Theory of Computation

L.EIC, 2nd Year

João M. P. Cardoso
Email: jmpc@acm.org

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO
DEI DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Outline

- Pumping Lemma for CFLs
- Chomsky Normal Form (CNF)
- Properties of CFLs
- Decision Problems about CFLs

Pumping Lemma for CFLs

For infinite CFLs

Pumping Lemma for CFLs

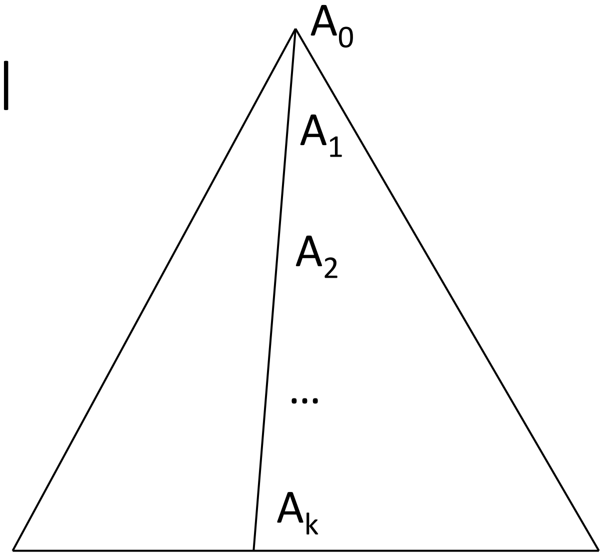
- Assume L is a CFL. There exists a constant n such that for every z in L with $|z| \geq n$ we can write z=uvwxy
  - $|vwx| \leq n$ (the middle part is not too long)
  - $vx \neq \varepsilon$ (at least one, v or x, is not the empty string)
  - For all $i \geq 0$, $uv^iwx^iy \in L$ (double pumping starting in 0)

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 $A \rightarrow a$)
- With *n* variables and without repetition:
  - The longest string *z* considering *n* variables occurs with $S \rightarrow V_1 V_2 \ldots V_{n-1}$ 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)

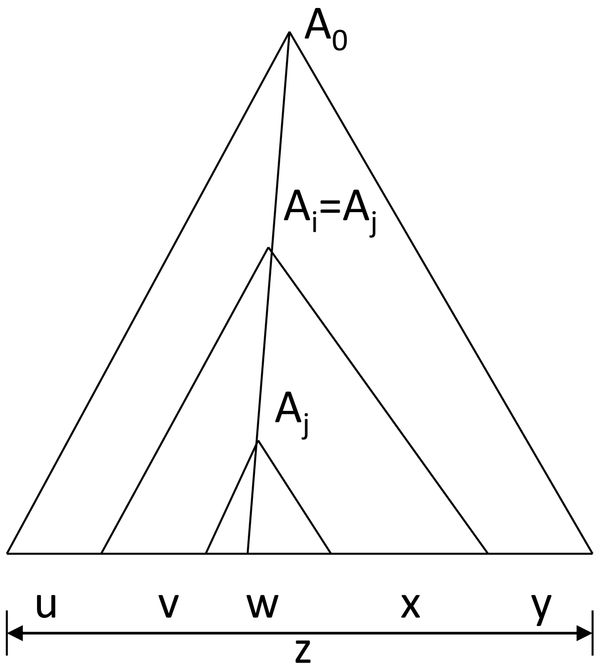
Insights of the Pumping Lemma for CFLs

- Grammar contains **n variables**, string z in L, $|z| \geq n$
- To have a yield of length $\geq n$, there must be repetitions of variables in the tree (there must be recursivity)



