Theory of Automata
Self Embedded-ness
Pumping Lemma for CFG

Week 11

Contents

- Self Embedded-ness
- Pumping Lemma for CFG

Self-Embeddedness

- The most important question about context-free languages is **whether or not all possible languages are context-free**.
- The answer is **no**. We are going to prove this in this chapter.
- Recall that if a CFG is in Chomsky Normal Form (CNF), then all its productions are of the two forms

$$\text{Nonterminal} \rightarrow \text{Nonterminal Nonterminal}$$

or

$$\text{Nonterminal} \rightarrow \text{terminal}$$

Theorem 32

Let G be a CFG in CNF. Let us call the productions of the form

$$\text{Nonterminal} \rightarrow \text{Nonterminal Nonterminal}$$

live productions.

Let us call the productions of the form

$$\text{Nonterminal} \rightarrow \text{terminal}$$

dead productions.

If we are restricted to using the live productions **at most once each**, we can generate **only finitely many words**.

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 (cont.)

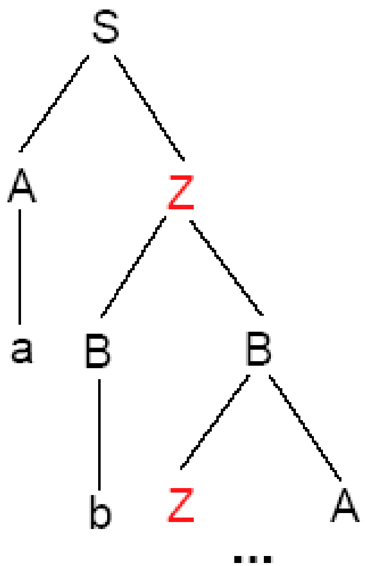
- Thus, every time we apply a live production, we increase the number of nonterminal by one. Every time we apply a dead production, we decrease the number of nonterminals by one.
- Hence, to start from a nonterminal S and eventually arrive at a string of only terminals (i.e., no nonterminals), we must **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.

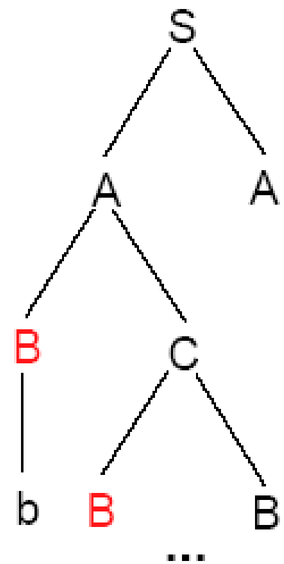
Proof of Theorem 32 (cont.)

- 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.

Notes

- We say that Y is **descended from** X (or Y is a **tree descendant** of X) if there is a **downward path** from X to Y in the derivation tree.





- In the left figure, the second Z is descended form the first Z, whereas in the right figure the second B is not descended from the first B.

- Since a grammar in CNF replaces every nonterminal with one or two symbols, the derivation tree of each word has the property that every node has one or two immediate descendants. Such a tree is called a **binary tree**.

