NON-REGULAR LANGUAGES AND THE PUMPING LEMMA

NON-REGULAR LANGUAGES

- The previous discussions have shown that a regular language can be described by a finite automaton and a regular expression.
- However, not every language is regular.
- Non-regular languages cannot be recognized by any finite automaton nor are there regular expressions that represent them.

NON-REGULAR LANGUAGES

- Example: Is the language $L_1 = \{0^x1^x \mid x > 0\}$ regular?

Language L_1 is the set of all strings that starts with x number of consecutive 0s followed by x number of consecutive 1s.

The total number of consecutive 0s is equal to the total number of consecutive 1s following it.

NON-REGULAR LANGUAGES

- Example: Is the language $L_1 = \{0^x1^x \mid x > 0\}$ regular?

The finite automaton for this language should be able to remember the number of 0s it has received at any point in time.

And then check if it is the same as the number of 1s it will be receiving.

But that would require an infinite number of states since the number of 0s is not limited.

So there is no finite automaton that can recognize L_1 since there are only a limited number of states. So language L_1 is not regular.

NON-REGULAR LANGUAGES

- Another example: Is the language $L_2 = \{w \mid w$ has an equal number of 0s and 1s$\}$ regular?

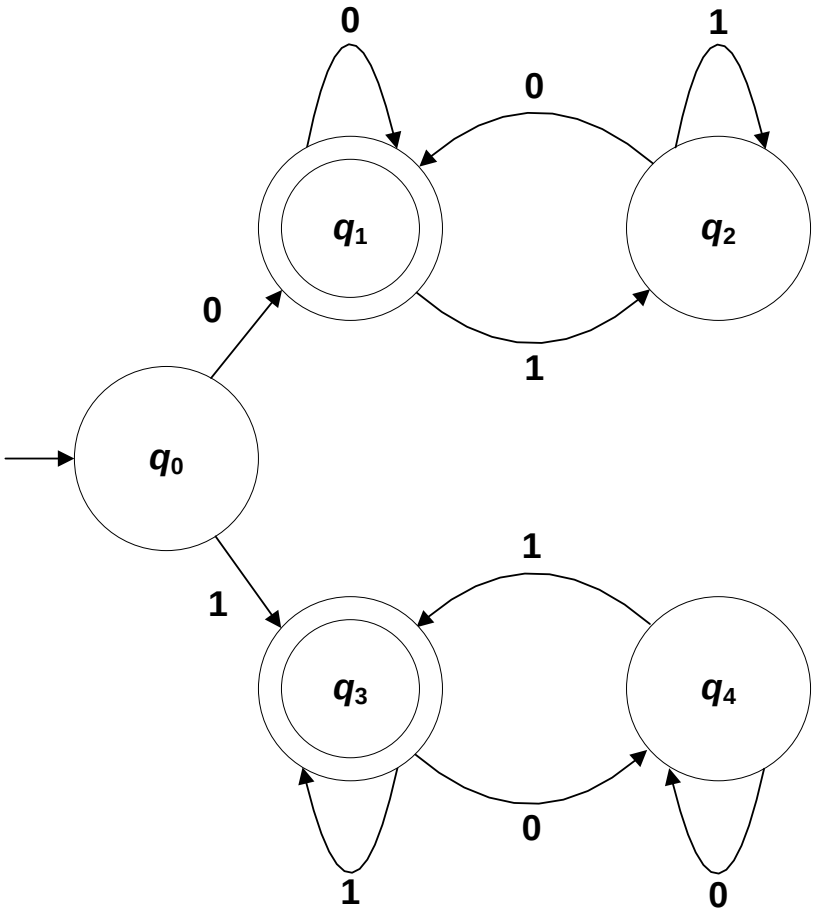
  Like language L_1, L_2 is also not regular since it requires the DFA to remember the number of 0s and 1s it has received.

  A DFA with only a limited number of states cannot accomplish this.

NON-REGULAR LANGUAGES

- However, consider the language $L_3 = \{w \mid w$ has an equal number of occurrences of 01 and 10 as substrings.

  Although it seems there are also unlimited possibilities, L_3 is regular.



