Chapter 9

Nature of Context-Free Languages

1. Prove that the set of all regular languages is a proper subset of the set of all context-free languages.

   Solution: Every regular language is context-free since every regular language has a right-linear grammar which is also a context-free grammar. Thus, regular languages are a subset of context-free languages. To show that they are a proper subset, we must show that at least one context-free language is not regular. We have already shown in Chapter 6 that several languages are not regular. Some of them, for example, wcw^R are in fact context-free. Thus, regular languages are a proper subset of context-free languages. (See also, Theorems 27-29 in Appendix B.)

2. We have seen that elements of the syntax of high-level programming languages such as arithmetic expressions, nested parentheses and nested if-then-else statements are all context-free languages. Assuming that programs written in the language are made up of a sequence of such expressions and statements each of which is context-free, show that the set of all such programs is also context-free.

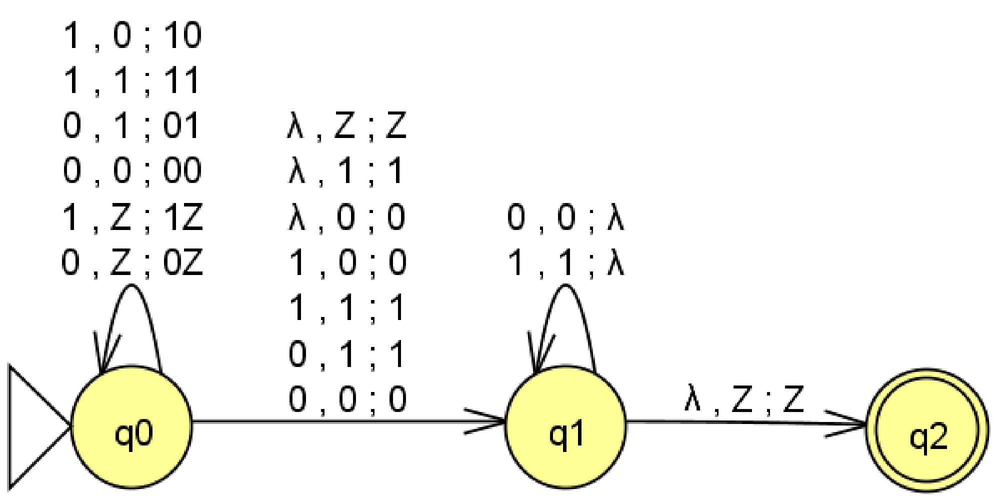
   Solution: We know that context-free languages are closed under concatenation and * closure. Therefore the above statement is true.

3. Show that the set of all binary strings which are palindromes and, when interpreted as positive integers, are divisible by 3, is context-free.

   Solution: Please see also the VIDEO SOLUTION for this exercise. The set of all binary strings which are palindromes is context-free:

$$S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \lambda$$

   We can also construct a PDA to accept all odd and even palindromes:



   The set of all binary strings that are divisible by 3 is regular (see the DFA in Figure-2.8). We know that the intersection of a context-free language with a regular language is context-free. Hence the given statement is true.

4. Give an example of a context-free language whose complement is not context-free (other than the example given in Sec. 9.1.4).

Solution: We know that $a^{2m}b^mc^m$ is not context-free (this can be shown using the Pumping Lemma; intuitively, this language involves two comparisons of counts: number of a s to be twice the number of b s and number of b s to be equal to number of c s; both are not possible using the single stack of a PDA) . Its complement is the subset of $a^*b^*c^*$ where either of the two pairs of numbers do not match (i.e., it has more a s than twice the number of b s, or less a s than twice the number of b s, or more b s than the number of c s, or less b s than the number of c s). This language is context-free since we can construct a CFG for it:

LHS		RHS	LHS		RHS
S	→	A	S	→	C
S	→	B	S	→	D
A	→	Ac	C	→	aC
A	→	aaAb	C	→	bCc
A	→	aA	C	→	bC
A	→	a	C	→	b
B	→	Bc	D	→	aD
B	→	aaBb	D	→	bDc
B	→	Bb	D	→	Dc
B	→	b	D	→	c

Thus, we have a context-free language $a^{2m}b^nc^k$, $m \neq n$, or $n \neq k$, whose complement $a^{2m}b^mc^m$ is not context-free.