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

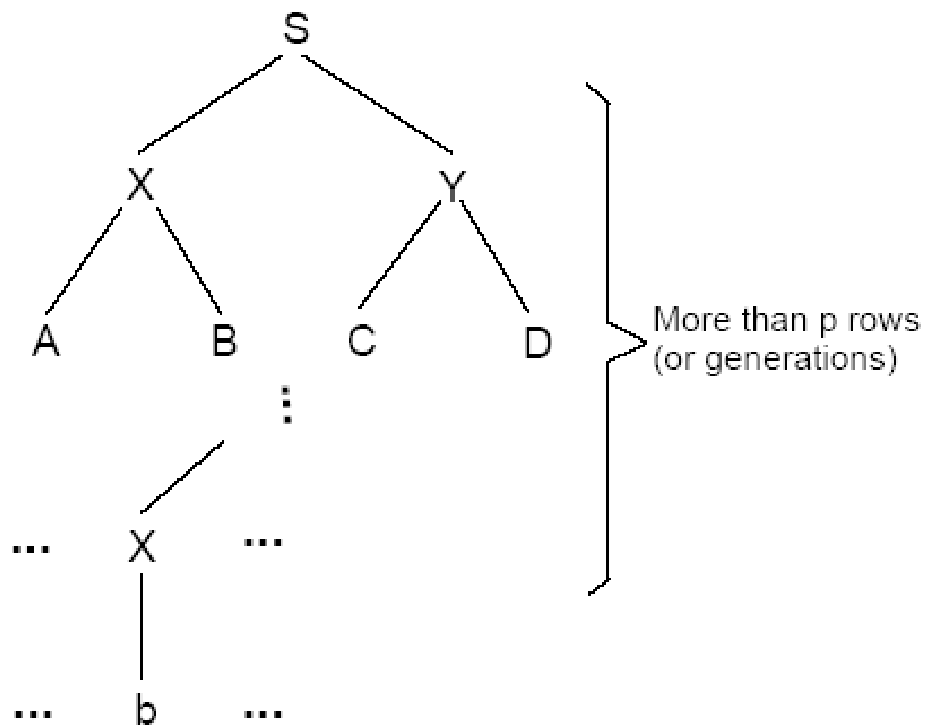
- Since at each row in the derivation tree of w, the number of symbols in the working string can at most double the last row, the condition $length(w) > 2^p$ ensures that the derivation tree for w has **more than** p rows (or generations).

- Let consider any terminal in w, say b, that was formed on the bottom row of the derivation tree by a dead production, say $X \to b$.

- 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 (cont.)



- 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 (cont.)

- 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.

Definition

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.

