Monash University
Faculty of Information Technology

FIT2014 Theory of Computation

Lecture 16

Chomsky Normal Form, Cocke-Younger-Kasami algorithm

slides by Graham Farr

Overview

- Chomsky Normal Form
- Nullability
- CYK Parsing algorithm

Chomsky Normal Form

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

$$\text{Nonterminal} \longrightarrow \text{Nonterminal Nonterminal}$$

(called a **live production**)
or

$$\text{Nonterminal} \longrightarrow \text{terminal}$$

(called a **dead production**).

Chomsky Normal Form

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

Proof.
Outline:

1. Eliminate ε-productions. (i.e., production rules of the form $X \longrightarrow \varepsilon$)
2. Eliminate unit productions. (i.e., production rules of the form $X \longrightarrow Y$)
3. Give each terminal its own corresponding nonterminal that produces it.
4. Use these nonterminals to eliminate terminals, except where they appear alone.
5. Break down rules that produce at least three nonterminals, using new nonterminals, to give a set of rules producing just two nonterminals.

1. Eliminate all ε-productions.

For every production rule $X \longrightarrow \varepsilon$:

For every other rule with X in the *body* (right-hand side):

Create new rules with all possible replacements of occurrences of X by ε (and keep the old rule).

Remove the rule $X \longrightarrow \varepsilon$.

For example:

old rules with X in body	new rules
$A \longrightarrow \mathtt{b}XQ$	$A \longrightarrow \mathtt{b}XQ$ and $A \longrightarrow \mathtt{b}Q$
$A \longrightarrow \mathtt{b}XQX$	$A \longrightarrow \mathtt{b}XQX$ and $A \longrightarrow \mathtt{b}QX$ and $A \longrightarrow \mathtt{b}XQ$ and $A \longrightarrow \mathtt{b}Q$
$A \longrightarrow X$	$A \longrightarrow X$ and $A \longrightarrow \varepsilon$

Chomsky Normal Form

Keep doing this until there are no more ε-productions.

Once this step is complete: no rule has an empty right-hand side.

Housekeeping:
Suppose we have a nonterminal that *never* appears on the left of any rule.

(This situation may be created by our elimination of some ε-productions.)

Then we can <u>delete</u> all rules where it appears on the right.

- If a rule has such a nonterminal on the right,
  then that nonterminal can never be replaced,
  so such a rule can never be used in any derivation of a string of terminals.
- This deletion is not strictly necessary for getting a valid CNF grammar.
  But it can yield a simpler result.

Chomsky Normal Form

2. Eliminate all unit productions.

For every production rule $X \longrightarrow Y$:

For every rule with Y on the left:

Create a new rule with X on the left instead of Y (& keep the old rule).

Remove the rule $X \longrightarrow Y$.

For example:

old rules with Y on left	new rules
$Y \longrightarrow \texttt{ab}QR$	$Y \longrightarrow \texttt{ab}QR$ and $X \longrightarrow \texttt{ab}QR$
$Y \longrightarrow Q$	$Y \longrightarrow Q$ and $X \longrightarrow Q$ (unless $X \longrightarrow Q$ has been dealt with previously)
$Y \longrightarrow X$	$Y \longrightarrow X$ and $X \longrightarrow X$

Chomsky Normal Form

2. *(continued)*

Keep doing this until there are no more unit productions.

Once this step is complete: every rule's right-hand side is either

- a single terminal, or
- at least two symbols (terminals and/or nonterminals)

Chomsky Normal Form

3. Give each terminal its own corresponding nonterminal that produces it.

For each terminal z, create a new nonterminal X_z and a new rule $X_z \longrightarrow z$.

