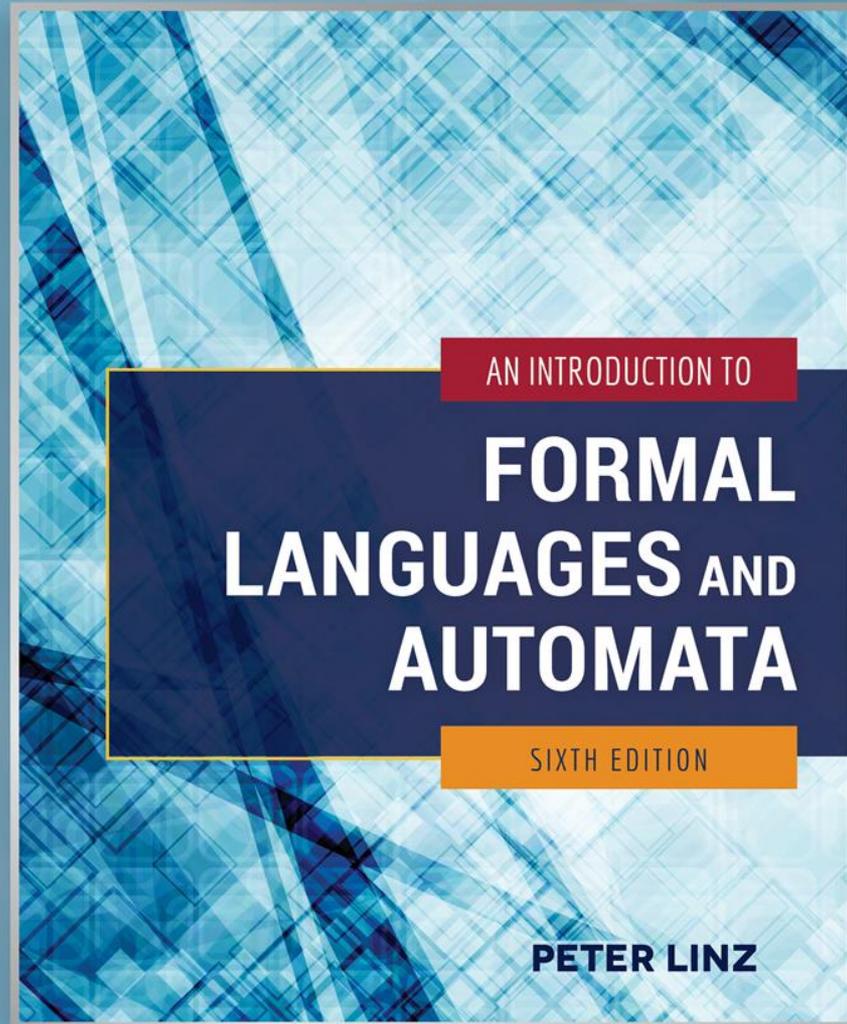


# Chapter 8

PROPERTIES OF  
CONTEXT-FREE  
LANGUAGES



# Learning Objectives

*At the conclusion of the chapter, the student will be able to:*

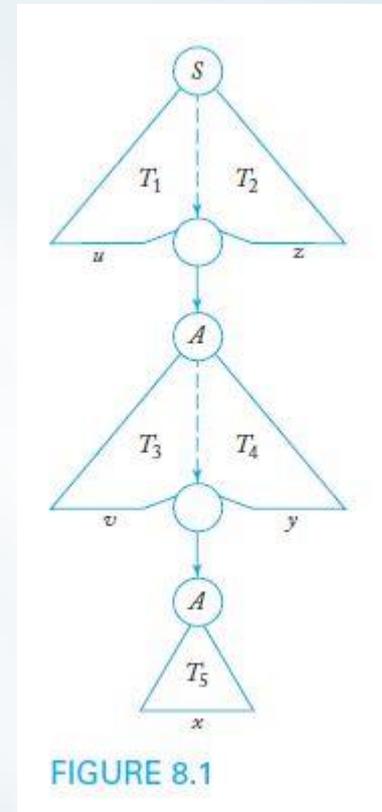
- Apply the pumping lemma to show that a language is not context-free
- State the closure properties applicable to context-free languages
- Prove that context-free languages are closed under union, concatenation, and star-closure
- Prove that context-free languages are not closed under either intersection or complementation
- Describe a membership algorithm for context-free languages
- Describe an algorithm to determine if a context-free language is empty
- Describe an algorithm to determine if a context-free language is infinite

