Theory of Computation

MIEIC, 2nd Year

João M. P. Cardoso

Dep. de Engenharia Informática
Faculdade de Engenharia (FEUP)
Universidade do Porto
Porto
Portugal

Email: jmpc@acm.org

U. PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Outline

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

Chomsky Normal Form (CNF)

Chomsky Normal Form (CNF)

- Simplification of CFGs → Chomsky normal form (CNF)
- 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 → AB | a
  - A → b

  - S → a [B is not generator]

  - S → a
  - A → b [A is not reachable]

  - then:
  - S → 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 not produce ε anymore
- 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_is 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

- Grammar:
  - S → AB
  - A → aAA | ε
  - B → bBB | ε
- A and B are nullable, then S is nullable as well
  - S → AB | A | B
  - A → aAA | aA | aA | a
  - B → bBB | bB | b
- Grammar without ε-productions:
  - S → AB | A | B
  - A → aAA | aA | a
  - B → bBB | bB | b

Elimination of Unit Productions

- Unit production: A $\rightarrow$ 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; introduce extra steps in derivations
- Elimination by expansion (see example in next slides)

Example: 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 a 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)

Elimination of Unit Productions

- Algorithm: determine all the unit pairs, derived only with unit productions
  - (A, A) is an unit pair
  - (A, B) is an unit pair and B $\rightarrow$ C, C variable; then (A, C) is an 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 $\rightarrow$ α in which B $\rightarrow$ α is a non unit production (includes A=B)

Grammar Without Unit Productions

I → a | b | Ia | Ib | I0 | I1

F → I | (E)

T → F | T × F

E → T | E + T

(E is the start variable)

Pair	Productions
(E, E)	E → E + T
(E, T)	E → T × F
(E, F)	E → (E)
(E, I)	E → a | b | Ia | Ib | I0 | I1
(T, T)	T → T × F
(T, F)	T → (E)
(T, I)	T → a | b | Ia | Ib | I0 | I1
(F, F)	F → (E)
(F, I)	F → a | b | Ia | Ib | I0 | I1
(I, I)	I → a | b | Ia | Ib | I0 | 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 whithout ε have a gramar in CNF, without non-useful symbos and in which all productions have the form:
  - A → BC (A, B, C are variables) or
  - A → 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 → 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 → d
  - Split bodies of length greater or equal then 3 in cascade productions with the form $A \rightarrow B_1B_2 \ldots B_k$ for $A \rightarrow B_1C_1$, $C_1 \rightarrow B_2C_2$, …

Example: Conversion to CNF

- Grammar of expressions

E → T | E + T

T → F | T × F

F → I | (E)

I → a | b | Ia | Ib | I0 | I1

Productions
E → E + T
E → T × F
E → (E)
E → a | b | Ia | Ib | I0 | I1
T → T × F
T → (E)
T → a | b | Ia | Ib | I0 | I1
F → (E)
F → a | b | Ia | Ib | I0 | I1
I → a | b | Ia | Ib | I0 | I1

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

Example: Conversion to CNF

- A → a B → b Z → 0 O → 1
- P → + M → × L → (R →)
- E → E**PT** | TMF | LER | a | b | IA | IB | IZ | IO
- T → TMF | LER | a | b | IA | IB | IZ | IO
- F → LER | a | b | IA | IB | IZ | IO
- I → a | b | IA | IB | IZ | IO

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

Example: 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)

Pumping Lemma for 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)