<u>Note:</u>

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

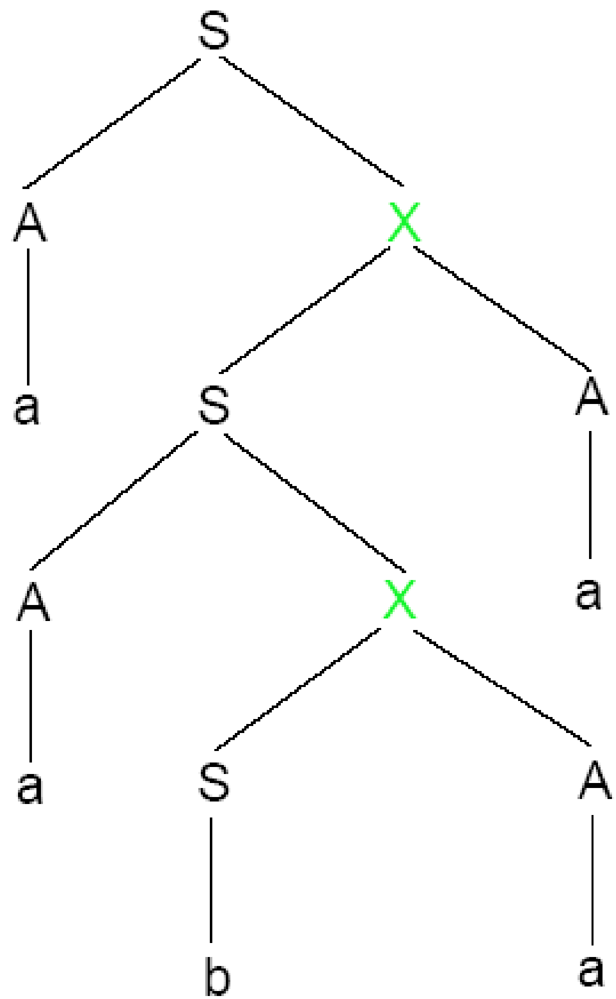
Example

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

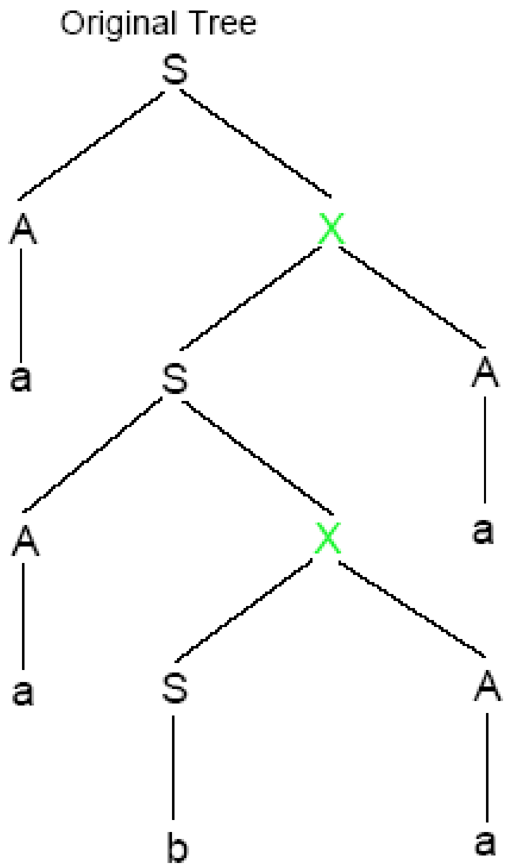
$$S \rightarrow AX|BY|AA|BB|a|b$$
$$X \rightarrow SA$$
$$Y \rightarrow SB$$
$$A \rightarrow a$$
$$B \rightarrow b$$

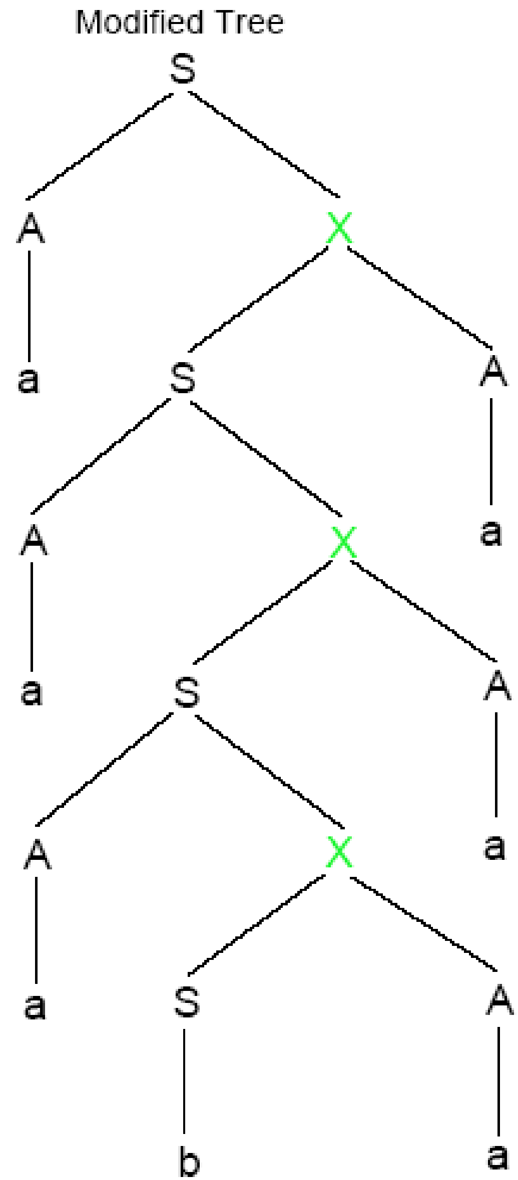
- There are 6 live productions, so according to Theorem 33, it would require a word of more than $2^6 = 64$ letters to **guarantee** that each derivation has a self-embedded nonterminal.

- 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.


