Theory of Computation
LEIC 010
2023-2024

Jácome Cunha & Luís Antunes

DEI@FEUP – DCC@FCUP
Universidade do Porto

Class 9
Proprieties of Context-Free Languages and More
(16/4/24)

Covered Concepts

1. Pumping Lemma for CFLs
2. Chomsky Normal Form for CFGs

Outline

Pumping Lemma for CFLs

Let's assume that L is a CFL:

- there exists a constant n such that,
- for every z in L with $|z| \geq n$
- we can write $z = uvwxy$ in which
- $|vwx| \leq n$ (the middle part is not too long)
- $vx \neq \epsilon$ (v and x are not simultaneously the empty word)
- for every $i \geq 0$, $uv^iwx^iy \in L$.

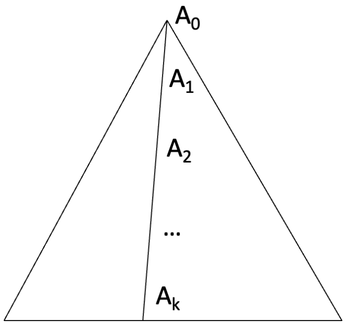
Insights of the Pumping Lemma for CFLs

- Let's focus on the length of the string z, and on terminals alone (i.e., only in productions like $A \to a$).
- With n variables and without repetition:
  - The longest string z considering n variables occurs with $V_1 \to V_2 \ldots V_n$, and $|z| \leq n - 1$.
  - For $|z| \geq n$, variables need to be repeated.
- If the string z is sufficiently long, there must be a repetition of symbols (variables).

Insights of the Pumping Lemma for CFLs (cont.)

Given a grammar containing n variables and a string z in L with $|z| \geq n$:

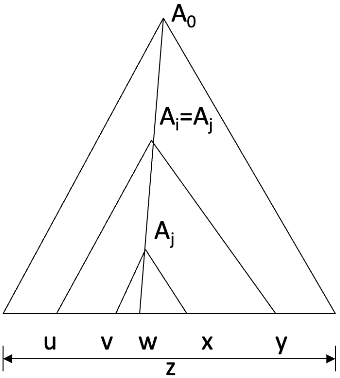
- The tree path $A_0 \dots A_k$ (length $k+1$) is the longest path for z.
- Since $k \geq n$ (as per Theorem 7.17 in Hopcroft's book), there are at least $n+1$ occurrences of $A_0 \dots A_k$.
- Given there are only n variables, at least one variable is repeated.
- To have a yield of length $\geq n$, there must be repetitions of variables in the tree (this implies recursivity).



Insights of the Pumping Lemma for CFLs (cont.)

Let's split the parse tree:

- Suppose the repeated variable in the tree is $A_i = A_j$.
- w is the string at the leaves of the subtree rooted at A_j.
- v and x are such that vwx is the string represented by the subtree rooted at A_i (at least one of v or x is not null).
- u and y are the parts of z to the left and right of vwx, respectively.

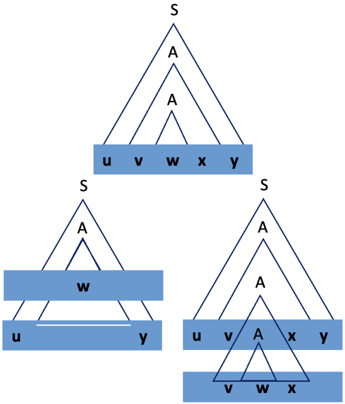


Insights of the Pumping Lemma for CFLs (cont.)

As $A_i = A_j$, we can:

- Substitute the subtree rooted at A_i with the subtree rooted at A_j, obtaining the case $i = 0$, uwy.
- Substitute the subtree rooted at A_j with the subtree rooted at A_i, obtaining the case $i = 2$, uv^2wx^2y, and repeat for $i = 3, \ldots$ (pumping).

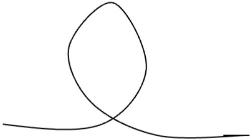
With repetitions of at least one variable A, one can have $S \rightarrow uAy$ and $A \rightarrow w \mid vAx$ where $|vx| \neq \epsilon$.



Insights of the Pumping Lemma for CFLs (cont.)

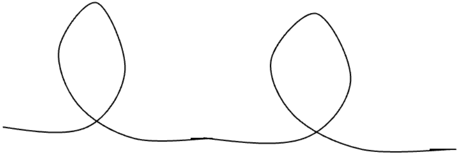
- Previously, we considered that the terminals were alone in each variable (productions $A \rightarrow a$).
- The general case, where terminals may exist in any variable and in a finite number,
  - only means that the length of string z is proportional to the number of variables (i.e., it might be the number of variables, n, plus the length of the yield of the terminals in the variables)
  - similar when we repeat variables in productions but without recursivity.