5. Show that the set of all strings over {a, b, c} that are either even palindromes or odd palindromes is context-free.

Solution: Please see also the VIDEO SOLUTION for this exercise. The set of all even palindromes over the given alphabet is context-free:

$$S \to aSa \mid bSb \mid cSc \mid \lambda$$

The set of all odd palindromes is also context-free:

$$S \to aSa \mid bSb \mid cSc \mid a \mid b \mid c$$

We know that the union of two context-free languages is context-free. Therefore the given language is context-free.

6. Can we show that the set of all strings over {a, b, c} that are neither even palindromes nor odd palindromes is context-free? Explain.

Solution: Yes, but not using the idea of complementation. Consider the set of all strings that are either even palindromes or odd palindromes. This language is context-free (since we can easily construct either a CFG or a PDA for it). Its complement, the set of all strings that are neither even palindromes nor odd palindromes may or may not be context-free since there is no guarantee that the complement of a context-free language is context-free.

However, for this problem, it is possible to construct a PDA (or a CFG) directly for the set of all strings that are neither even nor odd palindromes. See Figure-9.4 below (Exercise 12) for a PDA for all strings that are not odd palindromes (assuming that *c* occurs only as a separator for the two parts of the input string). Modifying this so that it accepts only those that are neither even nor odd palindromes is not straightforward. This is due to the nondeterministic nature of even palindromes. A nondeterministic PDA for even palindromes cannot be complemented easily by exchanging its final and non-final states (since one of the choices tried in parallel will succeed and even palindromes will also be accepted by such a PDA).

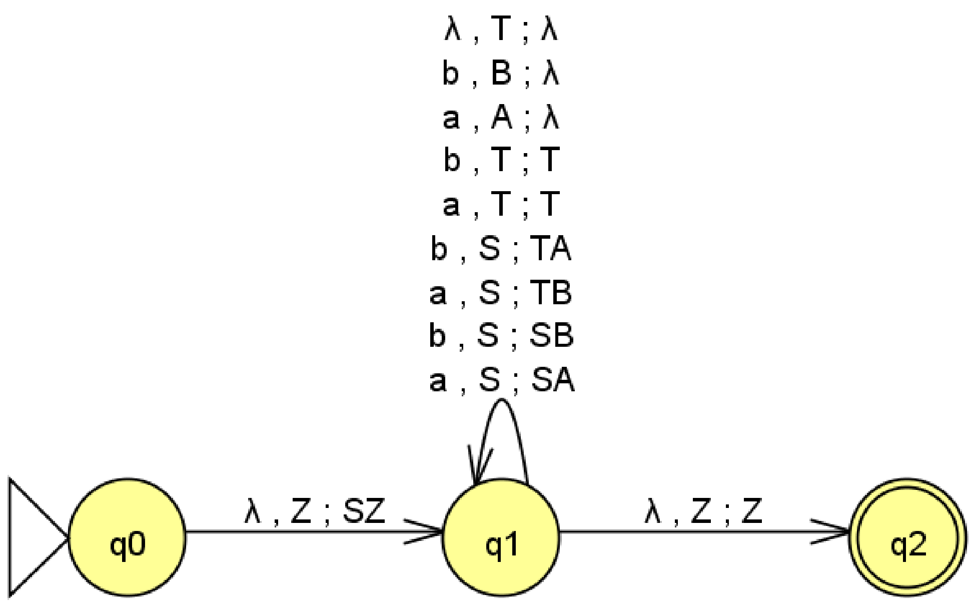
However, we will see how to construct a CFG and a PDA directly. A CFG for all strings that are not odd or even palindromes is (with only *a* and *b*):

LHS		RHS
S	$\rightarrow$	aSa
S	$\rightarrow$	bSb
S	$\rightarrow$	aAb
S	$\rightarrow$	bAa
A	$\rightarrow$	λ
A	$\rightarrow$	aA
A	$\rightarrow$	bA

This grammar can be converted to GNF:

LHS		RHS
S	$\rightarrow$	aSA
S	$\rightarrow$	bSB
S	$\rightarrow$	aTB
S	$\rightarrow$	bTA
T	$\rightarrow$	λ
T	$\rightarrow$	aT
T	$\rightarrow$	bT
A	$\rightarrow$	a
B	$\rightarrow$	b

A corresponding PDA is:



Since we could construct both a CFG and a PDA, the given language is certainly context-free, but not because its complement is context-free.

