume that this language were a CFL and could be generated by some CFG in CNF.
- Let w be a word with length greater than 2^p where p is the number of live productions of this CFG.

- We will show that *any* method of breaking w into 5 parts

$$w = uvxyz$$

will mean that

$$uv^2xy^2z$$

cannot be in $\{a^nb^na^n\}$.

- All words in $\{a^nb^na^n\}$ must have exactly one occurrence of the substring ab. If either the v-part or the y-part of w has the substring ab in it, then

$$uv^2xy^2z$$

will have more than one substring ab, and so it cannot be in $\{a^nb^na^n\}$. Therefore, neither v nor y contains ab.

- All words in $\{a^n b^n a^n\}$ must have exactly one occurrence of the substring ba. If either the v-part or the y-part of w has the substring ab in it, then

$$uv^2xy^2z$$

will have more than one substring ba, and so it cannot be in $\{a^n b^n a^n\}$. Therefore, neither v nor y contains ba.

- The only possibility left is that v and y must be all a's, all b's, or $\mathbf{\Lambda}$.

- But if v or y or both are blocks of one letter, then

$$uv^2xy^2z$$

has increased one or two clumps of solid letters a's or b's, but **not equally increased all three clumps** of solid letters in the word $a^n b^n a^n$.

- Thus, any attempt to partition $w \in \{a^n b^n a^n\}$ into $uvxyz$ according to Theorem 34 must fail to have $uvvxyyz$ in the language.
- Note that if v and y are both Λ, then the partition also fails, because Theorem 34 requires that v and y are **not both** Λ.
- Since Theorem 34 cannot be applied to the language $\{a^n b^n a^n\}$, this language is not a context-free language.

Theorem 35

Let L be a CFL in CNF with p live productions.

Then, an word w in L with length $> 2^p$ can be broken into 5 parts:

$$w = uvxyz$$

such that

$$length(vxy) \leq 2^p$$

$$length(x) > 0$$

$$length(v) + length(y) > 0$$

and such that all the words

$$uv^n xy^n z \quad \text{for } n = 1, 2, 3, \ldots$$

are in the language L.

- This is just another version of Theorem 34.

Another Way to Prove

The language {*anbncn*} is not a CFL.

- Proof: let $G = (V, \Sigma, S, P)$ be any CFG, $\Sigma = \{a,b,c\}$
- Suppose by way of contradiction that $L(G) = \{anbncn\}$ then G generates a pumping parse tree and for some k, $akbkck = uvwxy$, where v and x are not both ε and $uv2wx2y$ is in $L(G)$
- v and x must each contain only as, only bs, or only cs; otherwise $uv2wx2y$ is not even in $L(a^*b^*c^*)$
- So $uv2wx2y$ has more than k copies or one or two symbols, but only k of the third
- $uv2wx2y \notin \{anbncn\}$; by contradiction, $L(G) \neq \{anbncn\}$

The Insight

- There must be some string in $L(G)$ with a pumping parse tree: $akbkck = uvwxy$
- But no matter how you break up $akbkck$ into those substrings $uvwxy$ (where v and x are not both ε) you can show $uv^2wx^2y \notin \{anbncn\}$
- Either:
  - v or x has more than one kind of symbol
  - v and x have at most one kind of symbol each

- If *v* or *x* has more than one kind of symbol:
  - *uv2wx2y* would have *a*s after *b*s and/or *b*s after *c*s
  - Not even in $L(a^*b^*c^*)$, so certainly not in {*anbncn*}
  - Example:

$$\underbrace{a\ a}_{u}\ \underbrace{a\ a\ a}_{v}\ \underbrace{b\ b}_{w}\ \underbrace{b\ b\ b}_{x}\ \underbrace{c\ c\ c\ c}_{y}$$

- If *v* and *x* have at most one kind each:
  - *uv2wx2y* has more of one or two, but not all three

$$\underbrace{a\ a}_{u}\ \underbrace{a\ a\ a}_{v}\ \underbrace{b\ b}_{w}\ \underbrace{b\ b\ b}_{x}\ \underbrace{b\ c\ c\ c\ c\ c}_{y}$$

{*anbncn*} Is Not Context Free

1. Proof is by contradiction using the pumping lemma for context-free languages. Assume that $L = \{anbncn\}$ is context free, so the pumping lemma holds for L. Let k be as given by the pumping lemma.