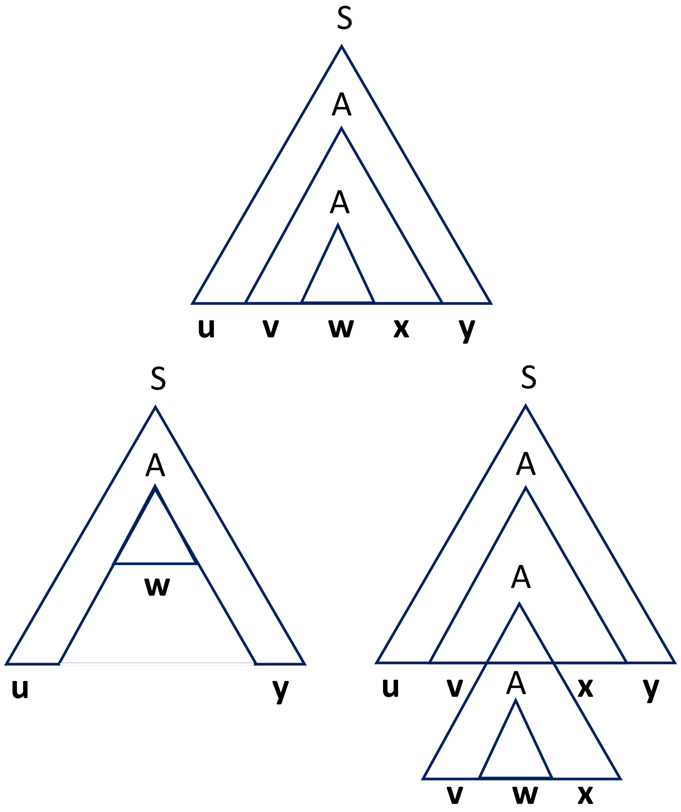
Insights of the Pumping Lemma for CFLs

- Let's split the tree:
  - w is the string in the leaves of the subtree A_j
  - v and x are such that vwx is the string represented by the subtree A_i (at least one v or x is not null)
  - u and y are the parts of z in the left and in the right of vwx, respectively



Insights of the Pumping Lemma for CFLs

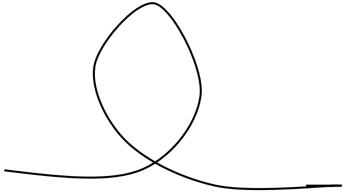
- As $A_i = A_j$, we can
  - Substitute the subtree of A_i by the subtree of A_j, obtaining the case i=0, uwy
  - Substitute the subtree of A_j by the subtree of A_i, obtaining the case i=2, uv^2wx^2y, and repeat for i=3, … (pumping)
- With repetitions of at least one variable A, one can have S → U A Y and A → W | V W X where $|VX| \neq \varepsilon$



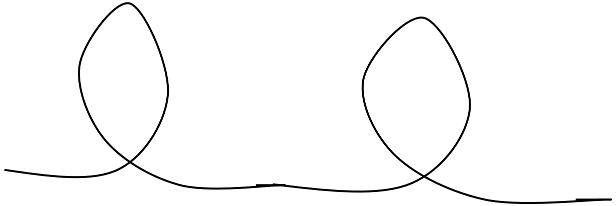
Insights of the Pumping Lemma for CFLs

- Previously, we considered that the terminals were alone in each variable (productions A→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



RL (DFA)



CFL (CFG)

- In case of RL: the pumping lemma results from the fact that the number of states of a DFA is finite
  - To accept a string sufficiently long the processing needs to repeat states
- In case of CFL: the pumping lemma results from the fact that the number of symbols in a CFG is finite
  - To accept a string sufficiently long the derivations must repeat variables

Prove that a Language is not a CFL

- Assume 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
  - $|vwx| \leq n$
  - $vx \neq \varepsilon$
  - For all $i \geq 0$, $uv^iwx^iy \in L$

- Consider $L = \{0^k1^k2^k \mid k \geq 1\}$. Show that L is not a CFL.
  - Supposing that L is a CFL, then there exist a constant ***n*** indicated by the pumping lemma; Let's select $z = 0^n1^n2^n$ which belongs to L and $|z|=3n \geq n$
  - Decomposing $z=uvwxy$, such that $|vwx| \leq n$ and v, x not both the empty string, we have *vwx* which cannot contain simultaneously 0s and 2s
  - **In case vwx does not contain 2s**: then *vx* includes only 0s and 1s and has at least one symbol. Then by the pumping lemma, *uwy* would have to belong to L, but it has *n* 2s and less than *n* 0s or 1s and thus not belong to L.
  - **In case vwx does not contain 0s**: similar argument.
  - We obtain the contradiction in both cases. Thus, the hypothesis is false and **L is not a CFL** *qed*