Pumping Lemmas



In the case of Regular Languages (RLs):

- The pumping lemma results from the fact that the number of states in a Deterministic Finite Automaton (DFA) is finite.
- To accept a sufficiently long string, the processing within the DFA needs to repeat states.



In the case of Context-Free Languages (CFLs):

- The pumping lemma results from the fact that the number of symbols in a Context-Free Grammar (CFG) is finite.
- To accept a sufficiently long string, the derivations within the CFG must repeat variables.

Example

Let's use the lemma to show that the language $L = \{0^k 1^k 2^k \mid k \geq 1\}$ is not a CFL.

- Assume that L is a CFL.
- Then, there exists a constant n as described in the lemma.
- Let's choose $z = 0^n 1^n 2^n$, which belongs to L and $|z| = 3n \geq n$.
- We can then decompose $z = uvwxy$, such that $|vwx| \leq n$ and v and x are not simultaneously the empty word.
- Then, vwx cannot simultaneously contain 0 and 2.
    - In the case that vxw does not contain 2, then vx includes only 0 and/or 1.
      Choosing $i = 0$, the word $uv^iwx^iy = uwy$ will have fewer 0 or 1 than 2.
      Therefore, uv^iwx^iy, $i = 0$, does not belong to L, thus L does not comply with the lemma's statement.
      Finding this contradiction, we can only conclude that our assumption that L is a CFL is wrong. Consequently, L is not a CFL.
    - In the case that vxw does not contain 0, the argument is analogous.

Exercise

Exercise

Consider $L = \{0^i 1^j 2^i 3^j \mid i \geq 1, j \geq 1\}$. *Show L is not a CFL.*

Exercise

Exercise

Consider $L = \{0^i 1^j 2^i 3^j \mid i \geq 1, j \geq 1\}$. *Show* L *is not a CFL.*

- If L is a CFL, then there exists a constant $n > 0$ such that for all z in L with $|z| \geq n$, we can write z as $uvwxy$ where $|vwx| \leq n$, v and x are not both the empty string, and $uv^i wx^i y$ belongs to L for all $i \geq 0$.
- Let's pick $w = 0^n 1^n 2^n 3^n$. Then $|w| = 4n \geq n$.
- Since $|vwx| \leq n$, vx can contain at most two different symbols at a time (either 0 and 1, or 1 and 2, or 2 and 3).
- If vx contains two different symbols, by choosing $i = 0$, we are removing a repetition of such symbols, and thus they will have one fewer repetition than the others.
- If vx contains only one symbol, choosing $i = 0$ will remove one repetition of such symbol, making it have fewer repetitions than the others.
- Thus, L cannot be regular.

Outline

Chomsky Normal Form for CFGs

- Any CFL that does not include the empty word can be generated by a CFG $G = (V, T, P, S)$ where all productions are of the form $A \to BC$ or $A \to a$, with $A, B, C \in V$ and $a \in T$.
- This grammar is said to be in *Chomsky Normal Form* (CNF).
- Starting from any CFG, we can apply various simplifications until we reach CNF:
  1. It is necessary to eliminate **useless symbols**, i.e., those variables or terminals that do not appear in any derivation of a terminal string from the start symbol;
  2. It is necessary to eliminate ϵ **productions**, i.e., those of the form $A \to \epsilon$, for some variable A;
  3. It is necessary to eliminate **unit productions**, i.e., those of the form $A \to B$ for variables A and B.

Useless Symbols

- X is **useful** for a grammar $G = (V, T, P, S)$ if there exists some derivation $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$, with $w \in T^*$, $X \in V \cup T$.
- If X is not useful, then it is **useless**.
- Omitting useless symbols from a grammar does not change the generated language.
- Therefore, we can detect and eliminate all useless symbols.

1. Elimination of Useless Symbols

- To eliminate useless symbols, it's necessary to identify the two capabilities a symbol needs to be considered useful:
  1. X is **generating** if $X \Rightarrow^* w$ for some word w.
     1. All terminals are generating since w can be generated by themselves, which is a derivation in zero steps.
     2. If $A \rightarrow \alpha$ and α consists only of generating symbols, then A is generating (including $\alpha = \epsilon$).
  2. X is **reachable** if there is a derivation $S \Rightarrow^* \alpha X \beta$ for some α and β.
     1. S is reachable.
     2. If A is reachable, then for $A \rightarrow \alpha$, all symbols in α are reachable.
- A symbol that is both generating and reachable is useful.
- To eliminate useless symbols, we first remove non-generating symbols, then eliminate non-reachable ones.