2. Choose $z = akbkck$. Now $z \in L$ and $|z| \geq k$ as required.

3. Let u, v, w, x, and y be as given by the pumping lemma, so that $uvwxy = akbkck$, v and x are not both ε, $|vwx| \leq k$, and for all i, $uviwxiy \in L$.

4. Now consider pumping with $i = 2$. The substrings v and x cannot contain more than one kind of symbol each—otherwise the string $uv2wx2y$ would not even be in $L(a^*b^*c^*)$. So the substrings v and x must fall within the string $akbkck$ in one of these ways…

{*anbncn*}, Continued

	a^k	b^k	c^k
1.	*v* *x*		
2.	*v*	*x*	
3.		*v* *x*	
4.		*v*	*x*
5.			*v* *x*
6.	*v*		*x*

But in all these cases, since *v* and *x* are not both ε, pumping changes the number of one or two of the symbols, but not all three. So *uv2wx2y* ∉ *L*.

This contradicts the pumping lemma. By contradiction, *L* = {*anbncn*} is not context free.

The language $\{anbmcn \mid m \leq n\}$ is not context free.

- Proof: by contradiction using the pumping lemma
- Assume $L = \{anbmcn \mid m \leq n\}$ is a CFL
- Let k be as given by the pumping lemma
- Choose $z = akbkck$, so we have $z \in L$ and $|z| \geq k$
- Now $uvwxy = akbkck$, v and x are not both ε, $|vwx| \leq k$, and for all i, $uviwxiy \in L$
- Now consider pumping with $i = 2$
- v and x cannot contain more than one kind of symbol each; otherwise $uv2wx2y \notin L(a^*b^*c^*)$
- That leaves 6 cases…

	a^k	b^k	c^k
1.	v x		
2.	v	x	
3.		v x	
4.		v	x
5.			v x
6.	v		x

- But cases 1-5 have $uv^2wx^2y \notin L$:
  - Case 1 has more *a*s than *c*s
  - Case 2 has more *a*s than *c*s, or more *b*s than *c*s, or both
  - Case 3 has more *b*s than *a*s and more *b*s than *c*s
  - Case 4 has more *b*s than *a*s, or more *c*s than *a*s, or both
  - Case 5 has more *c*s than *a*s and more *c*s than *b*s
- And case 6 contradicts $|vwx| \leq k$
- By contradiction, $L = \{a^n b^m c^n \mid m \leq n\}$ is not a CFL

$A = \{0^n 1^n 2^n \mid n \geq 0\}$ is not context free.

Proof Assume, to the contrary, that A is context free. By Pumping Lemma there exists a constant p such that every $w \in A$ of length $\geq p$ is divided into $w = uvxyz$ such that $|vxy| \leq p$, $|vy| \geq 1$, and for every $i \geq 0$, $uv^i xy^i z \in A$.

Let $w = 0^p 1^p 2^p$. Since $|vxy| \leq p$, vxy is either in 0^*1^* or 1^*2^*. So it is not the case uv^2xy^2z has the same number of 0s, 1s, as 2s.

$C = \{ww \mid w \in \{0,1\}^*\}$ is not context free.

Proof Assume C is context free. Let p the constant from the pumping lemma for C.

Let $w = 0^p1^p0^p1^p$, which is in C.

Let $w = uvxyz$ be the decomposition of w such that $|vy| > 0$, $|vxy| \leq p$, and for every $i \geq 0$, $uv^ixy^iz \in C$.

If v contains a symbol from the first 0^p then y cannot contain one from the second 0^p, so pumping doesn't work. If v contains only symbols from the first 1^p then y cannot contain one from the second 1^p, so pumping doesn't work. If v contains only symbols from the second 0^p1^p then pumping does not work.

Example

- The following language cannot be shown to be non-context-free by Theorem 34:

$$L = \{a^n b^m a^n b^m\}$$

  where n and m are integers $1, 2, 3, ...$, and n does not necessarily equal to m.

- However, we can use Theorem 35 to show that this language is non-context-free.

- Can you do it?