Using the lemma to prove

- Consider $L = \{0^k 1^k \mid k \geq 0\}$. Prove that L is not a CFL
  - Supposing that L is a CFL, then exists a constant n indicated by the pumping lemma; Let's select $z = 0^n 1^n$ ($n \geq 0$) which belongs to L
  - Decomposing z=uvwxy, such that $|vwx| \leq n$ and v, x are not both the empty string, and if we select $v=0^n$ e $x=1^n$
  - In this case $uv^i wx^i y$ belongs to L
  - *We do not obtain the intended contradiction as we found a decomposition of z that meets the pumping lemma*
- We cannot prove that L is not a CFL: because it is a CFL!
- Note that we can select a decomposition of z that does not meet the lemma and that cannot lead us to conclude that L is not a CFL!!!

Chomsky Normal Form (CNF)

A form to represent Context-Free Grammars (CFGs)

Simplification of CFGs

- Elimination of non-useful **symbols**
  - Useful symbol: $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$, $w \in T^*$
  - Generator symbol: $X \overset{*}{\Rightarrow} w$
    - Any terminal is generator of itself!
  - Reachable symbol: $S \overset{*}{\Rightarrow} \alpha X \beta$
  - Useful = generator + reachable
  - Eliminate first the non-generators and then the non-reachable

- Example
  - $S \rightarrow AB \mid a$
  - $A \rightarrow b$

  - $S \rightarrow a$ [B is not generator]

  - $S \rightarrow a$
  - $A \rightarrow b$ [A is not reachable]

  - then:
  - $S \rightarrow a$

Elimination of Non-useful Symbols

- Algorithm: identify the generator symbols
  - Terminals are generators
  - $A \rightarrow \alpha$ and α only has generators then A is generator
- Algorithm: identify the reachable symbols
  - S is reachable
  - A is reachable, $A \rightarrow \alpha$; then all the symbols in α are reachable

Elimination of ε-Productions

- Nullable variables: $A \overset{*}{\Rightarrow} \varepsilon$
- Transformation:
  - $B \rightarrow CAD$ is transformed in $B \rightarrow CD \mid CAD$ and A is changed to anymore derive ε
- Algorithm: identify the nullable variables
  - $A \rightarrow C_1\ C_2 \ldots C_k$, if all C_i are nullables then A is nullable
- If a language L has a CFG then L-$\{\varepsilon\}$ has a CFG without ε-productions
  - Identify all the nullable symbols
  - For each $A \rightarrow X_1\ X_2 \ldots X_k$ if m X_i's are nullables substitute by 2^m productions with all the combinations of presences of X_i
    - Exception: if m=k, we don't include the case of all X_i removed
  - Productions $A \rightarrow \varepsilon$ are eliminated

Example 1

- Grammar G:
  - S → AB
  - A → aAA | ε
  - B → bBB | ε

(1) A and B are nullable, then S is nullable as well:

- S → AB | A | B
- A → aAA | aA | aA | a
- B → bBB | bB | b

(2) Grammar without ε-productions:

- S → AB | A | B
- A → aAA | aA | a
- B → bBB | bB | b

- In this case, L(new grammar) = L(G) – {ε}

Elimination of Unit Productions

- Unit production: A → B, where A and B are variables
  - They can be useful in the elimination of ambiguity (example: language of arithmetic expressions)
  - They are not unavoidable as they introduce extra steps in derivations
- Elimination by expansion (see example in next slides)

Example 2: Elimination of Unit Productions

- Elimination by expansion (E is the start variable)
  - E → T | E + T
  - T → F | T × F
  - F → I | (E)
  - I → a | b | Ia | Ib | I0 | I1
- From E → T we can step to E → F | T × F, and E → I | (E) | T × F, and finally to E → a | b | Ia | Ib | I0 | I1 | (E) | T × F
  - Problem in the case of cycles (A → B, B → C, C → A)