S
A
X
a
S
A
A
X
a
a
S
A
b
a

- In the above tree, X is a self-embedded nonterminal.
- The tree above proceeds from S down to the first X, then to the second X, and to the final word.
- But once we have reached the second X, we could have repeated the same sequence of productions that the first X initiated, therefore arriving at a third X. From this third X, we can produce a string of all terminals just as the second X used to.
- This is illustrated by the following figure:


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
a
S
A
b
a

Definition

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:

$$S_1 \Rightarrow S_2 \Rightarrow S_3 \Rightarrow \ldots \Rightarrow S_n$$

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 Λ, and v and y are not both Λ, and such that all the words

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

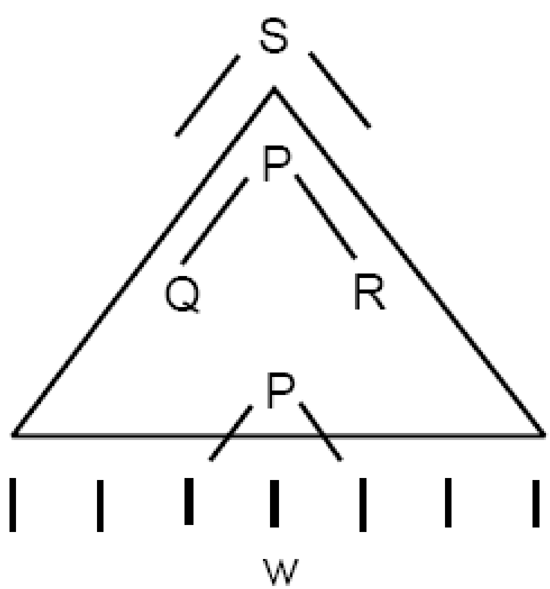
or in general,

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

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:

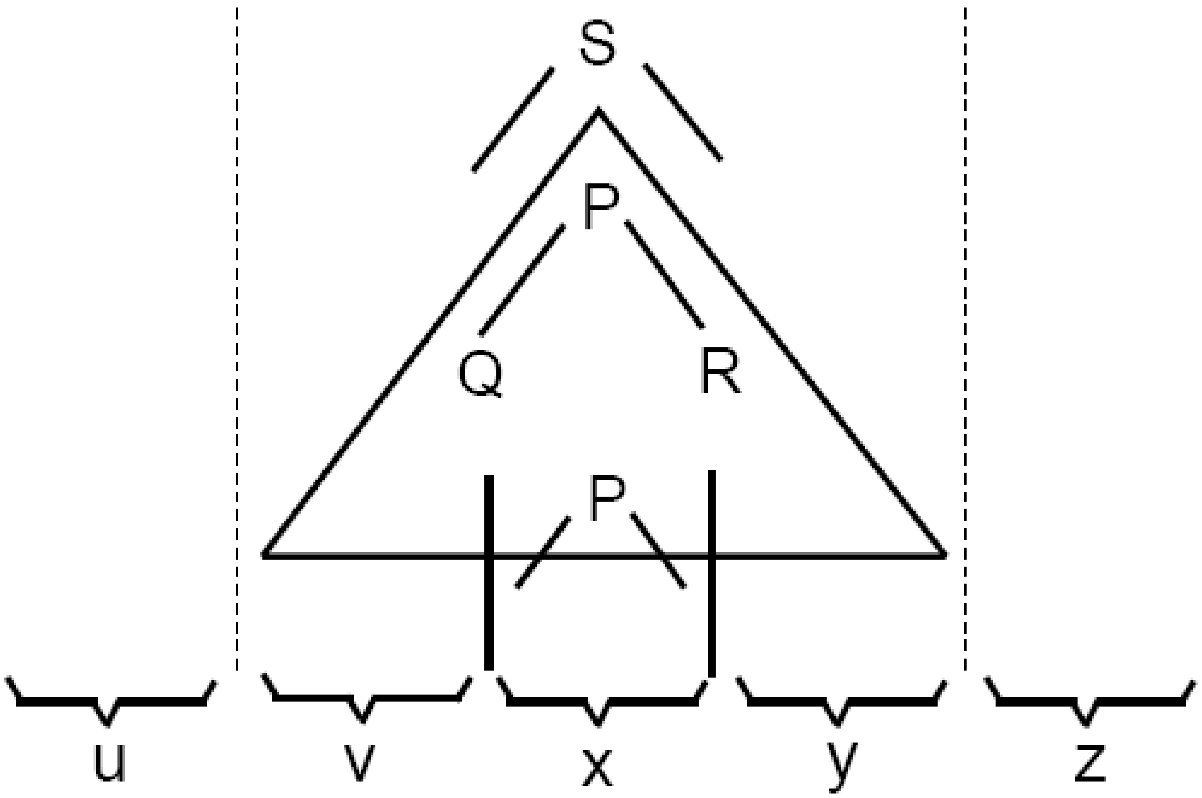


Proof of Theorem 34 (cont.)

- 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 $\mathbf{\Lambda}$).

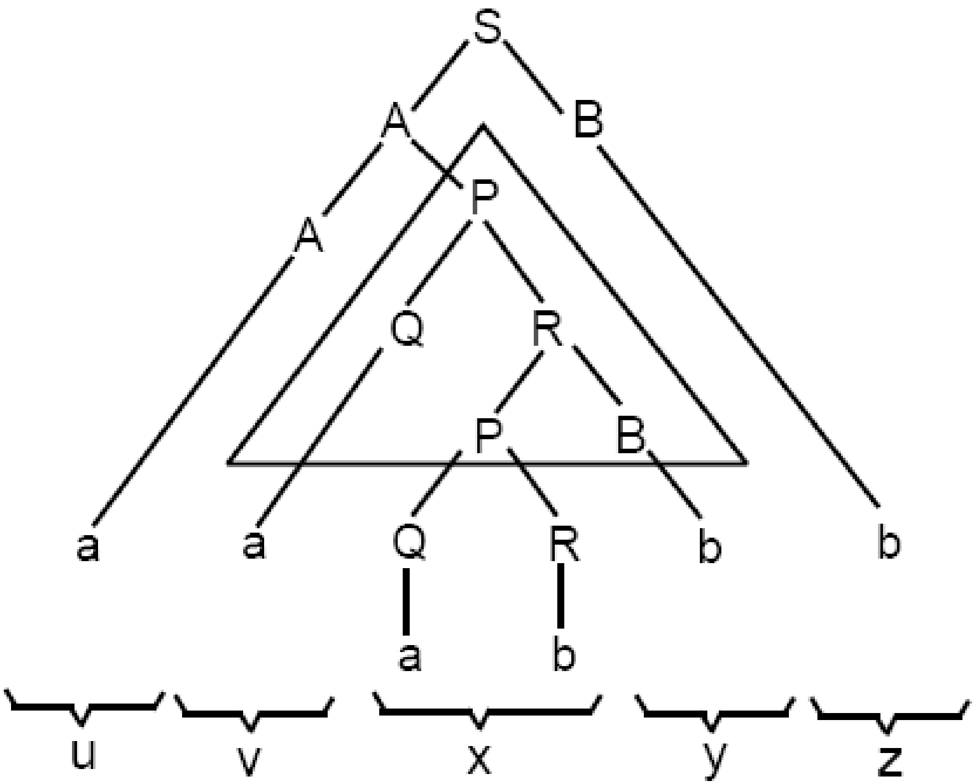
Proof of Theorem 34 (cont.)

- The following figure illustrate the 5 parts of w:



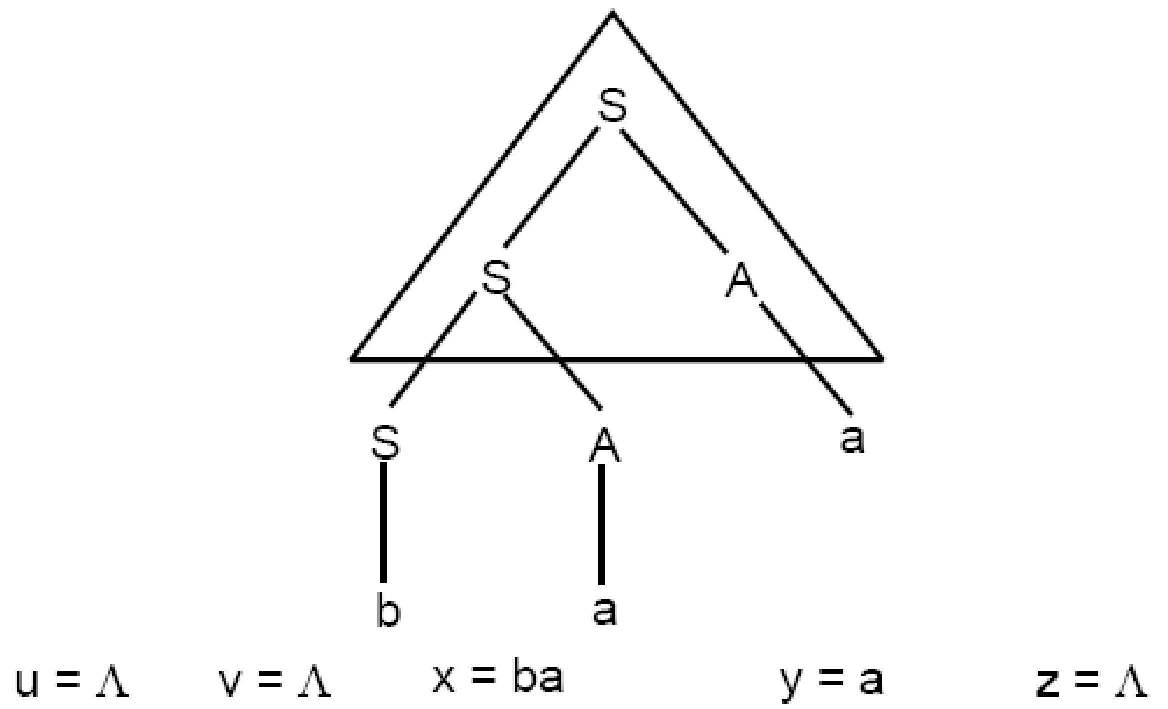
Proof of Theorem 34 (cont.)

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



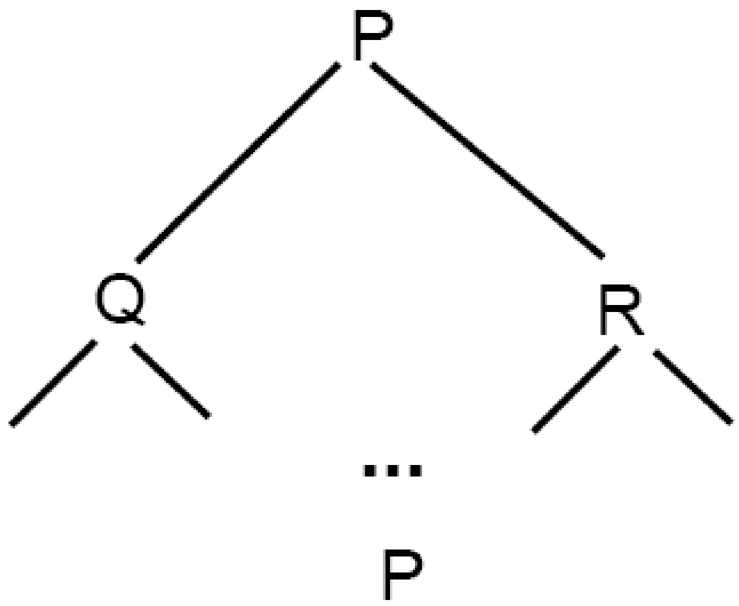
Proof of Theorem 34 (cont.)

- 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:



Proof of Theorem 34 (cont.)