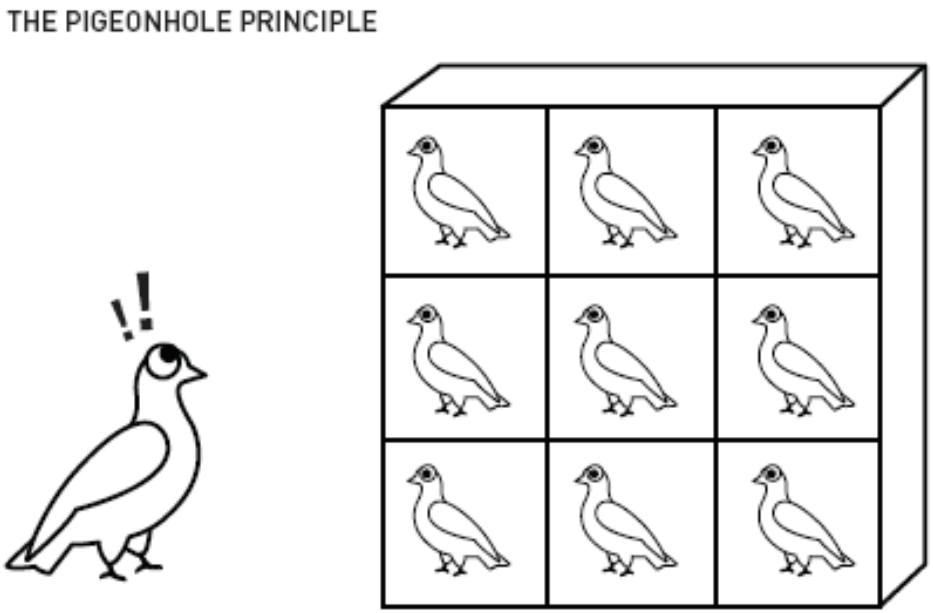
NON-REGULAR LANGUAGES

- It is therefore more difficult to prove the non-regularity of languages than to prove their regularity.

- To prove the non-regularity of languages, it is appropriate to discuss first one special property of regular languages that is based on the ***pigeonhole principle***.

- The pigeonhole principle is a powerful tool used in combinatorial math.

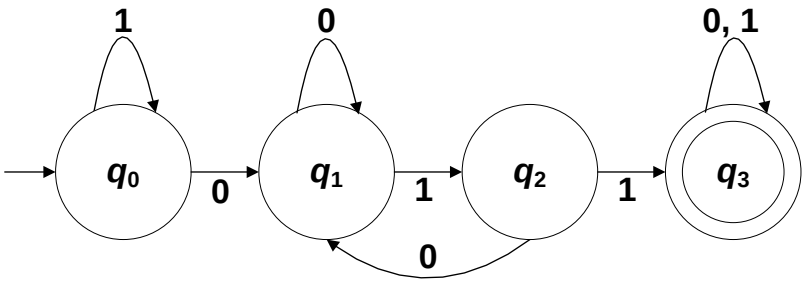
NON-REGULAR LANGUAGES



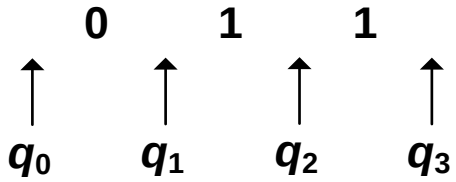
- The pigeonhole principle states that if there are *m* pigeonholes and *n* pigeons where $n > m$, there will be at least one pigeonhole with at least two pigeons

NON-REGULAR LANGUAGES

- Consider a 4-state DFA shown below:

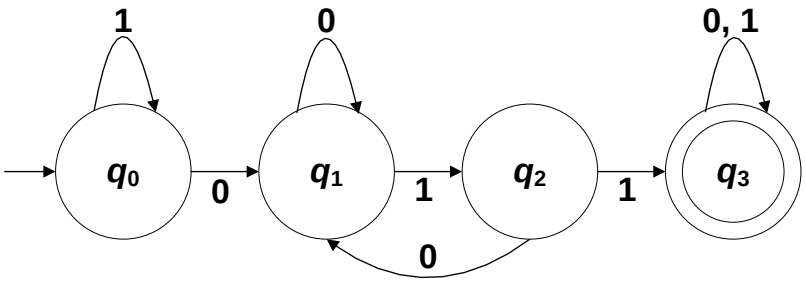


- Consider an input string 011 whose length is 3. The sequence of states that will be traversed during the computation are:

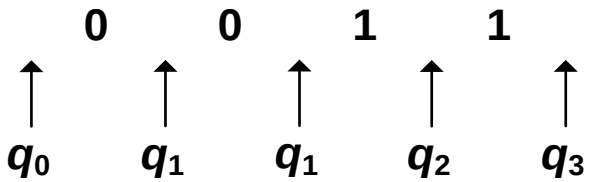


- Take note that four states were traversed and no state was repeated during the computation