7. If a programming language is context-free, every program that can be written in the language can be generated from the grammar of the language. Can we show that the set of all invalid programs, that is, those with syntax errors, is not context-free? Explain.

   Solution: This is not possible since the complement of a context-free language may or may not be context-free. As such, the set of all programs with syntax errors may or may not be context-free. We either have to construct a CFG/PDA for it to show that it is context-free or we have to use the Pumping Lemma for context-free languages to show that it is not context-free. Both are not easy, in general.

8. XHTML is a version of HTML that meets the stricter requirements of XML (e.g., case-sensitivity, proper nesting of tags, etc.). Assuming that both HTML and XML are context-free languages, can we say that XHTML is a context-free language?

   Solution: Not by directly combining the two languages since the intersection of two context-free languages need not be context-free. On the other hand, it is quite likely that we can construct a CFG for XHTML to show that it is indeed context-free.

9. We know that the set of strings $w = (a + b)^*$ is a regular language and therefore also a context-free language. We also know that the concatenation of two context-free languages is a context-free language. Consider the set of concatenations $w.w$. We showed in Example 9.2 that this language is not context-free. Is this a contradiction? Is this the language of the following grammar? Explain.

   $S \rightarrow AA, A \rightarrow aA \mid bA \mid \lambda$

Solution: There is no contradiction. This is similar to Exercise 2 in Chapter 6. The language of *w.w* is the set of all strings where the first and second halves are identical. By simply concatenating any string from $(a + b)^*$ with any (other) string from the same set, we do not get *w.w*. Instead, we get $(a + b)^*$ itself! The language of the above grammar is also just $(a + b)^*$, not *w.w*. In the grammar, there is no control over how the first *A* is expanded in relation to how the second *A* is expanded; there is no way to construct a CFG for *ww*.

10. What if in *w.w*, $w = (ab)^* + (ba)^*$? Is this language context-free or not?

    Solution: Yes, this language is context-free. In fact, it is regular! A RegEx for this language is:

    $$(abab)^* + (baba)^*$$

    A right-linear grammar for this language is:

    $S \to A \mid B, A \to ababA \mid \lambda, B \to babaB \mid \lambda$

11. Is there a deterministic language that is not context-free?

    Solution: Yes. Languages such as $a^nb^nc^n$ or $a^nb^na^nb^n$, $n > 0$, are deterministic but not context-free. They are deterministic context-sensitive languages.

    In fact, it is not known whether deterministic and non-deterministic linear-bounded automata (the machines of context-sensitive languages, see Chapter 11) are equivalent. Deterministic and non-deterministic Turing Machines (see Chapter 10) are however equivalent.

12. Consider the PDA shown in Figure-9.4 which accepts all strings over {*a, b, c*} that are *not* odd palindromes (with *c* occurring only as the separator). Since we could construct a PDA for it, this language must be context-free. However, its complement – the set of odd palindromes – is also context-free since we can easily construct a PDA or a CFG for it. We also know that the complement of any context-free language may not be context-free. Is there a contradiction here? Explain.

