Automata Theory and Formal Languages
Class 5

Marcin Luckner, PhD
mluckner@mini.pw.edu.pl

Version 1.2
23 listopada 2021

Context-free language

- A context-free grammar generates a context-free language.
- The only way to show that a language is context-free is the construction of grammar.
- It can be shown that a language is not context-free using the contraposition of pumping lemma for context-free languages.
- The Cocke–Younger–Kasami (CYK) algorithm allows us to show that a word belongs to the language.

Contraposition of pumping lemma

Contraposition of pumping lemma

If for each n_L exists $z \in L$ such that

$$(|z| \geqslant n_L) \wedge$$
$$[(\forall_{u,v,w,x,y} z = uvwxy \wedge |vwx| \leqslant n_L \wedge |vx| \geqslant 1)$$
$$\exists_{i=0,1,2\ldots} z_i = uv^i wx^i y \notin L]$$

then the language is not context-free.

Comparison of lemmata

- The lemmata for regular and context-free languages are similar but differs in several important aspects.

Regular language

- $z = uvw$
- $|uv| \leqslant n_L$
- $|v| \geqslant 1$
- $z_i = uv^i w$

Context-free language

- $z = uvwxy$
- $|vwx| \leqslant n_L$
- $|vx| \geqslant 1$
- $z_i = uv^i wx^i y$

Discussion of lemmata

- An additional segment u in $z = uvwxy$ causes that a block of elements that can be change goes through the whole words.
- The block width is n_L, so its influence can be reduced by suitable z.
- We can pump two elements int the same time $z_i = uv^i wx^i y$. Therefore, it is harder to create a word outside the language.

Contraposition of pumping lemma - example I

Using the contraposition of pumping lemma show that language $L = \{a^i b^j : j = i^2\}$ over alphabet $\Sigma = \{a, b\}$ is not context-free.

Contraposition of pumping lemma - example II

$\forall n \in N$ word

$$z = a^n b^{n*n}$$

belongs to language L.
Because $|vwx| \leqslant n$ we have to discuss three cases:

1. $vwx = a^k \implies v = a^p \wedge x = a^q$
2. $vwx = b^k \implies v = b^p \wedge x = b^q$
3.

$$\begin{aligned} vwx = a^k b^j \implies & (v = a^p \wedge x = a^q b^j) \vee \\ & (v = a^k b^p \wedge x = b^q) \vee \\ & (v = a^p \wedge x = b^q) \end{aligned}$$

where $k, j \in \{1, \ldots, n\} \wedge p, q \in \{0, \ldots, n\}$.

Contraposition of pumping lemma - example III

- Case 1: $(v = a^p \wedge x = a^q)$ choose $i = 0$.
  - then $z_0 = a^{n-(p+q)} b^{n*n} \notin L$
- Case 2: $(v = b^p \wedge x = b^q)$ choose $i = 0$.
  - then $z_0 = a^n b^{n*n-(p+q)} \notin L$

Contraposition of pumping lemma - example IV

- Case 3: there are three sub cases including two similar.
  - Case 1 ($v = a^p \wedge x = a^q b^j$): choose $i = 2$
    - after multiplication
      $z_2 = a^{n-k}(a^p)(a^p)a^{k-p-q}(a^q b^j)(a^q b^j)b^{n*n-j}$
    - so $z_2 = a^{n+p} b^j a^q b^{n*n} \notin L$
  - Case2 ($v = a^k b^p \wedge x = b^q$): choose $i = 2$
    - after multiplication
      $z_2 = a^{n-k}(a^k b^p)(a^k b^p)b^{j-p-q}(b^q)(b^q)b^{n*n-j}$
    - so $z_2 = a^n b^p a^k b^{n*n+q} \notin L$

Contraposition of pumping lemma - example IV

- In the last case ($v = a^p \wedge x = b^q$): choose $i = 2$.
- Then $z_2 = a^{n+p} b^{n*n+q}$.
- However z_2 can be in the language if $(n+p)^2 = n^2 + q$.
- Let's suppose that $p = 1$. In such case the number of b letters equals $(n+1)^2 = n^2 + 2n + 1$.
- That means that $q > n$ and we know that $|vwx| \leqslant n$, so it is not possible.
- For $p > 1$ value q grows even more rapidly. Therefore $z_2 = a^{n+p} b^{n*n+q} \notin L$
- So, the language is not context-free.

Algorithm Cocke–Younger–Kasami

- Algorithm Cocke–Younger–Kasami (CYK) allows us to determine if word w of length n is generated by grammar G.
- The input grammar must be in CNF.
- The algorithm is based on dynamic programming.

Algorithm CYK

1. We split word w into n one letter strings and find nonterminals that generate the letters. For that, we are looking for productions $A \rightarrow a$, where $a \in T$.
2. Knowing nonterminals that generate for w substrings of length k, we are looking for nonterminals that generate substrings of length $k+1$:
   - 2.1 We split a string of length $k+1$ into all possible pairs of substrings.
   - 2.2 Each prefix and corresponding suffix are already defined by nonterminals.
   - 2.3 We find all nonterminals A with production $A \rightarrow BC$, where $B, C \in V$ generate prefix and suffix.
   - 2.4 The founded nonterminals generate substrings of length $k+1$.
3. Finally, we get a set of nonterminals that generate a substring of length n that means word w. The grammar generates word w $\iff$ the set contains the starting symbol S.