Example 2: Elimination of Unit Productions

- Algorithm: determine all the unit pairs, derived only with unit productions
  - (A, A) is a unit pair
  - (A, B) is a unit pair and B → C, C variable; then (A, C) is a unit pair
- Example: (E, E), (T, T), (F, F), (E, T), (E, F), (E, I), (T, F), (T, I), (F, I)
- Elimination: substitute the existent productions in order that each unit pair (A, B) includes all the productions of the form $A \rightarrow \alpha$ in which $B \rightarrow \alpha$ is a non unit production (includes A=B)

Example 2: Grammar without Unit Productions

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

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)	$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Simplification Sequence

- If G is a CFG which generates a language with at least one string different from ε, there exist a CFG G_1 without ε-productions, unit productions and non-useful symbols and $L(G_1) = L(G) - \{\varepsilon\}$
  - Eliminate ε-productions
  - Eliminate unit productions
  - Eliminate non-useful symbols

Chomsky Normal Form (CNF)

- All the CFLs without ε have a grammar in CNF, without non-useful symbols and in which all productions have the form:
  - $A \rightarrow BC$ (A, B, C are variables) or
  - $A \rightarrow a$ (A is a variable and "a" is a terminal)
- Transformation
  - Start with a grammar without ε-productions, unit productions or non-useful symbols
  - Keep the productions $A \rightarrow a$
  - Transform all bodies with length greater or equal than 2 into bodies consisting of only variables
    - New variables D for terminals in those bodies, substitute $D \rightarrow d$
  - Split bodies of length greater or equal than 3 in cascade productions with the form $A \rightarrow B_1B_2 \ldots B_k$ in $A \rightarrow B_1C_1$, $C_1 \rightarrow B_2C_2$, …

Example 2: Conversion to CNF

- Grammar of expressions

$E \rightarrow T \mid E + T$

$T \rightarrow F \mid T \times F$

$F \rightarrow I \mid (E)$

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

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

- Variables for the terminals in bodies are isolated
  - $A \rightarrow a$ $\quad B \rightarrow b$ $\quad Z \rightarrow 0$ $\quad O \rightarrow 1$
  - $P \rightarrow +$ $\quad M \rightarrow \times$ $\quad L \rightarrow ($ $\quad R \rightarrow)$
- Substitute terminals by those variables
  - $E \rightarrow EPT \mid TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
  - $T \rightarrow TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
  - $F \rightarrow LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
  - $I \rightarrow a \mid b \mid IA \mid IB \mid IZ \mid IO$

Example 2: Conversion to CNF

- $A \rightarrow a \quad B \rightarrow b \quad Z \rightarrow 0 \quad O \rightarrow 1$
- $P \rightarrow + \quad M \rightarrow \times \quad L \rightarrow (\quad R \rightarrow)$
- $E \rightarrow E\mathbf{PT} \mid TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
- $T \rightarrow TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
- $F \rightarrow LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
- $I \rightarrow a \mid b \mid IA \mid IB \mid IZ \mid IO$

- Substitute long bodies
  - $E \rightarrow E\mathbf{C_1} \mid TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
  - $T \rightarrow TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
  - $F \rightarrow LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
  - $\mathbf{C_1} \rightarrow PT$
  - $C_2 \rightarrow MF$
  - $C_3 \rightarrow ER$

Example 2: Conversion to CNF

CFG original (E is the start variable):

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

CFG in CNF:

- $E \rightarrow EC_1 \mid TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
- $T \rightarrow TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
- $F \rightarrow LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$
- $C_1 \rightarrow PT$
- $C_2 \rightarrow MF$
- $C_3 \rightarrow ER$
- $I \rightarrow a \mid b \mid IA \mid IB \mid IZ \mid IO$

$A \rightarrow a$

$B \rightarrow b$

$Z \rightarrow 0$

$O \rightarrow 1$

$P \rightarrow +$

$M \rightarrow \times$

$L \rightarrow ($

$R \rightarrow)$

Exercise 1

- Consider the grammar and perform the following steps:
  - $S \rightarrow ASB \mid \varepsilon$
  - $A \rightarrow aAS \mid a$
  - $B \rightarrow SbS \mid A \mid bb$

a) Eliminate the ε-productions

b) Eliminate the unit productions

c) Eliminate the non-useful symbols

d) Write the grammar in the Chomsky Normal Form (CNF)