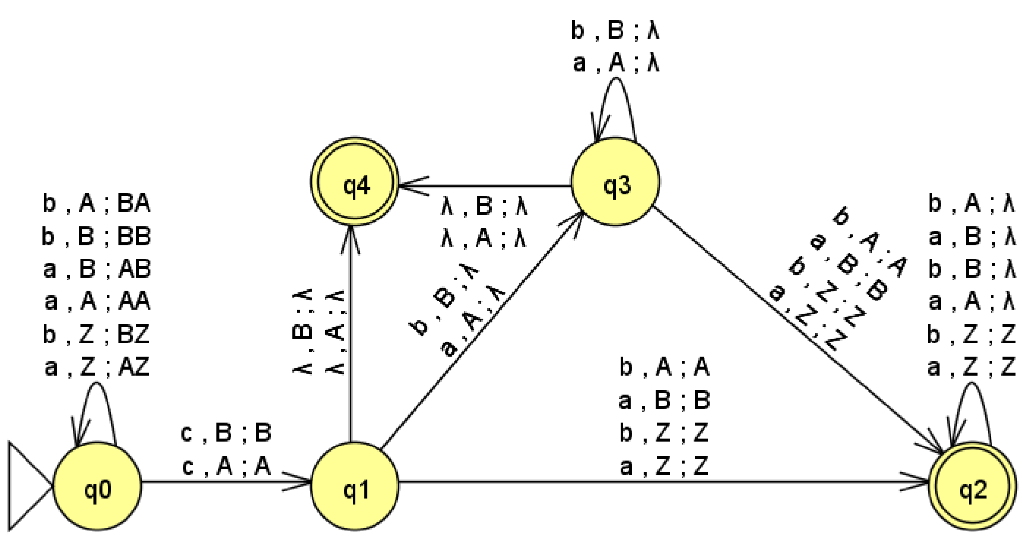


Figure-9.4 PDA for all strings that are not odd-palindromes

Solution: There is no contradiction here. The complement of a context-free language may very well be also context-free. There is just no guarantee that the complement of every context-free language is context-free. In fact, there are context-free languages whose complements are not context-free.

13. Are the languages of the following two grammars (with start symbols S_1 and S_2) the same? How can we be sure?
$S_1 \to aS_1b \mid \lambda$
$S_2 \to ATB \mid \lambda,\ T \to CTD \mid S_2,\ C \to A,\ D \to B,\ A \to a,\ B \to b$

Solution: Applying the method for eliminating unit productions (see Sec. 7.10), the second grammar can be reduced to:

$S_2 \to ATB \mid \lambda,\ T \to ATB \mid S_2,\ A \to a,\ B \to b$

and then further by applying the substitution rule (see Sec. 7.10 and Theorem 11 in Appendix B) to:

$S_2 \to aTb \mid \lambda,\ T \to aTb \mid S_2$

Now, substituting for T in S_2, we get:

$S_2 \to aS_2b \mid \lambda \mid aaS_2bb$

This is the same as the other grammar. Hence the languages of the two grammars are the same.

It may be noted however that there is no general method to compare two context-free grammars (see Sec. 12.7).

14. Show that $a^nb^mc^m$, where $n \neq m$, is not a context-free language.

Solution: Let us apply the Pumping Lemma in a proof by contradiction to show that the language is not context-free. Let us assume that it is context-free and choose $w = a^{m-1}b^mc^m$ where m is the unknown Pumping Lemma constant. When w is partitioned as $uvxyz$, whenever v and y contain only b s, only c s, or an unequal number of b s and c s, pumping them up or down will result in a mismatch between the number of b s and the number of c s and thus the string no longer belongs to the language. When v and y are such that they contain equal number numbers of b s and c s, pumping them up or down will make their number equal to the number of a s and once again the string does not belong to the language. We can choose a string with a corresponding number of a s to ensure this. Thus, in all cases, there is a contradiction and the given language is not context-free. (See also, Ogden's Lemma in Exercise 21 below.)

15. Show that $a^nb^mc^m$, where $n = 2m$, is not a context-free language. Why can't we construct a PDA that pushes a symbol onto the stack for each a and pops one symbol for each b or c?

Solution: Let us apply the Pumping Lemma in a proof by contradiction to show that the language is not context-free. Let us assume that it is context-free and choose $w = a^{2m}b^mc^m$ where m is the unknown Pumping Lemma constant. When w is partitioned as $uvxyz$, all possible cases result in imbalances and in a string that does not belong to the language. This is a contradiction and the given language is not context-free.