Example

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

- a and b generate themselves
- S generates a
- A generates b
- B is not generating, so the production $S \to AB$ can be eliminated.

$$S \to a$$
$$A \to b$$

- S reachable
- a reachable
- But A and b are not, so the production $A \to b$ can be eliminated.
- The final grammar without useless symbols is

$$S \to a$$

Exercise

Eliminate the useless symbols from the following CFG:

$$
\begin{aligned}
S &\rightarrow AB \mid CA \\
A &\rightarrow a \\
B &\rightarrow BC \mid AB \\
C &\rightarrow aB \mid b
\end{aligned}
$$

Exercise

Eliminate the useless symbols from the following CFG:

$$
\begin{aligned}
S &\to AB \mid CA \\
A &\to a \\
B &\to BC \mid AB \\
C &\to aB \mid b
\end{aligned}
$$

$$
\begin{aligned}
S &\to CA \\
A &\to a \\
C &\to b
\end{aligned}
$$

ϵ-Productions

- A variable A is nullable if $A \Rightarrow^* \epsilon$.
- If A is nullable and appears in the body of a production, say $B \to CAD$, we create two productions:
  1. $B \to CAD$
  2. $B \to CD$
- Subsequently, we remove all productions that derive ϵ.
- Algorithm:
  - If $A \to \epsilon$, then A is nullable.
  - If $B \to C_1 C_2 \ldots C_k$ and all C_i are nullable, then B is nullable.

2. Elimination of ϵ-Productions

- To eliminate ϵ productions:
  - Identify all nullable productions.
  - For each $A \to X_1 X_2 \ldots X_k$, if m X_i are nullable, replace the production with 2^m productions including all combinations of presence and absence of the X_i.
    - Exception: if $m = k$, do not include cases where all X_i are absent.
  - Productions in the form $A \to \epsilon$ are removed.

Example

Consider the CFG

$$S \to AB$$
$$A \to aAA \mid \epsilon$$
$$B \to bBB \mid \epsilon$$

A and B are nullable, so S is also nullable, resulting in:

$$S \to AB \mid A \mid B$$
$$A \to aAA \mid aA \mid aA \mid a$$
$$B \to bBB \mid bB \mid b$$

CFG without ϵ productions:

$$S \to AB \mid A \mid B$$
$$A \to aAA \mid aA \mid a$$
$$B \to bBB \mid bB \mid b$$

In this case, the language of the new CFG is the language of the original CFG minus ϵ.

Exercise

Consider the CFG

$$
\begin{aligned}
S &\to ASB \mid \epsilon \\
A &\to aAS \mid a \\
B &\to SbS \mid A \mid bb
\end{aligned}
$$

Find an equivalent grammar without ϵ productions.

Exercise

Consider the CFG

$$
\begin{aligned}
S &\to ASB \mid \epsilon \\
A &\to aAS \mid a \\
B &\to SbS \mid A \mid bb
\end{aligned}
$$

Find an equivalent grammar without ϵ productions.

$$
\begin{aligned}
S &\to ASB \mid AB \\
A &\to aAS \mid aA \mid a \\
B &\to SbS \mid bS \mid Sb \mid b \mid A \mid bb
\end{aligned}
$$

Unit Productions

- Unit production: $A \to B$, where A and B are variables.
- They can be useful in eliminating ambiguity (example: language of arithmetic expressions).
- However, they are not essential, as they introduce extra steps in derivations.

3. Elimination of Unit Productions

1. Determine all unit pairs, derived only with unit productions:
   1. (A, A) is a unit pair.
   2. If (A, B) is a unit pair and $B \rightarrow C$, where C is a variable, then (A, C) becomes a unit pair.
2. Elimination: replace existing productions so that each unit pair (A, B) includes all productions of the form $A \rightarrow \alpha$, where $B \rightarrow \alpha$ is a non-unit production (includes $A = B$).

Example

Consider the CFG

$$
\begin{aligned}
E &\to T \mid E + T \\
T &\to F \mid T \times F \\
F &\to I \mid (E) \\
I &\to a \mid b \mid Ia \mid Ib \mid I0 \mid I1
\end{aligned}
$$

The unit pairs are as follows:

- Base unit pairs:
  - (E, E), (T, T), (F, F), (I, I)
- Inductive unit pairs:
  - (E, E) and $E \to T \leadsto (E, T)$
  - (E, T) and $T \to F \leadsto (E, F)$
  - (E, F) and $F \to I \leadsto (E, I)$
  - (T, T) and $T \to F \leadsto (T, F)$
  - (T, F) and $F \to I \leadsto (T, I)$
  - (F, F) and $F \to I \leadsto (F, I)$

cont.