For example:
If our terminals are $\mathtt{a}, \mathtt{b}$, then create new nonterminals $X_{\mathtt{a}}, X_{\mathtt{b}}$ and new rules

$$\begin{aligned} X_{\mathtt{a}} &\longrightarrow \mathtt{a} \\ X_{\mathtt{b}} &\longrightarrow \mathtt{b} \end{aligned}$$

Chomsky Normal Form

4. In all rules that don't just produce a single terminal, replace each terminal by its corresponding new nonterminal.

$Y \longrightarrow \mathtt{ab}QR$ becomes $Y \longrightarrow X_{\mathtt{a}}X_{\mathtt{b}}QR$

Once this step is complete: every rule's right-hand side is either

- a single terminal, or
- at least two nonterminals.

Chomsky Normal Form

5. For every rule with more than two nonterminals on the right, create new nonterminals as needed to replace the rule by a set of rules with just two nonterminals on the right.

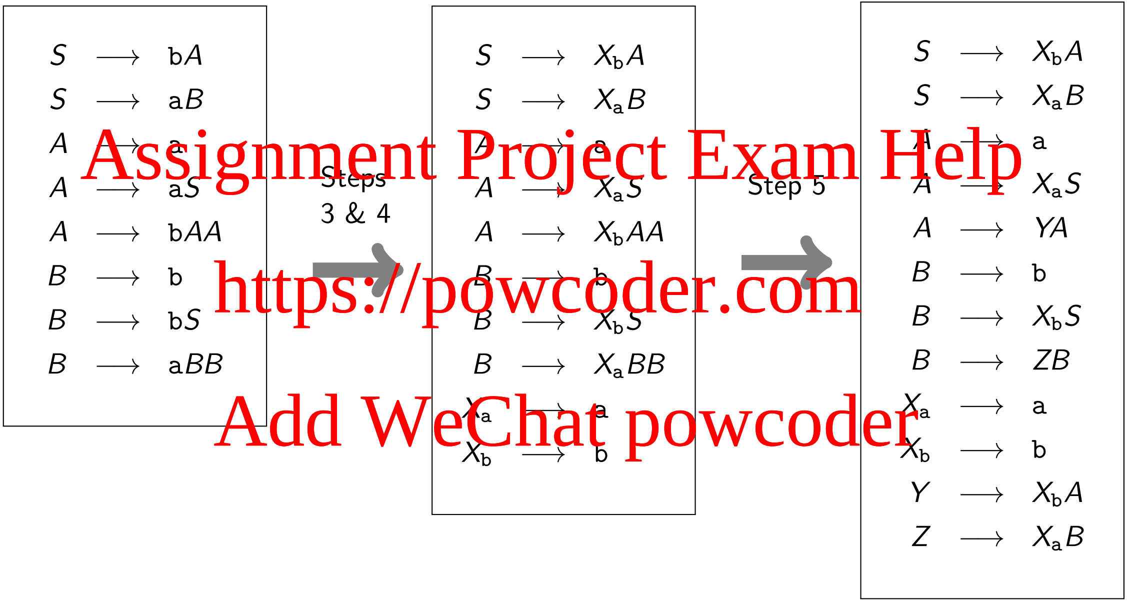
Once this step is complete: every rule's right-hand side is either

- a single terminal, or
- *exactly* two nonterminals.

Example:

old rule	new rules	
$Y \longrightarrow Z_1 Z_2 Z_3$	$Y \longrightarrow Z_{12} Z_3$	and
	$Z_{12} \longrightarrow Z_1 Z_2$	and
$Y \longrightarrow Z_1 Z_2 Z_3 Z_4$	$Y \longrightarrow Z_{12} Z_{34}$	and
	$Z_{12} \longrightarrow Z_1 Z_2$	and
	$Z_{34} \longrightarrow Z_3 Z_4$	
$Y \longrightarrow Z_1 Z_2 Z_3 Z_4 Z_5$	$Y \longrightarrow Z_{1234} Z_5$	and
	$Z_{1234} \longrightarrow Z_{12} Z_{34}$	and
	$Z_{12} \longrightarrow Z_1 Z_2$	and
	$Z_{34} \longrightarrow Z_3 Z_4$	

. . . where Z_{12}, Z_{34}, Z_{1234} are new nonterminals.