# A Pumping Lemma for Context-Free Languages

- Theorem 8.1: Given an infinite context-free language  $L$ , every sufficiently long string  $w$  in  $L$  can be broken into four parts  $uvxyz$  such that
  - $|vy| \geq 1$
  - $|vxy| \leq m$  (where  $m$  is an arbitrary integer  $\leq |w|$ )
  - An arbitrary, but equal number of repetitions of  $v$  and  $y$  yields another string in  $L$
- The “pumped” string consists of two separate parts ( $v$  and  $y$ ) and can occur anywhere in the string
- The pumping lemma can be used to show that, by contradiction, a certain language is not context-free

# An Illustration of the Pumping Lemma for Context-Free Languages

As shown in Figure 8.1, the pumping lemma for context-free languages can be illustrated by sketching a general derivation tree that shows a decomposition of the string into the required components



# Closure Properties for Context-Free Languages

- Theorem 8.3 states that if  $L_1$  and  $L_2$  are context-free languages, so are the languages that result from the following operations:
  - $L_1 \cup L_2$
  - $L_1 \cap L_2$
  - $L_1^*$
- In other words, the family of regular languages is closed under union, intersection, and star-closure.
- To prove these properties, we assume the existence of two context-free grammars  $G_1$  and  $G_2$  that generate the respective languages

# Proof of Closure under Union

- Assume that  $L_1$  and  $L_2$  are generated by the context-free grammars  $G_1 = (V_1, T_1, S_1, P_1)$  and  $G_2 = (V_2, T_2, S_2, P_2)$
- Without loss of generality, assume that the sets  $V_1$  and  $V_2$  are disjoint
- Create a new variable  $S_3$  which is not in  $V_1 \cup V_2$
- Construct a new grammar  $G_3 = (V_3, T_3, S_3, P_3)$  so that
  - $V_3 = V_1 \cup V_2 \cup \{S_3\}$
  - $T_3 = T_1 \cup T_2$
  - $P_3 = P_1 \cup P_2$
- Add to  $P_3$  a production that allows the new start symbol to derive either of the start symbols for  $L_1$  and  $L_2$   
$$S_3 \rightarrow S_1 \mid S_2$$
- Clearly,  $G_3$  is context-free and generates the union of  $L_1$  and  $L_2$ , thus completing the proof

# Proof of Closure under Concatenation

- Assume that  $L_1$  and  $L_2$  are generated by the context-free grammars  $G_1 = (V_1, T_1, S_1, P_1)$  and  $G_2 = (V_2, T_2, S_2, P_2)$
- Without loss of generality, assume that the sets  $V_1$  and  $V_2$  are disjoint
- Create a new variable  $S_4$  which is not in  $V_1 \cup V_2$
- Construct a new grammar  $G_4 = (V_4, T_4, S_4, P_4)$  so that
  - $V_4 = V_1 \cup V_2 \cup \{S_4\}$
  - $T_4 = T_1 \cup T_2$
  - $P_4 = P_1 \cup P_2$
- Add to  $P_4$  a production that allows the new start symbol to derive the concatenation of the start symbols for  $L_1$  and  $L_2$   
 $S_4 \rightarrow S_1 S_2$
- Clearly,  $G_4$  is context-free and generates the concatenation of  $L_1$  and  $L_2$ , thus completing the proof

