# Lecture 6 – Chomsky Normal Form, Pumping lemma for context-free languages

NTIN071 Automata and Grammars

Jakub Bulín (KTIML MFF UK)

Spring 2024

** Adapted from the Czech-lecture slides by Marta Vomlelová with gratitude. The translation, some modifications, and all errors are mine.*

# Recap of Lecture 5

- Grammars: general, context-sensitive, context-free, right-linear (regular) – Chomsky hierarchy
- The language of a grammar, derivation
- Right-linear grammars correspond to FA (and so do left/linear)
- Linear grammars are stronger
- Context-free grammars: parse tree and its yield
- (un)ambiguous grammars, inherently ambiguous languages

# 2.6 Chomsky Normal Form

# Chomsky normal form

The Chomsky normal form (ChNF) of a context-free grammar:

- all rules of the form $A \to BC$ or $A \to a$ ($A, B, C \in V$, $a \in T$)
- no useless symbols

**Theorem**

*For every context-free language $L$ such that $L \setminus \{\epsilon\} \neq \emptyset$ there exists a grammar in ChNF that generates $L \setminus \{\epsilon\}$.*

Applications:

- Test membership in $L$: the CYK algorithm (Sakai 1962)
- Prove the Pumping lemma for context-free languages

# Converting to ChNF

Take any context-free grammar for $L$ and simplify (in this order!):

1. eliminate $\epsilon$-productions $A \to \epsilon$ [here we lose $\epsilon \in L$]
2. eliminate unit productions $A \to B$
3. eliminate useless symbols
   - 3a. unreachable [from the start symbol]
   - 3b. nongenerating [a word over terminals]

Now we have a reduced grammar. To get to ChNF, we further:

4. separate terminals from bodies
5. break up longer bodies

# Step 1: Eliminate $\epsilon$-productions

A variable $A \in V$ is nullable if $A \Rightarrow^* \epsilon$. An algorithm to find them:

**basis:** for every $\epsilon$-production $A \to \epsilon$ mark $A$ as nullable

**induct:** if $B \to C_1 \dots C_k \in \mathcal{P}$ where all $C_i$ are nullable, $B$ is nullable

**To eliminate $\epsilon$-productions:** 1. find nullable variables, 2. remove $\epsilon$-productions, 3. process every production $A \to X_1 \dots X_k \in \mathcal{P}$:

- let $J \subseteq \{1, \dots, k\}$ be the positions of all nullable variables
- for every $J' \subseteq J$ create a copy of the production where $X_j$ for $j \in J'$ are deleted, except if $J = \{1, \dots, k\}$ require $J' \neq \emptyset$

**Example:** $\mathcal{P} = \{S \to AB, A \to aAB \mid \epsilon, B \to ABBA \mid \epsilon\}$

$S \to AB \mid A \mid B$ $A \to aAB \mid aA \mid aB \mid a$
$B \to ABBA \mid ABA \mid ABB \mid BBA \mid AA \mid AB \mid BA \mid BB \mid A \mid B$

# Step 2: Eliminate unit productions

**Idea**: for a unit production $A \to B$ copy rules for $B$ with head $A$, but unit productions can be composed, we need transitive closure:

Unit pairs $\mathcal{U} \subseteq V \times V$ are defined as follows:

- $(A, B) \in \mathcal{U}$ for every unit production $A \to B \in \mathcal{P}$
- if $(A, B) \in \mathcal{U}$ and $(B, C) \in \mathcal{U}$, then $(A, C) \in \mathcal{U}$

**To eliminate unit productions:**

1. find all unit pairs $\mathcal{U}$
2. remove all unit productions
3. for every unit pair $(A, B) \in \mathcal{U}$ and production $B \to \beta \in \mathcal{P}$ add the production $A \to \beta$ to $\mathcal{P}$

# Step 2: Eliminate unit productions – an example

$E \to T \mid E + T$
$F \to I \mid (E)$
$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
$T \to F \mid T * F$

unit pairs:
$(E, E), (E, F), (E, I), (E, T),$
$(F, F), (F, I),$
$(I, I),$
$(T, F), (T, I), (T, T)$

the result:
$E \to E + T \mid T * F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
$F \to (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
$T \to T * F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

## Step 3: Eliminate useless symbols

- $X \in V \cup T$ is a useful symbol (in $G$) if there exists a derivation of the form $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$ for some $w \in T^*$
- $X$ is useless if it is not useful
- $X$ is generating if $X \Rightarrow^* w$ for some $w \in T^*$
- $X$ is reachable if $S \Rightarrow^* \alpha X \beta$ for some $\alpha, \beta \in (V \cup T)^*$

Observe:

- useful $\Leftrightarrow$ generating and reachable
- useless $\Leftrightarrow$ nongenerating or unreachable (we eliminate both)
- all terminals are generating

## Step 3: Eliminate useless symbols – the algorithm

1. Find all generating symbols:

   **basis:** mark all terminals $a \in T$ as generating

   **induct:** for every production $A \to \beta$ where every symbol in the body $\beta$ is generating, mark the head $A$ as generating (incl. $A \to \epsilon$)

