Monash University
Faculty of Information Technology

# Lecture 13
# Chomsky Normal Form

Slides by David Albrecht (2011), some additions and modifications by Graham Farr (2013).

FIT2014 Theory of Computation

# Overview

- Chomsky Normal Form
- CYK Parsing algorithm
- Pumping Lemma

# Chomsky Normal Form

A **CFG** is said to be in **Chomsky Normal Form** if all the productions are in the form

**Nonterminal → Nonterminal Nonterminal**

(called a *live production*)

or

**Nonterminal → terminal**

(called a *dead production*)

For **any** context-free language **L**, the non-empty words in **L** can be generated by a grammar in **Chomsky Normal Form**.

# Proof

- First eliminate all **ε productions**.
  - may need some new productions, to keep the effect of an empty production
- For each terminal, **a**, ensure that there is a production of the form **A → a,** and replace **a** in all other productions by **A**.
  - (Note **A** may need to be a new non-terminal symbol.)
- Eliminate all **unit productions**.
- For any production that has more than 2 non-terminals on the right-hand side, split them into a sequence of productions that have only 2 non-terminals on the right-hand side.

- Consider the CFG

  $S \rightarrow bA \mid aB$

  $A \rightarrow a \mid aS \mid bAA$

  $B \rightarrow b \mid bS \mid aBB$

- Then:

  $S \rightarrow B_1A \mid A_1B$

  $A \rightarrow a \mid A_1S \mid B_1AA$ $\qquad A_1 \rightarrow a$

  $B \rightarrow b \mid B_1S \mid A_1BB$ $\qquad B_1 \rightarrow b$

- Then:

  $S \rightarrow B_1A \mid A_1B$

  $A \rightarrow a \mid A_1S \mid B_1R_1$ $\qquad A_1 \rightarrow a$

  $B \rightarrow b \mid B_1S \mid A_1R_2$ $\qquad B_1 \rightarrow b$

  $R_1 \rightarrow AA$

  $R_2 \rightarrow BB$

# Consequences

- **Cocke-Younger-Kasami (CYK) algorithm**
  - For each CFG and string **s**, we can decide whether or not **s** is generated by the CFG.
  - bottom-up parsing
- **Pumping Lemma for CFG**
  - is used to show there exists non-context free languages.

# CYK Algorithm

For each CFG and string **s**, we can decide whether or not **s** is generated by the CFG.

# Proof (re: CYK Algorithm)

Idea of Proof:

- Find the Chomsky Normal Form for the non-empty words generated by the grammar.
- Add $\mathbf{S} \rightarrow \boldsymbol{\varepsilon}$ if $\boldsymbol{\varepsilon}$ can be generated by the CFG.
- If $\mathbf{s} = \boldsymbol{\varepsilon}$ then state that $\mathbf{s}$ can be generated if and only if $\mathbf{S} \rightarrow \boldsymbol{\varepsilon}$ is a production.
- Assume $\mathbf{s} = \mathbf{t_1}\ \mathbf{t_2} \ldots \mathbf{t_n}$ is non-empty.
- For each letter $\mathbf{t_k}$ find the non-terminals which can produce $\mathbf{t_k}$
- For each of the following pairs:

  $$\mathbf{t_1\ t_2\ ,\ t_2\ t_3\ ,\ ..,\ t_{n-1}\ t_n}$$

  find the non-terminals that can produce the pair.
- For each of the following triples:

  $$\mathbf{t_1\ t_2\ t_3,\ t_2\ t_3\ t_4\ ,\ ..,\ t_{n-2}\ t_{n-1}\ t_n}$$

  find the non-terminals that can produce the triple.
- Continue, in this way to find the list of non-terminal that can produce $\mathbf{s} = \mathbf{t_1}\ \mathbf{t_2} \ldots \mathbf{t_n}$ . If $\mathbf{S}$ is one of the non-terminals then $\mathbf{s}$ can be generated, otherwise $\mathbf{s}$ cannot be generated.