# Proof of Closure under Star-Closure

- Assume that  $L_1$  is generated by the context-free grammars  $G_1 = (V_1, T_1, S_1, P_1)$
- Create a new variable  $S_5$  which is not in  $V_1$
- Construct a new grammar  $G_5 = (V_5, T_5, S_5, P_5)$  so that
  - $V_5 = V_1 \cup \{S_5\}$
  - $T_5 = T_1$
  - $P_5 = P_1$
- Add to  $P_5$  a production that allows the new start symbol  $S_5$  to derive the repetition of the start symbol for  $L_1$  any number of times
$$S_5 \rightarrow S_1 S_5 \mid \lambda$$
- Clearly,  $G_5$  is context-free and generates the star-closure of  $L_1$ , thus completing the proof

# No Closure under Intersection

- Unlike regular languages, the intersection of two context-free languages  $L_1$  and  $L_2$  does not necessarily produce a context-free language
- As a counterexample, consider the context-free languages

$$L_1 = \{ a^n b^n c^m : n \geq 0, m \geq 0 \}$$

$$L_2 = \{ a^n b^m c^m : n \geq 0, m \geq 0 \}$$

- However, the intersection  $L_1$  and  $L_2$  is the language

$$L_3 = \{ a^n b^n c^n : n \geq 0 \}$$

- $L_3$  can be shown not to be context-free by applying the pumping lemma for context-free languages

# No Closure under Complementation

- The complement of a context-free language  $L_1$  does not necessarily produce a context-free language
- The proof is by contradiction: given two context-free languages  $L_1$  and  $L_2$ , assume that their complements are also context-free
- By Theorem 8.3, the union of the complements must also produce a context-free language  $L_3$
- Using our assumption, the complement of  $L_3$  is also context-free
- However, using the set identity below, we conclude that the complement of  $L_3$  is the intersection of  $L_1$  and  $L_2$ , which has been shown not to be context-free, thus contradicting our assumption.

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}.$$

# Elementary Questions about Context-Free Languages

- Given a context-free language  $L$  and an arbitrary string  $w$ , is there an algorithm to determine whether or not  $w$  is in  $L$ ?
- Given a context-free language  $L$ , is there an algorithm to determine if  $L$  is empty?
- Given a context-free language  $L$ , is there an algorithm to determine if  $L$  is infinite?
- Given two context-free grammars  $G_1$  and  $G_2$ , is there an algorithm to determine if  $L(G_1) = L(G_2)$ ?

# A Membership Algorithm for Context-Free Languages

- The combination of Theorems 5.2 and 6.5 confirms the existence of a membership algorithm for context-free languages
- By Theorem 5.2, exhaustive parsing is guaranteed to give the correct result for any context-free grammar that contains neither  $\lambda$ -productions nor unit-productions
- By Theorem 6.5, such a grammar can always be produced if the language does not include  $\lambda$
- Alternatively, a npda to accept the language can be constructed as established by Theorem 7.1

# Determining Whether a Context-Free Language is Empty

- Theorem 8.6 confirms the existence of an algorithm to determine if a context-free language  $L(G)$  is empty
- For simplicity, assume that  $\lambda$  is not in  $L(G)$
- Apply the algorithm for removing useless symbols and productions
- If the start symbol is found to be useless, then  $L(G)$  is empty; otherwise,  $L(G)$  contains at least one string

# Determining Whether a Context-Free Language is Infinite

- Theorem 8.7 confirms the existence of an algorithm to determine if a context-free language  $L(G)$  is infinite
- Apply the algorithms for removing  $\lambda$ -productions, unit-productions, and useless productions
- If  $G$  has a variable  $A$  for which there is a derivation that allows  $A$  to produce a sentential form  $xAy$ , then  $L(G)$  is infinite
- Otherwise,  $L(G)$  is finite
- Can be implemented by building a dependency graph which contains an edge from  $A$  to  $B$  for every rule of the form  $A \rightarrow xBy$

# Determining Whether Two Context-Free Languages are Equal

- Given two context-free grammars  $G_1$  and  $G_2$ , is there an algorithm to determine if  $L(G_1) = L(G_2)$ ?
- If the languages are finite, the answer can be found by performing a string-by-string comparison
- However, for general context-free languages, no algorithm exists to determine equality