2. Remove all nongenerating symbols and rules containing them
3. Find all reachable symbols

   **basis:** mark $S$ as reachable

   **induct:** for every production $A \to \beta$ where the head $A$ is reachable mark every symbol in the body $\beta$ as reachable

4. Remove all unreachable symbols and rules containing them

- The order is important! Eliminating nongenerating symbols can create new unreachable symbols, but not vice versa.
- **Example:** eliminate nongenerating $B$, then unreachable $A$

$$S \to AB \mid a \qquad S \to a \qquad S \to a$$
$$A \to b \qquad A \to b$$

## Steps 4 & 5: Separate terminals and break up long bodies

**Step 4: Separate terminals from bodies**

For every $a \in T$, introduce a new variable $V_a$ and the rule $V_a \to a$.

For every rule $A \to \beta$ with $|\beta| \geq 2$, replace every terminal $a$ by $V_a$.

**Step 5: Break up longer bodies**

Replace every rule $A \to B_1 \dots B_k$ with $k \geq 3$ with:

$$\begin{aligned} A &\to B_1 C_1 \\ C_1 &\to B_2 C_2 \\ &\vdots \\ C_{k-2} &\to B_{k-1} B_k \end{aligned}$$

where $C_1, \dots, C_{k-2}$ are new variables (only used for this purpose).
[Alternatively, instead of a chain use a binary tree.]

# Conversion to Chomsky Normal Form

ChNF: only useful symbols and rules $A \to BC$ or $A \to a$

**Theorem**

*For every context-free language $L$ such that $L \setminus \{\epsilon\} \neq \emptyset$ there exists a grammar in ChNF that generates $L \setminus \{\epsilon\}$.*

**Proof.**

Take a context-free grammar $G$ for $L$. Modify it by applying steps 1, 2, 3a, 3b, 4, and 5, in order. Clearly, the result is in ChNF. After step 1 we get $G'$ such that $L(G') = L(G) \setminus \{\epsilon\}$; the remaining steps produce equivalent grammars. Steps 2-5 don't add any $\epsilon$-productions, 3-5 don't add unit productions, 3b-5 don't add nongenerating symbols, 4-5 don't add useless, etc. $\square$

Note: If we only apply 1, 2, 3a, and 3b, we get a reduced grammar: only useful symbols, no $\epsilon$-productions, no unit productions.

# Example

$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$F \to I \mid (E)$

$T \to F \mid T * F$

$E \to T \mid E + T$

---

**reduce + separate**

$I \to a \mid b \mid IA \mid IB \mid IZ \mid IU$

$F \to LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IU$

$T \to TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IU$

$E \to EPT \mid TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IU$

$A \to a,\ B \to b,\ Z \to 0,\ U \to 1,$
$P \to +,\ M \to *,\ L \to (,\ R \to )$

**break up longer bodies**

$F \to LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IU$

$T \to TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IU$

$E \to EC_1 \mid TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IU$

$C_1 \to PT$

$C_2 \to MF$

$C_3 \to ER$

$I, A, B, Z, U, P, M, L, R$ same as on the left side

## 2.7 Pumping lemma for context-free languages

# Pumping lemma for context-free languages

**Theorem (Pumping Lemma for Context Free Languages)**

*Let $L \subseteq \Sigma^*$ be context-free. Then there exists $n \in \mathbb{N}$ s.t. for any $z \in L, |z| \geq n$ there are $u, v, w, x, y \in \Sigma^*$ s.t. $z = uvwxy$ and:*

*(i) $|vwx| \leq n$ (ii) $|vx| > 0$ (iii) $uv^iwx^iy \in L$ for all $i \geq 0$*

**Proof idea:** Take a ChNF grammar for $L$. If $z \in L$ is long enough, a parse tree for $z$ must contain a path from $S$ to a leaf (terminal) of length $|V| + 1$. Some nonterminal $N \in V$ repeats on this path giving two subtrees with root $N$: a larger one containing a smaller one. Replace the larger with a copy of the smaller ($i = 0$) or the smaller with a copy of the larger ($i = 2$).

**What is long enough?** If $|z| > 2^{k-1}$, then the depth of the tree is $k + 1$. (All inner nodes not immediately above a leaf are binary!)

