

## Example 4

**Question:** write a grammar for  $a^n b^n$

**Answer:** Sure! Here's a context-free grammar for the language of strings of the form " $a^n b^n$ ":

$S \rightarrow ab$   
 $S \rightarrow aSb$

Explanation:

- The start symbol is  $S$ .
- The production rule  $S \rightarrow ab$  generates the base case where there is one "a" followed by one "b".
- The production rule  $S \rightarrow aSb$  generates strings with one or more "a"s followed by an equal number of "b"s.

Using this grammar, you can derive strings of the form " $a^n b^n$ " where  $n$  is a positive integer. For example, applying the production rules, you can derive the following strings:

$S \rightarrow aSb \rightarrow aaSbb \rightarrow aaasbbb \rightarrow aaaaSbbbb \rightarrow aaaaabbbbb$

Each application of the production rule  $S \rightarrow aSb$  adds one "a" and one "b" to the string.

Please note that this grammar only generates strings of the form " $a^n b^n$ ". It does not generate strings with unmatched "a"s or "b"s or any other variations.

---

**Question:** design an NPDA for  $a^n b^n$

**Answer:** Sure! Here's a design for a Non-Deterministic Pushdown Automaton (NPDA) that recognizes the language of strings of the form " $a^n b^n$ ":

States:

- $q_0$  (initial state)
- $q_1$
- $q_2$
- $q_3$  (final state)

Alphabet:  $\{a, b\}$

Stack alphabet:  $\{Z_0\}$  ( $Z_0$  is the initial stack symbol)

Transitions:

1.  $(q_0, a, Z_0) \rightarrow (q_1, Z_0)$
2.  $(q_1, a, Z_0) \rightarrow (q_1, aZ_0)$
3.  $(q_1, b, a) \rightarrow (q_2, \epsilon)$
4.  $(q_2, b, a) \rightarrow (q_2, \epsilon)$
5.  $(q_2, \epsilon, Z_0) \rightarrow (q_3, Z_0)$

Explanation:

- The NPDA starts in the initial state  $q_0$  with the initial stack symbol  $Z_0$  on top of the stack.
- When it reads an 'a' in state  $q_0$ , it transitions to state  $q_1$  and pushes  $Z_0$  on the stack.
- In state  $q_1$ , for each 'a' encountered, it stays in  $q_1$  and pushes 'a' on the stack.
- When it reads a 'b' in state  $q_1$ , it transitions to state  $q_2$  and pops an 'a' from the stack.
- In state  $q_2$ , for each 'b' encountered, it stays in  $q_2$  and pops an 'a' from the stack.
- After reading all the 'b's, if the stack is empty (epsilon transition in  $q_2$ ), it transitions to the final state  $q_3$ .

Acceptance:

- The NPDA accepts the input string if it ends in the final state  $q_3$  and the stack is empty.

This NPDA recognizes the language of strings of the form " $a^n b^n$ ", where  $n$  is a non-negative integer. It matches the number of 'a's with the number of 'b's by using the stack to keep track of the 'a's.

---

**Question:** design an NPDA for  $a^n b^n$  and  $\Sigma = \{a, b\}$

**Answer:** Here's the correct design for an NPDA that recognizes the language of strings of the form " $a^n b^n$ ":

States:

- $q_0$  (initial state)
- $q_1$
- $q_2$
- $q_3$
- $q_4$  (final state)

Alphabet:  $\{a, b\}$

Stack alphabet:  $\{Z_0\}$  ( $Z_0$  is the initial stack symbol)

Transitions:

1.  $(q_0, \epsilon, Z_0) \rightarrow (q_1, Z_0)$
2.  $(q_1, a, Z_0) \rightarrow (q_1, aZ_0)$
3.  $(q_1, b, Z_0) \rightarrow (q_1, bZ_0)$
4.  $(q_1, \epsilon, \epsilon) \rightarrow (q_2, Z_0)$
5.  $(q_2, a, a) \rightarrow (q_2, \epsilon)$
6.  $(q_2, b, b) \rightarrow (q_2, \epsilon)$

7.  $(q_2, \epsilon, Z_0) \rightarrow (q_3, Z_0)$
8.  $(q_3, a, a) \rightarrow (q_3, \epsilon)$
9.  $(q_3, b, b) \rightarrow (q_3, \epsilon)$
10.  $(q_3, \epsilon, Z_0) \rightarrow (q_4, Z_0)$

Explanation:

- The NPDA starts in the initial state  $q_0$  with the initial stack symbol  $Z_0$  on top of the stack.
- In state  $q_1$ , it reads the input symbols and pushes them onto the stack.
- After reading all symbols, it transitions to state  $q_2$  to reach the middle of the string.
- In state  $q_2$ , it pops the symbols from the stack without comparing them to the input string.
- In state  $q_3$ , it matches each symbol on the stack with the remaining input symbols by popping them.
- If all symbols are matched and the stack is empty, it transitions to the final state  $q_4$ .