# CYK Algorithm

Exercises:

Turn this sketch proof into a correct proof by induction.

Determine the complexity of the algorithm, in big-O notation.

# Pumping Lemma

- Let **G** be any CFG in CNF with k non-terminal symbols and **w** is any word generated by **G** with length greater than $\mathbf{2^{k-1}}$.
- Then there exists strings **u**, **v**, **x**, **y**, and **z** such that
  - **w = uvxyz**
  - **v** and **y** are not both **ε**,
  - $|\mathbf{vxy}| \leq 2^k$, and
  - $\mathbf{uv^2xy^2z, \ldots, uv^nxy^nz, \ldots}$ are generated by **G**.

# Proof

Chomsky Normal Form: derivation tree is *binary*.

If max path length of binary tree = $\ell$,

then # leaves $\leq 2^{\ell}$ .

In derivation tree for Chomsky Normal Form, leaves correspond to letters (terminal symbols), and each leaf is an only child (i.e., its parent has degree 2 in the graph, and has no other children).

- Terminals come only from productions of the form

  Non-terminal $\rightarrow$ terminal

So actually # leaves $\leq 2^{\ell\text{-}1}$ .

# Proof

Put $k$ = # non-terminal symbols.

Let $w$ be any string of length $> 2^{k-1}$.

Longest path in derivation tree for $w$ has length $\geq k+1$.

If longest path had length $\leq k$, then # leaves $\leq 2^{k-1}$ (see previous slide), so $|w| \leq 2^{k-1}$, since leaves correspond to terminals.

Now, recall: for a path, # nodes = length + 1.

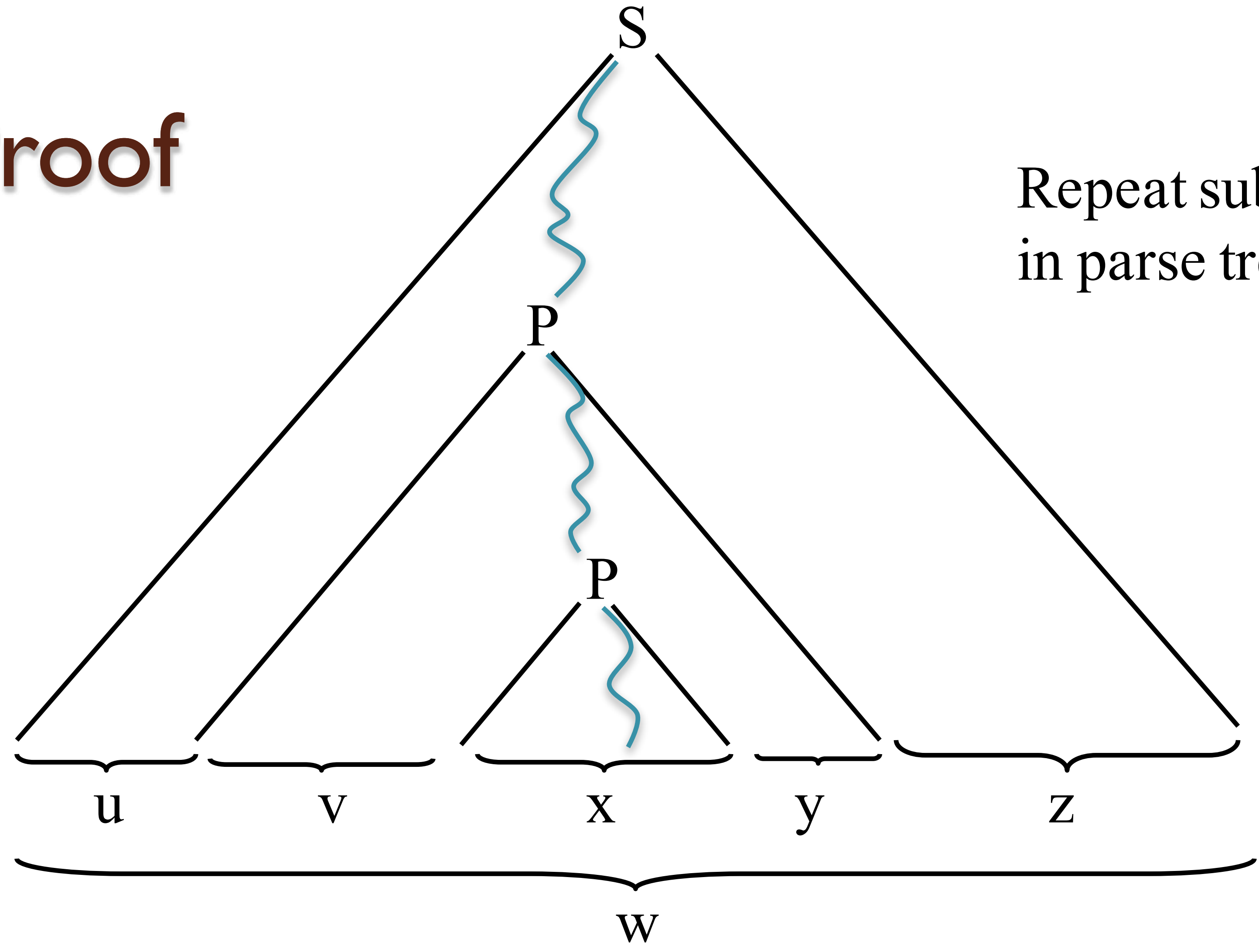
So # nodes in longest path $\geq k+2$.