Reasoning in terms of a PDA, the given strategy only ensures that the sum of *b* s and *c* s is equal to the number of *a* s; it allows for the possibility of unequal numbers of *b* s and *c* s. In fact, the language of such a PDA is the language of addition, not the given language.

16. Show that $a^n b^m c^k$, where $n < m < k$, is not a context-free language.

    Solution: Let us apply the Pumping Lemma in a proof by contradiction to show that the language is not context-free. Let us assume that it is context-free and choose $w = a^{m-1}b^m c^{m+1}$ where m is the unknown Pumping Lemma constant. When w is partitioned as *uvxyz*, in all possible cases, we can pump up or down to violate one of the required inequalities, thus setting up a contradiction. The given language is not context-free.

17. Show that $a^n b^m c^k$, where $k = m^n$, is not a context-free language.

    Solution: Let us apply the Pumping Lemma in a proof by contradiction to show that the language is not context-free. Let us assume that it is context-free and choose $w = a^m b^m c^k$ where $k = m^m$ and m is the unknown Pumping Lemma constant. When w is partitioned as *uvxyz*, all cases result in strings where the relation between k and m is no longer valid. This is a contradiction and thus the given language is not context-free.

18. Show that division is not a context-free language.

    Solution: The language of unary division is $a^m b^n c^{m / n}$. Let us apply the Pumping Lemma in a proof by contradiction to show that the language is not context-free. Let us assume that it is context-free and choose $w = a^m b^m c^1$ where m is the unknown Pumping Lemma constant. When w is partitioned as *uvxyz*, all cases result in strings in which the number of *c* s is not equal to the number of *a* s divided by the number of *b* s. In fact, the numbers may not even divide evenly once we pump the string up or down. Thus, the language of division is not context-free.

19. Is the set of all strings of the form $a^n b^m c^k$, where $k = m \times n$ and $k < 1000$, $m \geq 0$, $n \geq 0$, a context-free language or not?

    Solution: This is a finite language. Therefore, it is a regular language (see Theorem 30 in Appendix B) and also a context-free language.

20. Show that the set of all strings of the form 1^p, where p is a prime number (i.e., prime numbers represented in the unary number system), is not context-free.

    Solution: Let us apply the Pumping Lemma in a proof by contradiction to show that the language is not context-free. Let us assume that it is context-free and choose $w = a^p$ where p is the m^{th} prime number (counting upwards from 2). When w is partitioned as *uvxyz*, the repeating patterns can contain anywhere from 1 to m number of *a* s. Pumping up and down will result in a number that is not a prime, that is, we don't get the $(m - 1)$th prime or the $(m + 1)$th prime. Thus, we get strings that do not belong to the language. Because of this contradiction, we must conclude that the language of prime numbers is not context-free.

21. There is a stronger version of the Pumping Lemma for context-free languages known as *Ogden's Lemma* which states that we need not consider the entire string w but just the symbols at any m (or more) chosen positions in the string w. The string w is partitioned into $uvxyz$ as in the Pumping Lemma. The rest of the formulation and application of the lemma are also similar to that of the Pumping Lemma except that we now say that the two repeating patterns, that is vy, must contain at least one of the chosen symbols. Ogden's Lemma can be used to show that certain languages are not context-free even though the regular Pumping Lemma fails to set up a contradiction for such languages. One such language is $0^n1^n2^k$, where $n \neq k$. Show using Ogden's Lemma that this language is not context-free.

    Solution: Consider an application of Ogden's Lemma in a proof by contradiction to show that the given language is not context-free. Let us assume that it is context-free and choose $w = a^{m+1}b^{m+1}c^m$ where m is the unknown Ogden's Lemma constant. Let any m symbols be marked in this string. When w is partitioned as $uvxyz$, whenever v and y contain only marked a s, only marked b s, or an unequal number of marked a s and marked b s, pumping them up or down will result in a mismatch between the number of a s and the number of b s and thus the string no longer belongs to the language. When v and y are such that they contain equal number numbers of marked a s and b s, pumping them up or down will make their number equal to the number of c s and once again the string does not belong to the language. We can choose a string with a corresponding number of marked c s to ensure this. Thus, in all cases, there is a contradiction and the given language is not context-free.