Normal Forms
per
grammatiche Context-free

Context free

Ogni produzioni ha la forma:

$$A \to stringa$$

variabile

Terminali o costanti

Chomsky Normal Form

Ogni produzioni ha la forma:

$$A \to BC \qquad \text{or} \qquad A \to a$$

variabile variabile terminal

esempi:

$$S \to AS$$
$$S \to a$$
$$A \to SA$$
$$A \to b$$

Chomsky
Normal Form

$$S \to AS$$
$$S \to AAS$$
$$A \to SA$$
$$A \to aa$$

Not Chomsky
Normal Form

Conversione nella Chomsky Normal Form

esempio:

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

Not Chomsky Normal Form

Convertiamo questa grammatica nella Chomsky Normal Form

Introduciamo nuove variabili per i terminali:

$$T_a, T_b, T_c$$

$$S \to ABa$$
$$A \to aab$$
$$B \to Ac$$

$$S \to ABT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Introduciamo una nuova variabile intermedia

Per rompere la prima produzione: V_1

$$\boxed{S \rightarrow ABT_a}$$

$$A \rightarrow T_a T_a T_b$$

$$B \rightarrow AT_c$$

$$T_a \rightarrow a$$

$$T_b \rightarrow b$$

$$T_c \rightarrow c$$

$$\boxed{\begin{aligned} S &\rightarrow AV_1 \\ V_1 &\rightarrow BT_a \end{aligned}}$$

$$A \rightarrow T_a T_a T_b$$

$$B \rightarrow AT_c$$

$$T_a \rightarrow a$$

$$T_b \rightarrow b$$

$$T_c \rightarrow c$$

Introduciamo la variabile intermedia : V_2

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$\boxed{A \to T_aT_aT_b}$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

$\Rightarrow$

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$\boxed{\begin{array}{l} A \to T_aV_2 \\ V_2 \to T_aT_b \end{array}}$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

grammatica in Chomsky Normal Form:

$$S \to AV_1$$

$$V_1 \to BT_a$$

$$A \to T_a V_2$$

$$V_2 \to T_a T_b$$

$$B \to AT_c$$

$$T_a \to a$$

$$T_b \to b$$

$$T_c \to c$$

Iniziale grammatica

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

In generale:

Per ogni grammatica context-free
(che non produce λ)
non in Chomsky Normal Form

Possiamo ottenere:

una grammatica equivalente
in Chomsky Normal Form

La procedura

First remove:

variabili che si possono annulare

(variabili inutili, optional)

Poi, per ogni simbolo : a

Nuova variabile: T_a

Nuova produzione $T_a \to a$

Nelle produzioni con lunghezza maggiore o uguale a due

poni al posto di a il non terminale T_a

produzioni della forma $A \to a$

Non necessitano di cambio!

Rimpiazza
ogni produzione $\quad A \to C_1 C_2 \cdots C_n$

con
$$A \to C_1 V_1$$
$$V_1 \to C_2 V_2$$
$$\cdots$$
$$V_{n-2} \to C_{n-1} C_n$$

Nuove variabili intermedie : $V_1, V_2, \ldots, V_{n-2}$

Observations

- Chomsky normal forms are good for parsing and proving theorems

- It is easy to find the Chomsky normal form for any context-free grammatica

The Pumping Lemma for CFL's

Statement

Intuition

- Recall the pumping lemma for regular languages.
- It told 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 and discover an infinite sequence of strings that had to be in the language.

Intuition – (2)

- For CFL's the situation is a little more complicated.
- We can always find two pieces of any sufficiently long string to "pump" in tandem.
  - That is: if we repeat each of the two pieces the same number of times, we get another string in the language.

Statement of the CFL Pumping Lemma

For every context-free language L

There is an integer n, such that

For every string z in L of length $\geq$ n

There exists z = uvwxy such that:

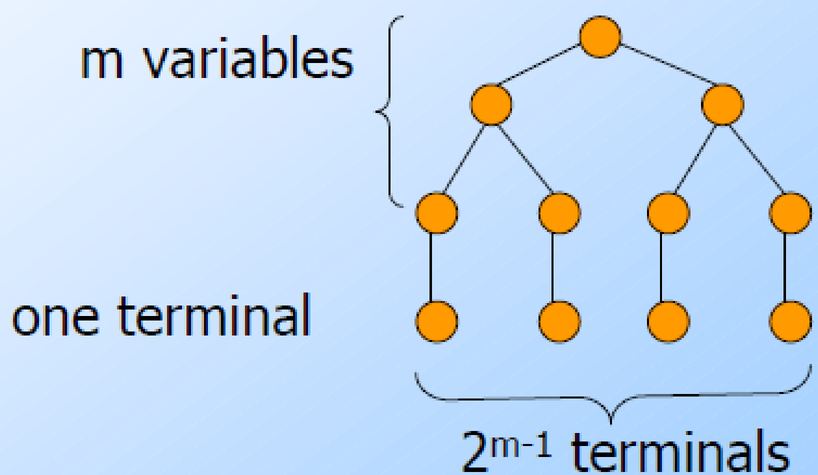
1. |vwx| $\leq$ n.
2. |vx| > 0.
3. For all i $\geq$ 0, uv^iwx^iy is in L.