Acceptance:

- The NPDA accepts the input string if it ends in the final state  $q_4$  and the stack is empty.

This NPDA recognizes the language of strings of the form " $ww^R$ ", where  $w$  is any string of symbols from the alphabet  $\{a, b\}$  and  $w^R$  denotes the reverse of  $w$ . The NPDA pushes the input symbols onto the stack, then reaches the middle of the string, and finally matches the symbols in a deterministic manner to check for the palindrome-like structure.

**Question:** prove that  $a^n b^n$  is not regular

**Answer:** To prove that the language  $L = \{a^n b^n \mid n \geq 0\}$  is not regular, we can use the pumping lemma for regular languages. The pumping lemma states that if  $L$  is a regular language, there exists a pumping length  $p$  such that any string  $s$  in  $L$  with  $|s| \geq p$  can be divided into three parts:  $s = xyz$ , where:

1.  $|xy| \leq p$ ,
2.  $|y| > 0$ , and
3. for all  $k \geq 0$ , the string  $xy^kz$  is also in  $L$ .

Now, let's assume that  $L = \{a^n b^n \mid n \geq 0\}$  is regular and consider a pumping length  $p$  for this language.

Let's take the string  $s = a^p b^p$ . According to the pumping lemma, we can divide  $s$  into three parts:  $s = xyz$ , where  $|xy| \leq p$  and  $|y| > 0$ .

Now, consider the possible cases for the placement of  $x$ ,  $y$ , and  $z$  within  $s$ :

1. If  $y$  consists only of  $a$ 's (i.e.,  $y = a^k$  for some  $k > 0$ ), then pumping up ( $k > 1$ ) will result in a string that contains more  $a$ 's than  $b$ 's, violating the condition that the number of  $a$ 's should be equal to the number of  $b$ 's.
2. If  $y$  consists of both  $a$ 's and  $b$ 's (i.e.,  $y = a^m b^k$  for some  $m > 0$  and  $k > 0$ ), then pumping up ( $k > 1$ ) will either introduce more  $a$ 's after the  $b$ 's or move the  $b$ 's ahead of the  $a$ 's, violating the condition that the  $a$ 's should come before the  $b$ 's.

3. If  $y$  consists only of  $b$ 's (i.e.,  $y = b^k$  for some  $k > 0$ ), then pumping down ( $k > 1$ ) will result in a string that contains more  $b$ 's than  $a$ 's, violating the condition that the number of  $a$ 's should be equal to the number of  $b$ 's.

In all cases, pumping up or down will lead to a string that is not in the language  $L = \{a^n b^n \mid n \geq 0\}$ . Therefore, we have reached a contradiction, which means that our assumption that  $L$  is regular is incorrect.

Hence, we can conclude that the language  $L = \{a^n b^n \mid n \geq 0\}$  is not a regular language.

**Question:** This is wrong. Cases 2 and 3 cannot happen because  $|xy| \leq p$

**Answer:** Apologies for the incorrect response. You are correct, cases 2 and 3 cannot happen in this particular language. Let's provide a corrected proof:

To prove that the language  $L = \{a^n b^n \mid n \geq 0\}$  is not regular, we can use the pumping lemma for regular languages. The pumping lemma states that if  $L$  is a regular language, there exists a pumping length  $p$  such that any string  $s$  in  $L$  with  $|s| \geq p$  can be divided into three parts:  $s = xyz$ , where:

1.  $|xy| \leq p$ ,
2.  $|y| > 0$ , and
3. for all  $k \geq 0$ , the string  $xy^kz$  is also in  $L$ .

Let's assume that  $L = \{a^n b^n \mid n \geq 0\}$  is a regular language. According to the pumping lemma, there exists a pumping length  $p$  for  $L$ .

Consider the string  $s = a^p b^p$ , where  $|s| = 2p \geq p$ . According to the pumping lemma, we can write  $s$  as  $s = xyz$ , satisfying the conditions of the lemma.

Based on the pumping lemma conditions, we have the following possibilities for the division of  $s = xyz$ :

1. The substring  $y$  consists of only  $a$ 's ( $y = a^k$ , where  $0 < k \leq p$ ).
  - In this case, pumping up ( $k > 1$ ) will result in a string where the number of  $a$ 's exceeds the number of  $b$ 's, violating the condition that the number of  $a$ 's should be equal to the number of  $b$ 's.

Since we cannot pump the string in a way that maintains the equal number of  $a$ 's and  $b$ 's, we have a contradiction to the pumping lemma. Therefore, our assumption that  $L$  is a regular language must be incorrect.

Hence, we can conclude that the language  $L = \{a^n b^n \mid n \geq 0\}$  is not a regular language.