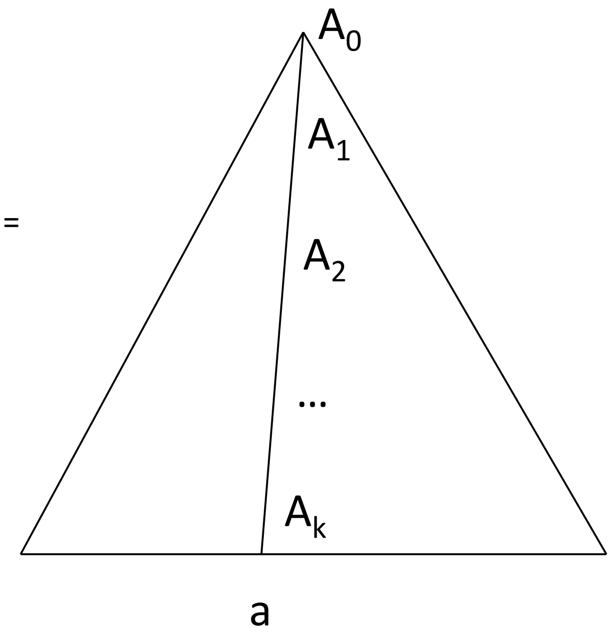
Pumping Lemma for CFLs

- Let's focus on the size of the syntax (analysis) tree
- Consider only the case of the CNF:
  - binary trees in which the leaves are terminals alone (productions $A \rightarrow a$)
  - In a syntax tree with w in the leaves, if the length of the longest path is n

    then $|w| \leq 2^{n-1}$

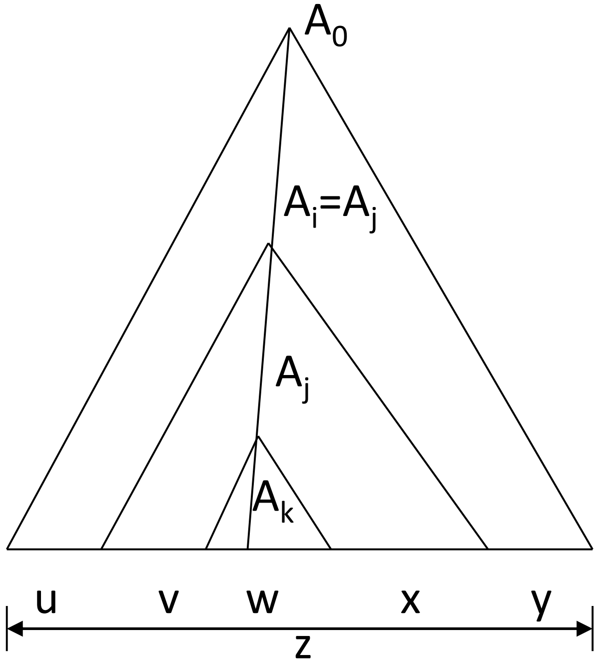
Proof of the Pumping Lemma for CFLs

- Obtain a CNF grammar G for L
- G contains m variables. Select $n=2^m$. String z in L $|z| \geq n$.
- Any analysis tree with the longest path until m represents strings until $2^{m-1} = n/2$
  - z would be too long; tree for z has longest path greater or equal then m+1
- In the right figure, the path $A_0 \ldots A_k a$ has length k+1, $k \geq m$
  - There is at least m+1 variables in the path; thus there is at least one repetition of variables (from A_{k-m} to A_k).
  - Assume $A_i = A_j$ with $k-m \leq i < j \leq k$



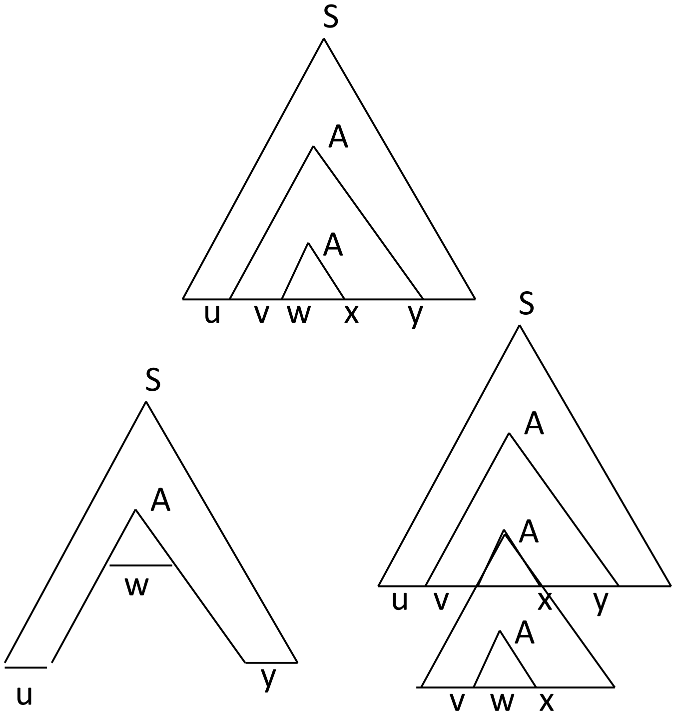
Proof of the Pumping Lemma for CFLs

- If the string z is sufficiently long, there must be a repetition of symbols
- 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 (as there aren’t unitary productions 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