$$\begin{array}{lcl} S & \longrightarrow & \mathrm{b}A \\ S & \longrightarrow & \mathrm{a}B \\ A & \longrightarrow & \mathrm{a} \\ A & \longrightarrow & \mathrm{a}S \\ A & \longrightarrow & \mathrm{b}AA \\ B & \longrightarrow & \mathrm{b} \\ B & \longrightarrow & \mathrm{b}S \\ B & \longrightarrow & \mathrm{a}BB \end{array}$$

$$\begin{array}{lcl} S & \longrightarrow & X_{\mathrm{b}}A \\ S & \longrightarrow & X_{\mathrm{a}}B \\ A & \longrightarrow & \mathrm{a} \\ A & \longrightarrow & X_{\mathrm{a}}S \\ A & \longrightarrow & X_{\mathrm{b}}AA \\ B & \longrightarrow & \mathrm{b} \\ B & \longrightarrow & X_{\mathrm{b}}S \\ B & \longrightarrow & X_{\mathrm{a}}BB \\ X_{\mathrm{a}} & \longrightarrow & \mathrm{a} \\ X_{\mathrm{b}} & \longrightarrow & \mathrm{b} \end{array}$$

$$\begin{array}{lcl} S & \longrightarrow & X_{\mathrm{b}}A \\ S & \longrightarrow & X_{\mathrm{a}}B \\ A & \longrightarrow & \mathrm{a} \\ A & \longrightarrow & X_{\mathrm{a}}S \\ A & \longrightarrow & YA \\ B & \longrightarrow & \mathrm{b} \\ B & \longrightarrow & X_{\mathrm{b}}S \\ B & \longrightarrow & ZB \\ X_{\mathrm{a}} & \longrightarrow & \mathrm{a} \\ X_{\mathrm{b}} & \longrightarrow & \mathrm{b} \\ Y & \longrightarrow & X_{\mathrm{b}}A \\ Z & \longrightarrow & X_{\mathrm{a}}B \end{array}$$

Consequences

Cocke-Younger-Kasami (CYK) algorithm (today)

- Given a CFG and a string,
  decides whether or not the string can be generated by the CFG.
- polynomial time
- a bottom-up parsing algorithm.

Pumping Lemma for CFG (next lecture)

- for proving that certain languages are not context-free.

Nullability

Given a CFG, how to decide whether or not it generates the empty string?

A nonterminal A is **nullable** if the empty string can be derived from it:

$$A \Longrightarrow \cdots \Longrightarrow \varepsilon.$$

Algorithm:

1. For every rule of the form $X \longrightarrow \varepsilon$, mark X as nullable.
2. While there is a rule $Y \longrightarrow Y_1 Y_2 \cdots Y_k$ that *only* produces nonterminals *and* all those nonterminals have been marked:
   - Mark Y.
3. If S has been marked, Accept, else Reject.

CYK Algorithm

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

Input: $s = t_1 t_2 \dots t_n$, where each t_i is a letter and $n \geq 0$.

If $s = \varepsilon$ then use the Nullability algorithm.

From now on, s is nonempty.

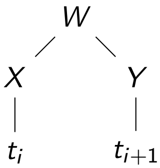
Find the Chomsky Normal Form for the non-empty words generated by the grammar.

For each letter t_k find the nonterminals which can produce t_k.

CYK Algorithm

For each pair of consecutive letters $t_i t_{i+1}$ (where $1 \leq i \leq n-1$), find the nonterminals that can generate the pair, as follows:

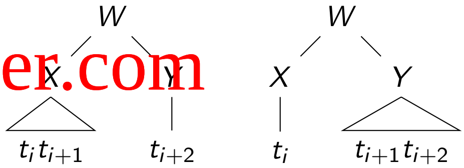
- For each pair X, Y such that
  X generates t_i and Y generates t_{i+1},
  find all W such that there is a rule $W \longrightarrow XY$.