Proof of the Pumping Lemma

- Start with a CNF grammar for $L - \{\epsilon\}$.
- Let the grammar have m variables.
- Pick $n = 2^m$.
- Let $|z| \geq n$.
- We claim ("*Lemma 1*") that a parse tree with yield z must have a path of length m+2 or more.

Proof of Lemma 1

- If all paths in the parse tree of a CNF grammar are of length $\leq$ m+1, then the longest yield has length 2^{m-1}, as in:

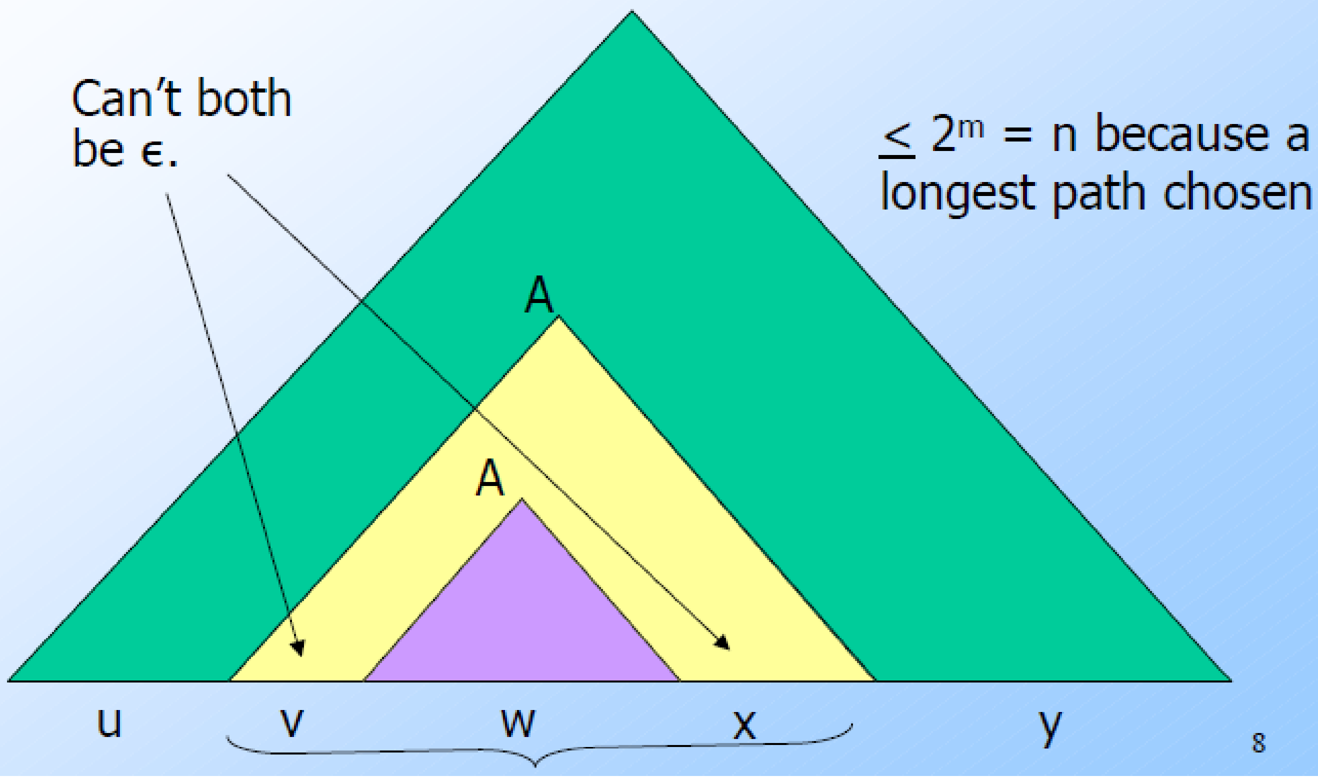


6