# The proof in a picture

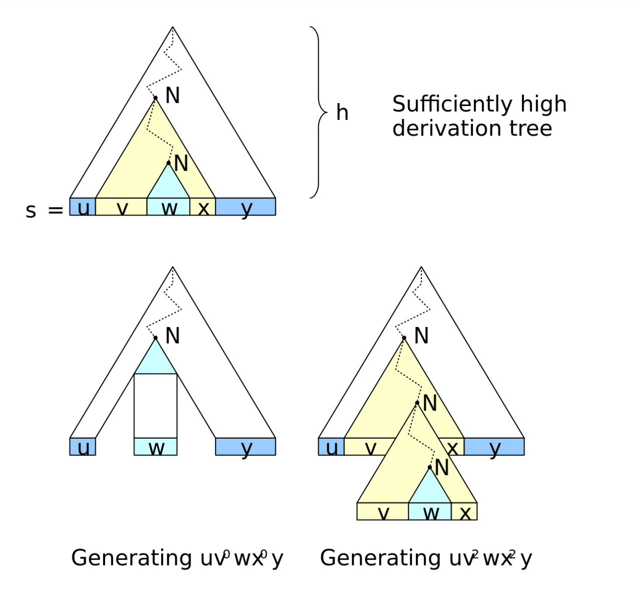


Image by Jochen Burghardt, CC BY-SA 3.0, from Wikipedia

# The proof

If $L = \emptyset$ and $L = \{\epsilon\}$ trivial, take $n = 1$. Otherwise take a ChNF grammar for $L$. Set $n = 2^{|V|-1} + 1$ . Let $z \in L$ with $|z| \geq n$.

A parse tree for $z$ contains a path from $S$ to a terminal $t$ of length at least $|V| + 1$. At least two of the last $|V| + 1$ nonterminals on this path must be the same. Let $A^1, A^2$ be such a pair that is closest to $t$. Let $T^1$, $T^2$ be the subtrees rooted at $A^1, A^2$.

The path from $A^1$ to $t$ is the longest one in $T^1$ and has length at most $(k + 1)$. Thus $|vwx| \leq n$.

There are two paths from $A^1$ (ChNF!): one leads to $T^2$, the other to the rest, it must generate at least one letter (no $\epsilon$-productions). Thus $|vx| > 0$.

# The proof cont'd

The word $z = uvwxy$ is derived as follows:

- $A^2 \Rightarrow^* w$
- $A^1 \Rightarrow^* vA^2x \Rightarrow^* vwx$
- $S \Rightarrow^* uA^1y \Rightarrow^* uvA^2xy \Rightarrow^* uvwxy$

For $i = 0$: replace $T^1$ by $T^2$

$$S \Rightarrow^* uA^2y \Rightarrow^* uwy$$

For $i = 2$: replace $T^2$ by a copy of $T^1$

$$S \Rightarrow^* uA^1y \Rightarrow^* uvA^1xy \Rightarrow^* uvvA^2xxy \Rightarrow^* uvvwxxy$$

For $i \geq 3$ repeat the above. $\square$

# Application: proving a laguage is not context-free

**Example**

The language $L = \{0^n 1^n 2^n \mid n \geq 0\}$ is not context-free.

Suppose for contradiction that it is. Let $n$ be constant from the Pumping lemma. Choose $z = 0^n 1^n 2^n \in L$. Clearly $|z| \geq n$.

The Pumping lemma gives us a split $z = uvwxy$ satisfying (i)–(iii). Since $|vwx| \leq n$, the pumped part $vx$ contains at most two of the symbols $0, 1, 2$. Pumping will violate equal number of symbols. $\square$

**Example**

The language $L = \{0^i 1^j 2^k \mid 0 \leq i \leq j \leq k\}$ is not context-free.

Similar as above, also $z = 0^n 1^n 2^n$, at most two symbols pumped:

- if 0 or 1 are pumped, but 2 is not: pump up $(i = 2)$
- if 1 or 2 are pumped, but 0 is not: pump down $(i = 0)$ $\square$

# More examples

**Example**

$L = \{0^j 1^k 2^j 3^k \mid j, k \geq 0\}$ is not context-free.

Similar as before, choose $z = 0^n 1^n 2^n 3^n$, $vx$ must contain some symbol. But from $|vwx| \leq n$ we know that it can contain neither both 0 and 2, nor both 1 and 3. In any case, the equal number of symbols 0 and 2 or 1 and 3 is violated. $\square$

**Example**

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

Choose $z = 0^n 1^n 0^n 1^n$, then $|z| \geq n$. The pumped part can cover neither both blocks of 0s nor both blocks of 1s. Four cases to consider: $vx$ contains a symbol from the 1st block of 0s, 1st block of 1s, 2nd 0s, 2nd 1s. In all cases we get a violation. $\square$

# It is not a characterization

The Pumping lemma is again only an implication, not equivalence:

**Example**

$L = \{a^i b^j c^k d^\ell | i = 0 \text{ or } j = k = \ell\}$ can be pumped. But it is not context-free.

| | |
|---|---|
| $i = 0 : b^j c^k d^l$ | can be pumped in any letter |
| $i > 0 : a^i b^n c^n d^n$ | can be pumped in $a^*$ |

What to do in such cases?

- Ogden's lemma: generalize Pumping lemma, mark some of the letters, some marked symbol is pumped
- use closure properties of context-free languages

# Summary of Lecture 6

- Reducing a grammar: removing $\epsilon$-productions, unit productions, useless symbols
- Chomsky Normal Form of a context-free grammar
- Pumping lemma for context-free languages, application: proving non-context-freeness