CYK Algorithm

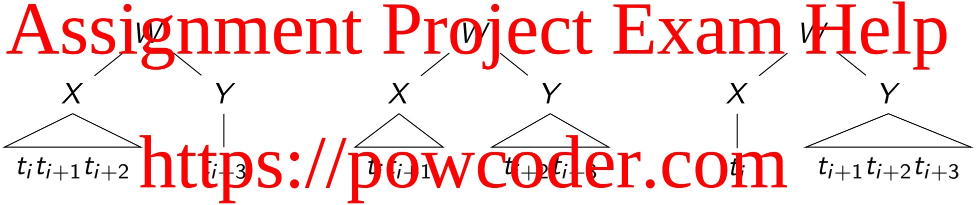
For each triple of consecutive letters $t_i t_{i+1} t_{i+2}$, find the nonterminals that can generate the triple, as follows:

- For each pair X, Y such that $X \overset{*}{\Longrightarrow} t_i t_{i+1}$ and $Y \overset{*}{\Longrightarrow} t_{i+2}$, find all W such that there is a rule $W \longrightarrow XY$.
- For each pair X, Y such that $X \overset{*}{\Longrightarrow} t_i$ and $Y \overset{*}{\Longrightarrow} t_{i+1} t_{i+2}$, find all W such that there is a rule $W \longrightarrow XY$.



CYK Algorithm

For each quadruple of consecutive letters $t_i t_{i+1} t_{i+2} t_{i+3}$,
find the nonterminals that can generate it:



Continue, in this way . . .

Eventually, find all nonterminals that can produce $s = t_1 t_2 \dots t_n$.
If S is one of them,
then Accept, as s can be generated; otherwise, Reject, as s cannot be generated.

CYK Algorithm

$$\begin{array}{lcl} S & \rightarrow & \mathtt{a}S\mathtt{a} \\ S & \rightarrow & \mathtt{b} \end{array} \xrightarrow{\text{CNF}} \begin{array}{lcl} S & \rightarrow & TA \\ S & \rightarrow & \mathtt{b} \\ A & \rightarrow & \mathtt{a} \\ T & \rightarrow & AS \end{array}$$

Input string: aabaa

Single letters

$A \rightarrow \mathtt{a}$

$S \rightarrow \mathtt{b}$

Pairs

$?? \Rightarrow AA \Rightarrow \mathtt{aa}$

$T \Rightarrow AS \Rightarrow \mathtt{ab}$

$?? \Rightarrow SA \Rightarrow \mathtt{ba}$

$?? \Rightarrow AA \Rightarrow \mathtt{aa}$

CYK Algorithm

Triples

$?? \Rightarrow \mathtt{aa\,b}$

$?? \Rightarrow AT \Rightarrow \mathtt{a\,ab}$

$S \Rightarrow TA \Rightarrow \mathtt{ab\,a}$

$?? \Rightarrow \mathtt{a\,ba}$

$?? \Rightarrow \mathtt{ba\,a}$

$?? \Rightarrow \mathtt{b\,aa}$

4-tuples

$?? \Rightarrow \mathtt{aab\,a}$

$?? \Rightarrow \mathtt{aa\,ba}$

$T \Rightarrow AS \Rightarrow \mathtt{a\,aba}$

$?? \Rightarrow SA \Rightarrow \mathtt{aba\,a}$

$?? \Rightarrow \mathtt{ab\,aa}$

$?? \Rightarrow \mathtt{a\,baa}$

5-tuples

$S \Rightarrow TA \Rightarrow \mathtt{aaba\,a}$

$?? \Rightarrow \mathtt{aab\,aa}$

$?? \Rightarrow \mathtt{aa\,baa}$

$?? \Rightarrow \mathtt{a\,abaa}$

So S can generate aabaa, and we are done.

CYK Algorithm

Exercises:

Write the algorithm more formally.

Prove by induction that the algorithm works.

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

Revision

- Know the uses of Chomsky Normal Form, and be able to convert a grammar to it.
- Avoid confusion between
  Chomsky Normal Form (CNF) and Conjunctive Normal Form (CNF)!
- Know and use the CYK algorithm.

Reading: Sipser, pp. 108–111.