The last of these is a leaf (terminal node). All the others are non-terminals. So the path has $\geq k+1$ non-terminals.

But the number of different non-terminals is $\leq k$.

So the path contains two identical non-terminals. Call them P.

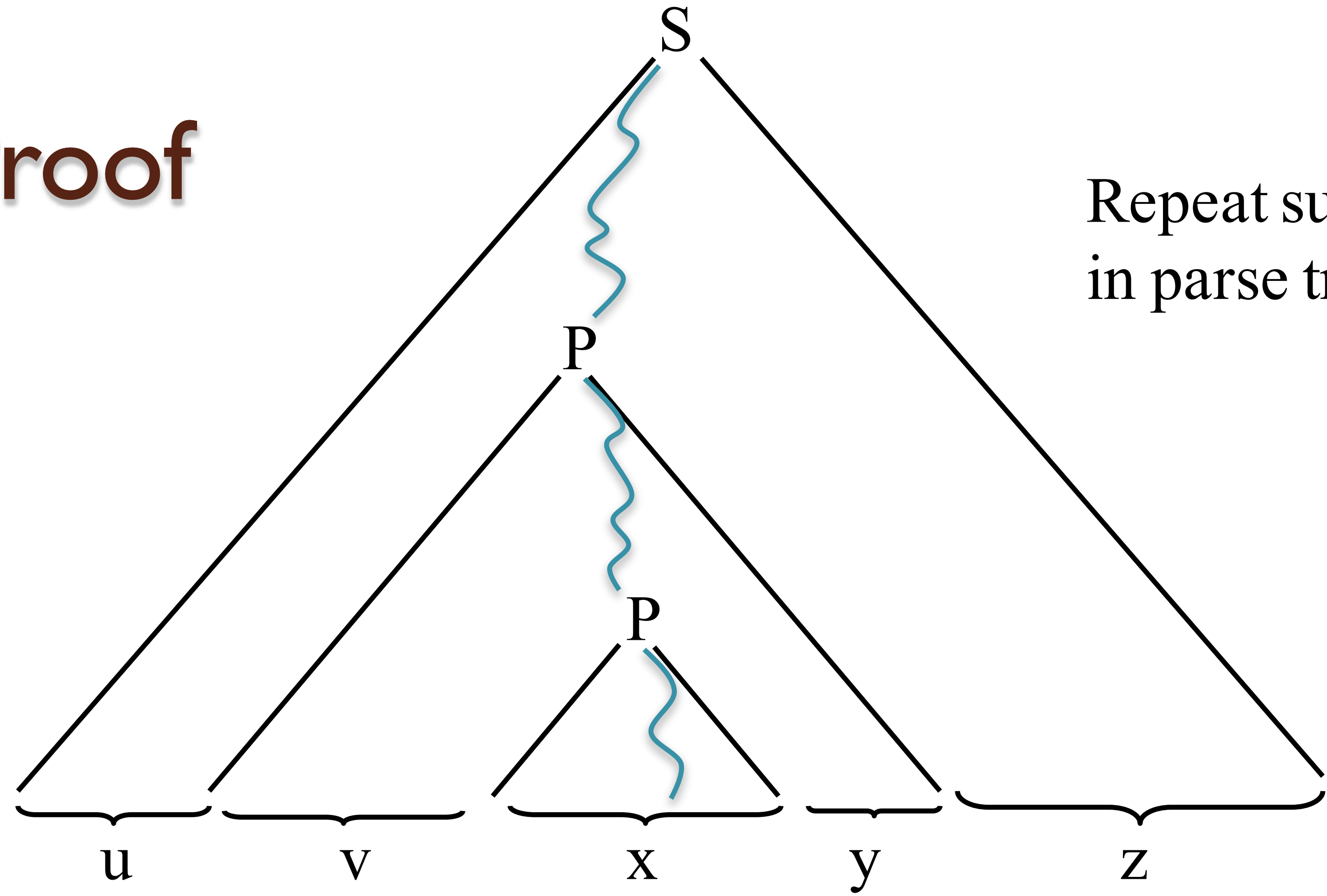
# Proof




Repeat subtree in parse tree.