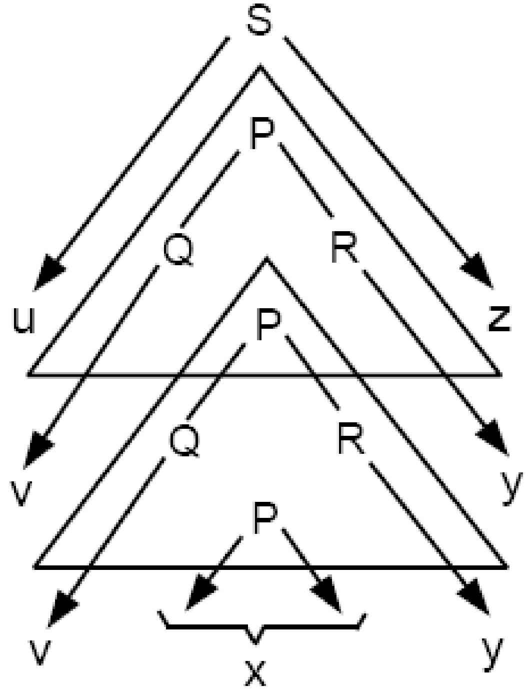
- However, either v is not Λ, y is not Λ, or both are not Λ.
- This is because in the figure



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.)

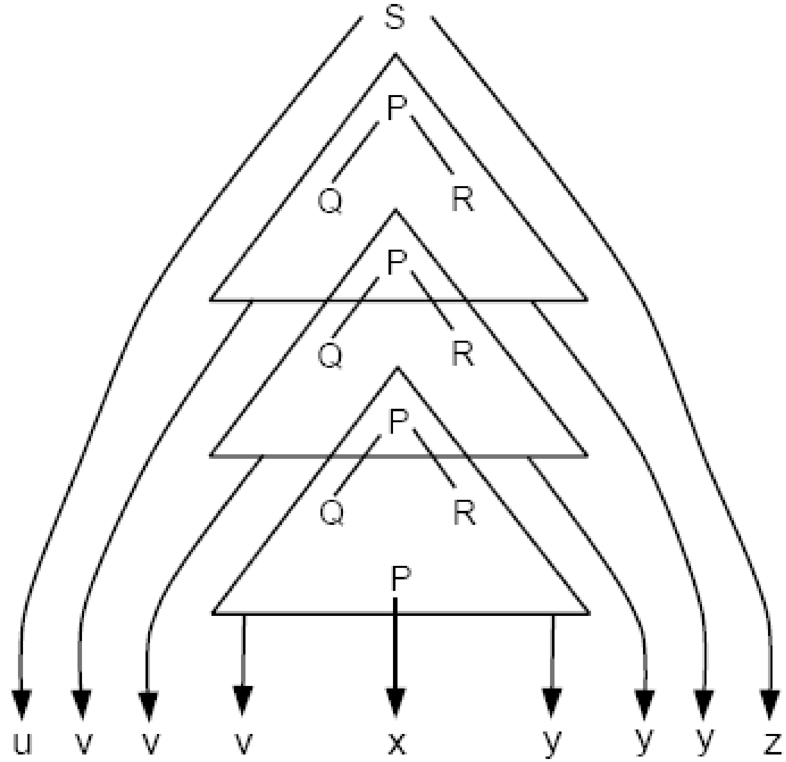
- If we are now iterating the triangle, what would be the end word? In particular, consider this doubled tree:



- As we can see from the figure, we shall generate the word $\mathbf{uvvxyyz}$, 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

$$\mathrm{uv^n xy^n z}$$

  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 $\{a^n b^n a^n\}$.

- All words in $\{a^n b^n a^n\}$ 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 $\{a^n b^n a^n\}$. 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 Λ.

- 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** $\boldsymbol{\Lambda}$.

- 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.
- See the short discussion at the top of page 371 for a proof.

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 $1, 2, 3, \ldots$, 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?

Check for following CFGs

- CFL
  - $a^n b^n$
  - $a^n b^n c^m$
- Non-CFL
  - $a^n b^n a^n b^n$
  - $a^n b^{2n} a^n$
  - $a^n b^n c^n d^n$