Algorithm CYK - example I

Use the algorithm CYK to check if word *anna* belongs to the language generated by the following gramma
$G = \{V = \{A, B, C, D, E, F, N, S\}, T = \{a, n, w\}, P, S\}$

P:

$$S \to NN|NA$$
$$N \to BB|BD|w$$
$$B \to n$$
$$D \to NB$$
$$A \to CC|EC|CF$$
$$C \to a$$
$$E \to CA$$
$$F \to NC$$

Algorithm CYK - example II

P:

$$S \rightarrow NN|NA$$
$$N \rightarrow BB|BD|w$$
$$B \rightarrow n$$
$$D \rightarrow NB$$
$$A \rightarrow CC|EC|CF$$
$$C \rightarrow a$$
$$E \rightarrow CA$$
$$F \rightarrow NC$$

- Find variables that generate terminals

C	B	B	C
a	n	n	a

Algorithm CYK - example III

P:

$$S \rightarrow NN|NA$$
$$N \rightarrow BB|BD|w$$
$$B \rightarrow n$$
$$D \rightarrow NB$$
$$A \rightarrow CC|EC|CF$$
$$C \rightarrow a$$
$$E \rightarrow CA$$
$$F \rightarrow NC$$

- Find variables that generate substrings of length 2

$\emptyset$	N_{BB}	$\emptyset$	
C	B	B	C
a	n	n	a

Algorithm CYK - example IV

P:

$$S \to NN|NA$$
$$N \to BB|BD|w$$
$$B \to n$$
$$D \to NB$$
$$A \to CC|EC|CF$$
$$C \to a$$
$$E \to CA$$
$$F \to NC$$

- Find variables that generate substrings of length 3

$\emptyset$	F_{NC}		
$\emptyset$	N_{BB}	$\emptyset$	
C	B	B	C
a	n	n	a

Algorithm CYK - example V

P:

$$S \to NN|NA$$
$$N \to BB|BD|w$$
$$B \to n$$
$$D \to NB$$
$$A \to CC|EC|CF$$
$$C \to a$$
$$E \to CA$$
$$F \to NC$$

- Find variables that generate word *anna*

A_{CF}			
$\emptyset$	F_{NC}		
$\emptyset$	N_{BB}	$\emptyset$	
C	B	B	C
a	n	n	a

- Variable A generates word $w = anna$, but $w \notin L$ because $S \neq A$.

Algorithm CYK - example VI

- Is word *wanna* in the language?

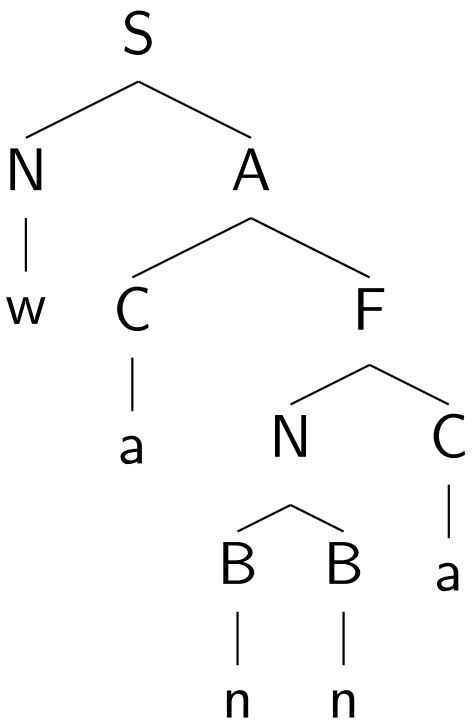
P:

$$S \to NN|NA$$
$$N \to BB|BD|w$$
$$B \to n$$
$$D \to NB$$
$$A \to CC|EC|CF$$
$$C \to a$$
$$E \to CA$$
$$F \to NC$$

S_{NA}				
$\emptyset$	A_{CF}			
$\emptyset$	$\emptyset$	F_{NC}		
F_{NC}	$\emptyset$	N_{BB}	$\emptyset$	
N	C	B	B	C
w	a	n	n	a

- Word $w =$ *wanna* belongs to the language because it can be generated from S

Derivation tree



- By indexing nonterminals to show which production was used, we can reconstruct a derivation of the word
- A derivation is reconstructed by a tree called the derivation tree.
- A word can have more than one derivation tree.

Assignments I

1. Use the algorithm CYK to check if words
   - 1.1 *baaba*, *aaaaa*, *aaaaaa* belong to the following language
     $G = \{V = \{A, B, C, S\}, T = \{a, b\}, P, S\}$
     $$P: \quad S \to AB|BC$$
     $$A \to BA|a$$
     $$B \to CC|b$$
     $$C \to AB|a$$
   - 1.2 (()(())) belongs to the following language
     $G = \{V = \{S\}, T = \{C,)\}, P, S\}$
     $$P: \ S \to SS|(S)|\epsilon$$

Assignments II

2. Using the contraposition of pumping lemma show that the following languages are not context-free.
   - 2.1 Language L over alphabet $\Sigma = \{a, b, c\}$, such that the number of occurrences of symbols a, b, c is equal. $abbacc \in L$, $accaabb \notin L$
   - 2.2 Language $L = \{a^i b^j c^k : i \leqslant j \wedge j \leqslant k\}$ over alphabet $\Sigma = \{a, b, c\}$
   - 2.3 Language $L = \{a^i b^j c^k : i \neq j \wedge j \neq k\}$ over alphabet $\Sigma = \{a, b, c\}$