uvxyz

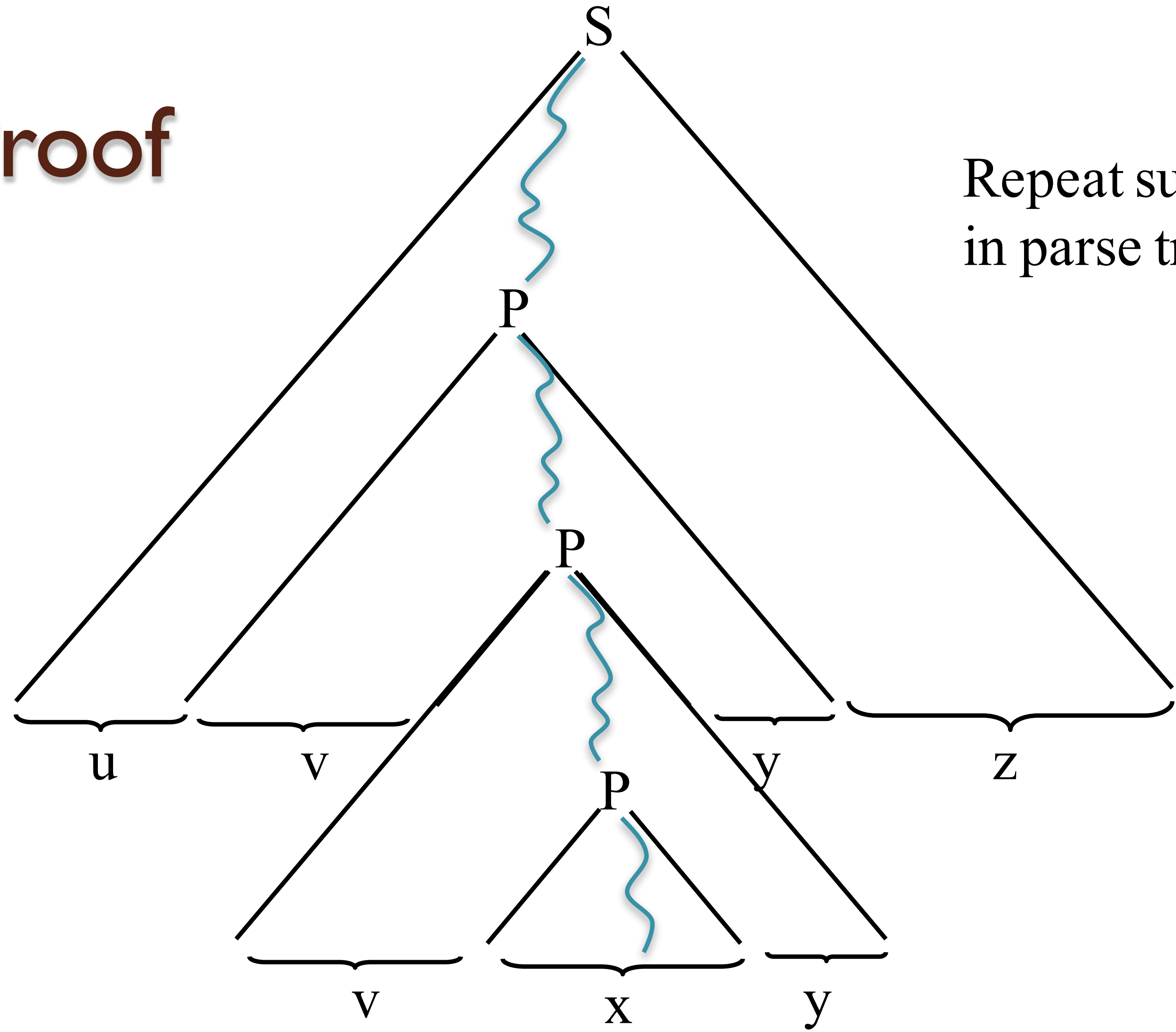
# Proof




Repeat subtree in parse tree.