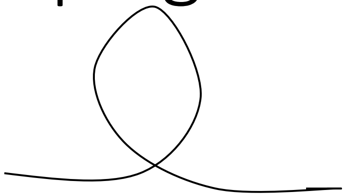


Proof of the Pumping Lemma for CFLs

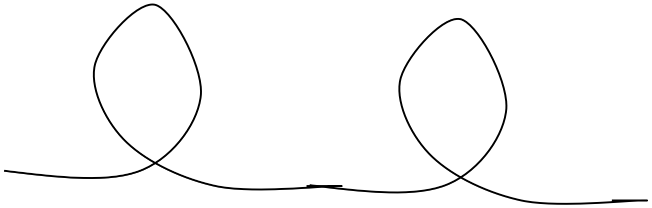
- As $A_i=A_j$, we can
  - Substitute the subtree of A_i by the subtree of A_j, obtaining the case i=0, uwy.
  - Substitute the subtree of A_j by the subtree of A_i, obtaining the case i=2, uv^2wx^2y and repeat for i=3, … (pumping)
- $|vwx|\leq n$ because we took in a A_i near the bottom of the tree, $k-i\leq m$, longest path of A_i to m+1, string $2^m=n$



Pumping Lemma



LR (DFA)



CFL (CFG)

- In case of LR: the pumping lemma results from the fact that the number of states of a DFA is finite
  - To accept a string with 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 symbols

Prove that a Language is not a CFL

- Consider $L = \{0^k 1^k 2^k \mid k \geq 1\}$. 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 $z = 0^n 1^n 2^n$ 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 dos 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 dos not contain 0s**: similar argument.
  - We obtain the contradiction in both cases; thus the hypothesis is false and **L is not a CFL**

Problems in Proofs

- Be $L = \{0^k 1^k \mid k \geq 1\}$. Show 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 $z = 0^n 1^n$ $(n \geq 1)$ 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^iwx^iy belongs to L
  - We do not obtain the intended contradiction
- We cannot prove that L is not a CFL
  - Because it is a CFL!

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= $a_1 \ldots a_n$ then s(w) is the language of all the strings $x_1 \ldots x_n$ 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:
  - $\Sigma=\{0,1\}$, $s(0)=\{a^n b^n \mid n \geq 1\}$, $s(1)=\{aa,bb\}$
  - Se w=01, s(w) = s(0)s(1) = $\{a^n b^n aa \mid n \geq 1\} \cup \{a^n b^{n+2} \mid n \geq 1\}$
  - Se L=L(0*), s(L) = (s(0))* = $a^{n1}b^{n1} \ldots a^{nk}b^{nk}$, para n1, …, nk qq
- 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 (*)
- Homomorphism and homomorphism inverse
- Reverse
- Intersection with an LR
  - Note: intersection with a CFL is not guaranteed to result in a CFL

CFL and Intersection

- Consider $L_1 = \{0^n1^n2^i \mid n\geq 1, i\geq 1\}$ and $L_2 = \{0^i1^n2^n \mid n\geq 1, i\geq 1\}$
- L_1 and L_2 are CFLs

$S \rightarrow AB$	$S \rightarrow AB$
$A \rightarrow 0A1 \mid 01$	$A \rightarrow 0A \mid 0$
$B \rightarrow 2B \mid 2$	$B \rightarrow 1B2 \mid 12$

- $L_1 \cap L_2 = \{0^n1^n2^n \mid n\geq 1\}$
  - Has been already proved that it is not a CFL
- Thus, the CFLs are not closed for the intersection

Decision Properties for CFLs

Test if a Language is Empty

- Verify if S is generator
  - With adequate data structure is O(n)
  - See Hopcroft's book

Test if String Belongs to a CFL

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

Input string: baaba

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

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

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

Undecidable Problems

- There are not algorithms to answer to the following questions:
  - Is a given CFG ambiguous?
  - Is a given CFL inherently ambiguous?
  - The intersection of two CFLs is an empty language?
  - Two given CFLs define the same language?
  - A given CFL is the language Σ^*, where Σ is the alphabet?