NON-REGULAR LANGUAGES

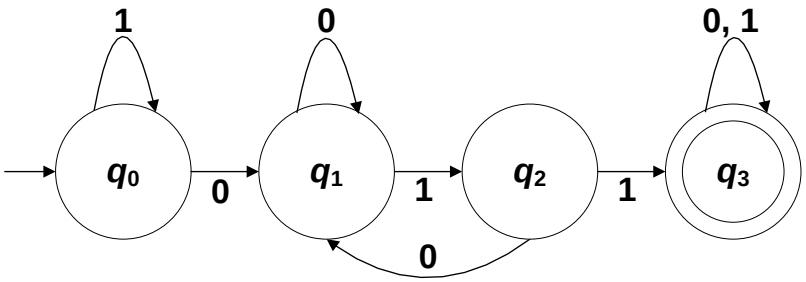


- Now consider an input string 0011 whose length is 4. The sequence of states that will be traversed during the computation are:

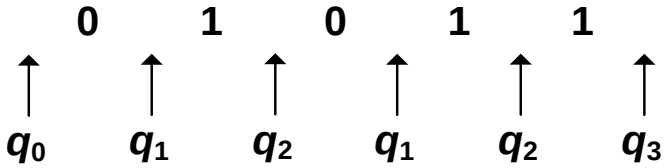


- There are five states that were traversed during the computation of 0011 (q_0, q_1, q_1, q_2, and q_3). However, there are only four states in the DFA. Therefore, there will be at least one state that will be traversed at least twice (repeated state). In this example, state q_1 was repeated.

NON-REGULAR LANGUAGES



- Next consider an input string 01011 whose length is 5. The sequence of states that will be traversed during the computation are:



- There are six states traversed during the computation of the input string 01011 (q_0, q_1, q_2, q_1, q_2, and q_3). But since there are only four states in the DFA, there will be at least one state that will be traversed at least twice. In this case, states q_1 and q_2 were repeated.

NON-REGULAR LANGUAGES

- Always take note that:

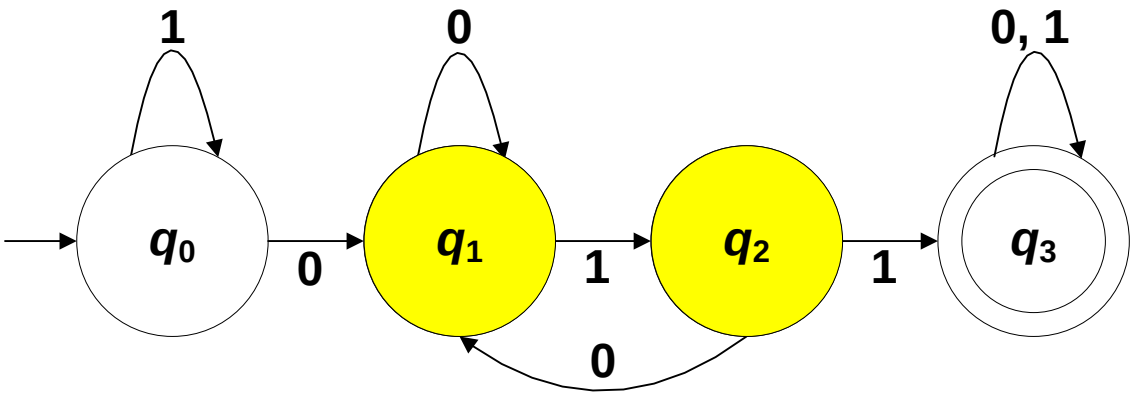
  number of states traversed = |w| + 1

- In general, if the length of the input string is greater than or equal to the number of states of the DFA, there will be more states traversed than there are number of states in the DFA.

  Therefore, there will be at least one state that will be repeated.

- Furthermore, the repeated state or states form a loop.