uvxyz

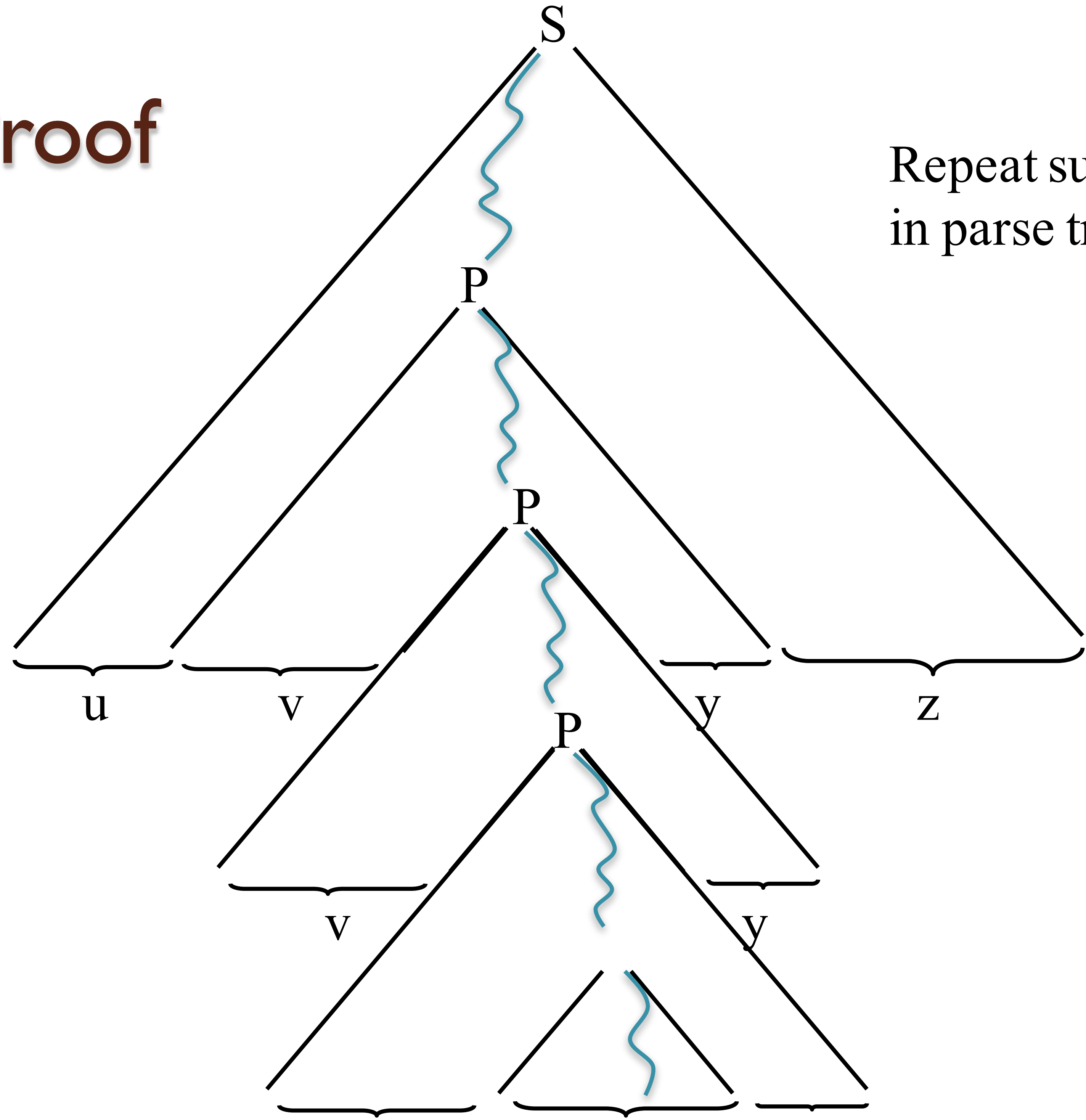
# Proof




Repeat subtree in parse tree.

$uv^2xy^2z$

# Proof




Repeat subtree
in parse tree.