From the unit pairs on the last slide, the following productions are formed:

Pair	Productions
(E, E)	$E \rightarrow E + T$
(E, T)	$E \rightarrow T \times F$
(E, F)	$E \rightarrow (E)$
(E, I)	$E \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(T, T)	$T \rightarrow T \times F$
(T, F)	$T \rightarrow (E)$
(T, I)	$T \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(F, F)	$F \rightarrow (E)$
(F, I)	$F \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(I, I)	$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

cont.

The final CFG is:

$$
\begin{aligned}
E &\rightarrow E + T \mid T \times F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
T &\rightarrow T \times F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F &\rightarrow (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
I &\rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1
\end{aligned}
$$

Exercise

Consider the CFG

$$
\begin{aligned}
S &\to ASB \mid AB \\
A &\to aAS \mid aA \mid a \\
B &\to SbS \mid bS \mid Sb \mid b \mid A \mid bb
\end{aligned}
$$

Find an equivalent CFG without unit productions.

Exercise

Consider the CFG

$$\begin{aligned} S &\rightarrow ASB \mid AB \\ A &\rightarrow aAS \mid aA \mid a \\ B &\rightarrow SbS \mid bS \mid Sb \mid b \mid A \mid bb \end{aligned}$$

Find an equivalent CFG without unit productions.

$$\begin{aligned} S &\rightarrow ASB \mid AB \\ A &\rightarrow aAS \mid aA \mid a \\ B &\rightarrow SbS \mid bS \mid Sb \mid b \mid\mid aAS \mid aA \mid a \mid bb \end{aligned}$$

Simplification Sequence

If G is a CFG that generates a language with at least one string different from ϵ, then there exists a CFG G_1 without ϵ productions, unit productions, and useless symbols such that $L(G_1) = L(G) - \{\epsilon\}$.

In this order:

1. Eliminate ϵ-productions
2. Eliminate unit productions
3. Eliminate useless symbols

Chomsky Normal Form

All CFGs $G = (V, T, P, S)$ without ϵ can have a grammar in CNF, free of useless symbols, where all productions are of the form:

- $A \rightarrow BC$, where A, B, $C \in V$
- $A \rightarrow a$, where $A \in V$ and $a \in T$

To achieve this, start with a grammar free of ϵ-productions, unit productions, and useless symbols. Then:

- For each terminal a that appears in a production body of length 2 or more, create a new variable A.
  - This variable has only one production: $A \rightarrow a$.
  - Now, use A to replace a in production bodies of length 2 or more.
- For productions $A \rightarrow B_1 B_2 \dots B_k$, with $k \geq 3$, break down the bodies into cascading productions in the form:

  $A \rightarrow B_1 C_1$
  $C_1 \rightarrow B_2 C_2$
  $C_2 \rightarrow B_3 C_3$
  ...

Example

Consider the CFG of expressions

$$
\begin{aligned}
E &\to E + T \mid T \times F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
T &\to T \times F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F &\to (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
I &\to a \mid b \mid Ia \mid Ib \mid I0 \mid I1
\end{aligned}
$$

Create variable for the terminal symbols that appear in productions' bodies:

$$
\begin{array}{ll}
A \to a & B \to b \\
Z \to 0 & O \to 1 \\
P \to + & M \to \times \\
L \to (& R \to)
\end{array}
$$

cont.

Substitute the terminal by those variables:

$$\begin{aligned}
E &\to EPT \mid TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
T &\to TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
F &\to LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
I &\to a \mid b \mid IA \mid IB \mid IZ \mid IO
\end{aligned}$$

Substitute the bodies of production with size greater than 3:

$$\begin{aligned}
E &\to EC_1 \mid TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
T &\to TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
F &\to LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
C_1 &\to PT \\
C_2 &\to MF \\
C_3 &\to ER
\end{aligned}$$

CNF for Languages with ϵ

- When the language L of the original grammar includes ϵ, the language of the CNF grammar excludes ϵ.
- In practice, it's common to add a new initial variable to the CNF grammar that has two productions, one producing the initial variable of the CNF grammar and the other producing ϵ.
- For example, if $S \to AB$ is the start variable of the CNF grammar, you can add the following variable to generate ϵ:

$$S_1 \to S \mid \epsilon$$

S_1 is now the start variable.

Leituras

[HMU07] Chap. 7.1, 7.2

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman.
Introduction to Automata Theory, Languages and Computation.
3rd edition, 2007.