NON-REGULAR LANGUAGES

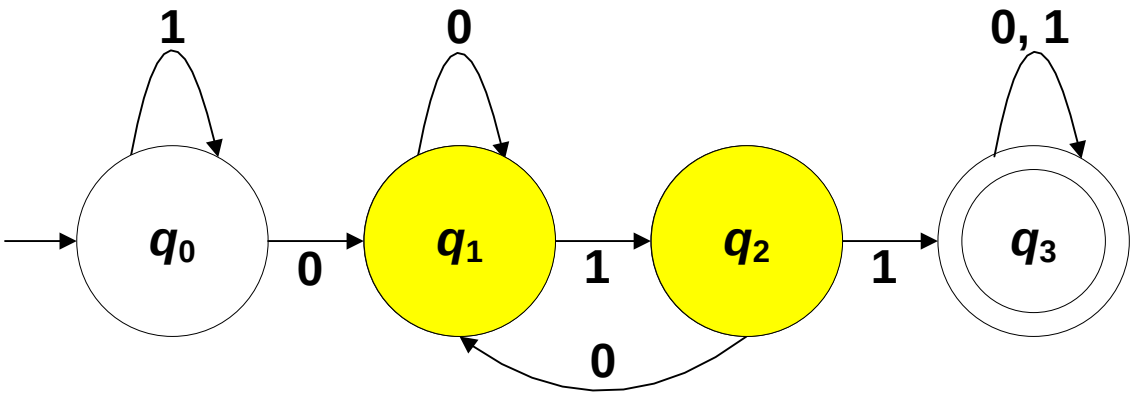


- In the third example, when the input string was 01011, the loop started when the DFA is at state q_2 (after receiving the first two symbols 01).

  Then it moves back to state q_1 when it receives a 0.

  And from state q_1, it goes back to state q_2 when it receives a 1.

NON-REGULAR LANGUAGES



- Looking at the given input string 01**011**, the loop was caused by the second 01 substring.

- Notice now that if the substring 01 that formed the loop is repeated any number of times, the input string is still accepted (it is still a member of the language).

- So the input strings 01**01011** and 01**0101011** are still accepted by the DFA.

NON-REGULAR LANGUAGES

- All regular languages have this property that:
  - If they have strings whose length is equal or greater than the number of states of their DFA,
  - then these strings have a substring that can be repeated an arbitrary number of times, and the resulting strings will still be in language (accepted by the DFA).

01011

01011011

0101011011

010101011011

01010101011011

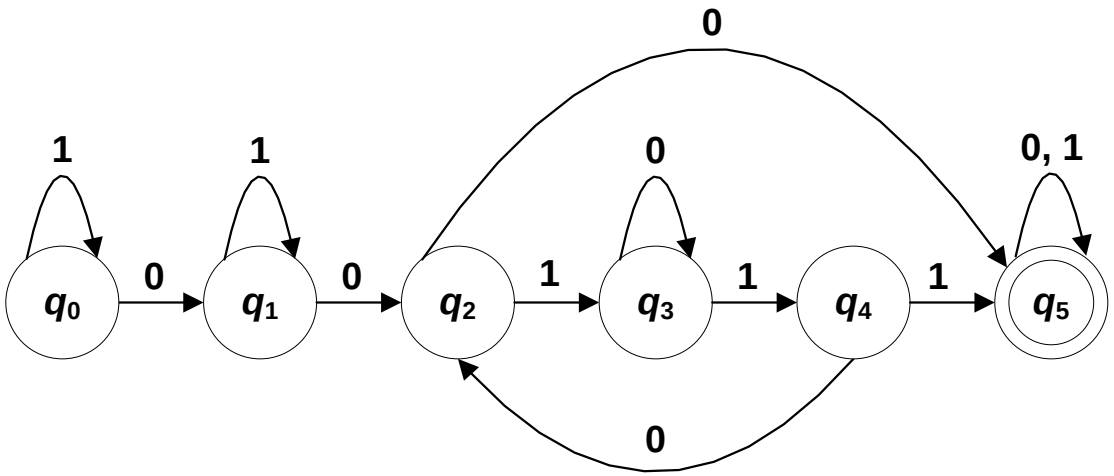
0101010101011011

NON-REGULAR LANGUAGES

- Repeating a part of a string any number of times is called ***pumping*** the string.
- This property of regular languages can be used to determine if a language is non-regular.
- All that has to be done is to show that they do not have this property.
- This property is formalized by the ***pumping lemma***.