$uv^3xy^3z$

# Proof

So we can generate

$uvxyz$, $uv^2xy^2z$, $uv^3xy^3z$, ……, $uv^nxy^nz$, …

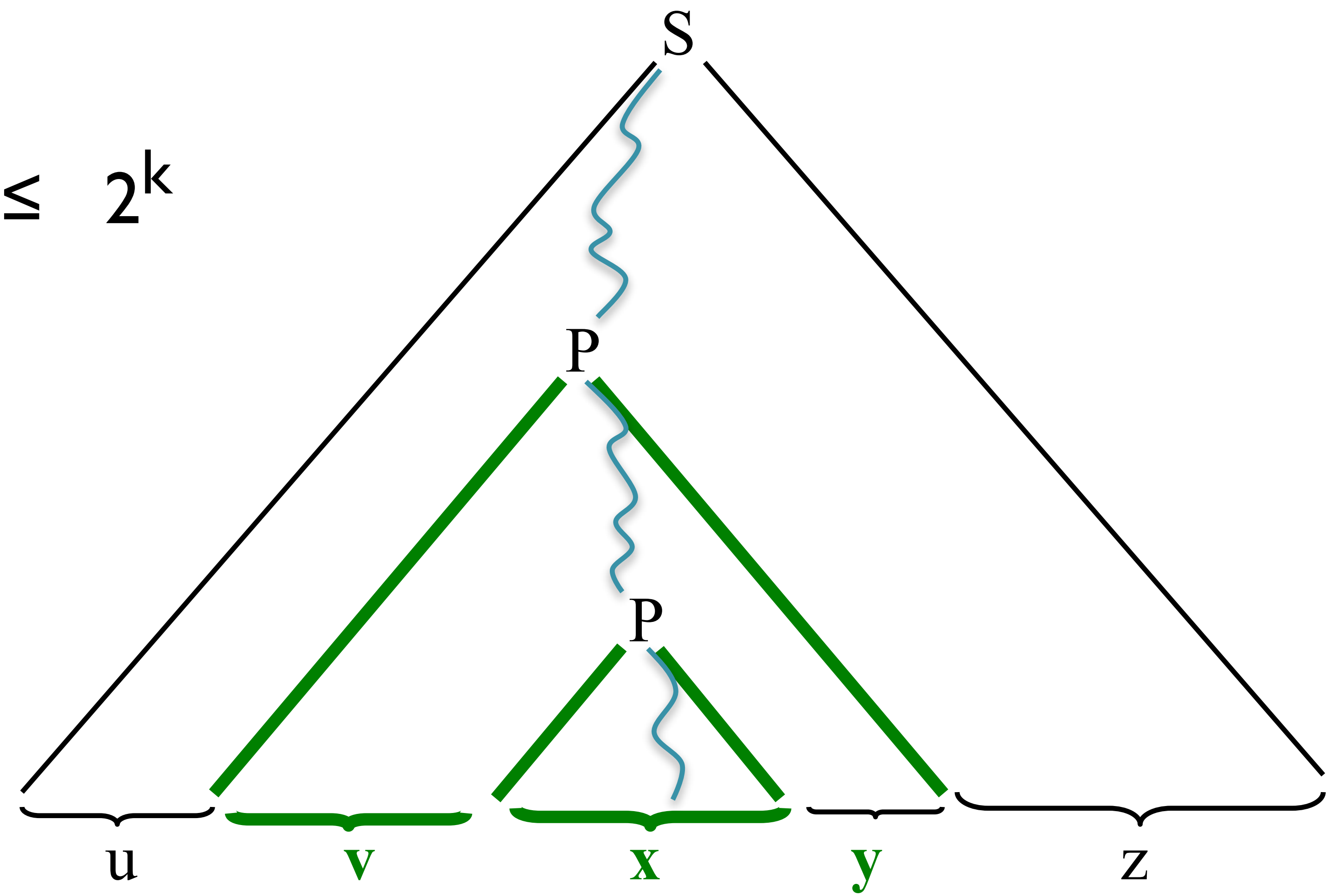
Note also:

We can choose the two repeated non-terminals $P$ to be as far down the path as possible, so there are no other repeated non-terminals in the path from the higher $P$ down to the leaf.

So the path from the upper $P$ downwards has at most $k$ non-terminal nodes apart from the upper P. So the word derived from $P$ by this subtree has at most $2^k$ letters.

# Proof

$|vxy| \leq 2^k$



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

Assume **L** is a context free language.

Then there exists a CFG which generates **L**.

Convert this CFG to CNF.

Suppose it has **k non-terminal** symbols.

Take $\mathbf{N > 2^{k-1}/3}$

Let $\mathbf{w = a^N b^N a^N}$.

Then $\mathbf{length(w) > 2^{k-1}}$

- So, by the Pumping Lemma, there exist strings **u**, **v**, **x**, **y**, and **z**
- such that
  - **w = uvxyz**
  - **v** and **y** are not both **ε**, and
  - $\mathbf{uv^2xy^2z}$, …, $\mathbf{uv^{\mathit{n}}xy^{\mathit{n}}z}$,… are all in L.
- Case 1: **ab** is in **v** or **y**.
- Case 2: **ba** is in **v** or **y**.
- Case 3: **v** and **y** are all **a**'s or all **b**'s or one of them is **ε**.
- Consider: $\mathbf{uv^2xy^2z}$. Contradiction
- Therefore **L** is a non-context free language.

# Revision

- Know the uses of Chomsky Normal Form.
- Know and use the CYK algorithm.
- Know that there exist non-Context Free languages.
- Know an example of a language which is not a Context Free Language.
- Use Pumping Lemma to prove that certain languages are not context-free.
- Reading: Sipser, pp 108-111, 125-129.

## *Preparation*

Read

•M.Sipser, "Introduction to the Theory of Computation", Chapter 3.