Back to the Proof of the Pumping Lemma

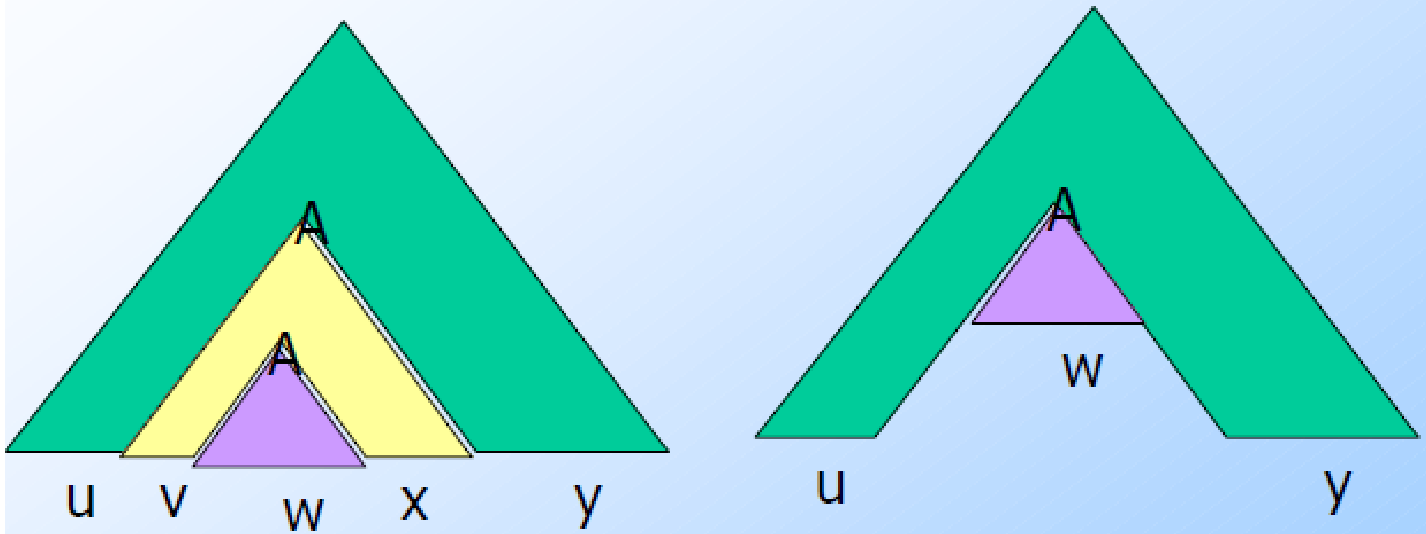
- Now we know that the parse tree for z has a path with at least m+1 variables.
- Consider some longest path.
- There are only m different variables, so among the lowest m+1 we can find two nodes with the same label, say A.
- The parse tree thus looks like:

Parse Tree in the Pumping-Lemma Proof

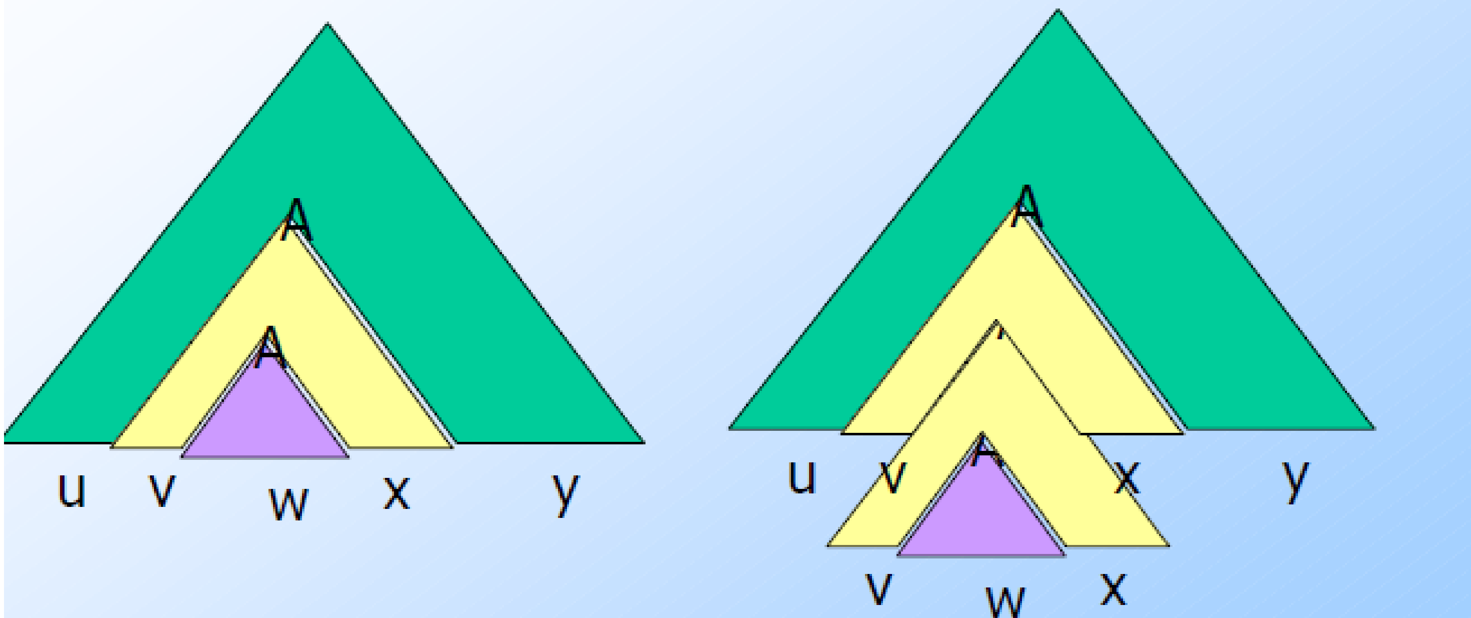


8

Pump Zero Times



Pump Twice



Pump Thrice Etc., Etc.