CNF in Practice

- When the language L of the original grammar includes ε, the language of the CNF grammar excludes ε
- In practice it is common to add a new start variable to the CNF grammar which has two productions,
  - one producing the start variable of the CNF grammar and
  - the other producing ε
- Example:
  - Being S → AB the start variable of the CNF grammar
  - One can add the following variable to have a grammar generating ε
    - S1 → S | ε (S1 is now the start variable)

Closure Properties of CFLs

Substitution

- Be Σ an alphabet; foreach of its symbols *a* define a function (substitution) which associates a language L_a to the symbol
  - Strings: if $w= a_1 \dots a_n$ then $s(w)$ is the language of all the strings $x_1 \dots x_n$ such that x_i is in $s(a_i)$
  - Languages: $s(L)$ is the union of all $s(w)$ such that $w \in L$
- Example:
  - $\Sigma=\{0,1\}$, $s(0)=\{a^n b^n \mid n \geq 1\}$, $s(1)=\{aa,bb\}$
  - If $w=01$, $s(w) = s(0)s(1) = \{a^n b^n aa \mid n \geq 1\} \cup \{a^n b^{n+2} \mid n \geq 1\}$
  - If $L=L(0^*)$, $s(L) = (s(0))^* = a^{n1} b^{n1} \dots a^{nk} b^{nk}$, for n1, …, nk
- ***Theorem: if L is a CFL and s() a substitution, which associates to each symbol a CFL, then s(L) is a CFL.***

The CFLs are closed for:

- Union
- Concatenation
- Closure (*)
- Substitution
- Homomorphism and homomorphism inverse
- Reverse
- Intersection with an RL
  - Note: intersection with a CFL is not guaranteed to result in a CFL! (see next slide)

CFLs and Intersection

- Consider $L_1 = \{0^n1^n2^i \mid n\geq1, i\geq1\}$ and $L_2 = \{0^i1^n2^n \mid n\geq1, i\geq1\}$
- L_1 and L_2 are CFLs

▶ S → AB	S → AB
▶ A → 0A1 | 01	A → 0A | 0
▶ B → 2B | 2	B → 1B2 | 12

- $L_1 \cap L_2 = \{0^n1^n2^n \mid n\geq1\}$
  - Has been already proved that it is not a CFL
- Thus, ***the CFLs are not closed for the intersection***

Test if String belongs to a CFL

The Example of the CYK algorithm (note that there are other alternatives, including the use of a PDA)

Test if String belongs to a CFL

- Cocke-Younger-Kasami (CYK) Algorithm : CFG in CNF
  - X_{ij} – represents the set of variables that produce string[i:j]
  - $O(n^3)$, using dynamic programming, fill of a table

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
a_1	a_2	a_3	a_4	a_5

S → AB | BC

A → BA | a

B → CC | b

C → AB | a

Input string: baaba

{S,A,C}				
	{S,A,C}			
	{B}	{B}		
{S,A}	{B}	{S,C}	{S,A}	
{B}	{A,C}	{A,C}	{B}	{A,C}
b	a	a	b	a

X_{12}: $X_{11}X_{22}$; X_{24}: $X_{22}X_{34} \cup X_{23}X_{44}$

Conclusion: answer is positive if S is in X_{15}; and is negative, otherwise

Seminal papers about CNF and CYK

- CNF:
  - Noam Chomsky, “**On Certain Formal Properties of Grammars**.” *Inf. Control.* 2 (1959): 137-167. https://doi.org/10.1016/S0019-9958(59)90362-6
- CYK:
  - John Cocke, “**Programming languages and their compilers: Preliminary notes,”** New York University, USA, 1969.
  - T. Kasami, “**An efficient recognition and syntax algorithm for context-free languages,”** Scientific report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA, 1965.
  - D. Younger, “**Recognition and parsing of context-free languages in time n^3,**” Information and Control, 10, pp. 189-208, 1967. https://doi.org/10.1016/S0019-9958(67)80007-X
  - Itiroo Sakai, “**Syntax in universal translation,**” In Proceedings 1961 International Conference on Machine Translation of Languages and Applied Language Analysis, Her Majesty’s Stationery Office, London, p. 593-608, 1962. https://aclanthology.org/1961.earlymt-1.31.pdf