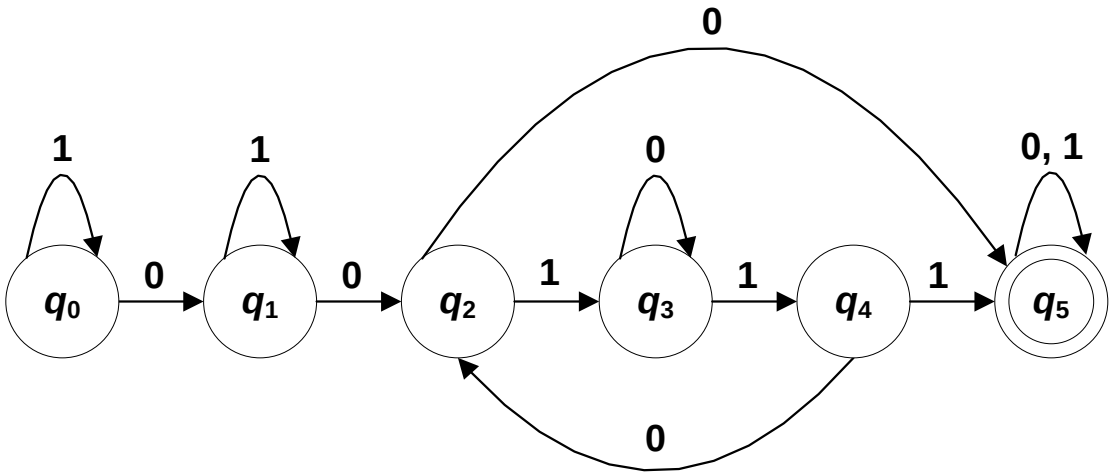
THE PUMPING LEMMA

- Consider the following DFA:

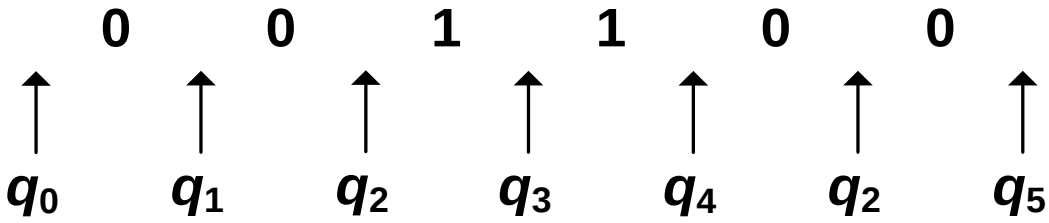


Assume that the input string w = 001100. Let m = 6 be the number of states of the DFA. Since $|w| \geq m$, then at least one state will be traversed at least twice.

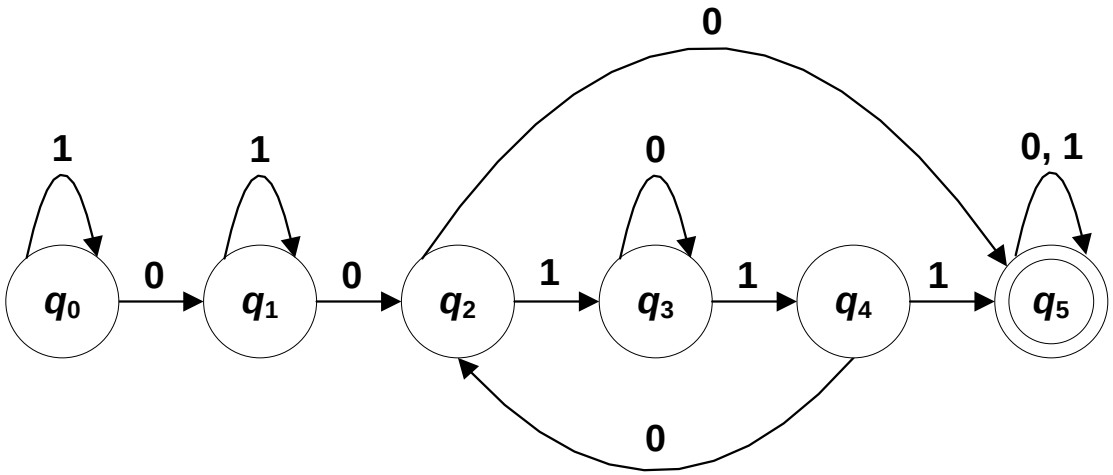
THE PUMPING LEMMA



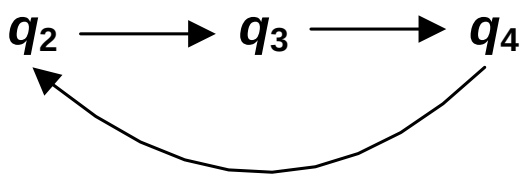
For w = 001100, the states that will be traversed are



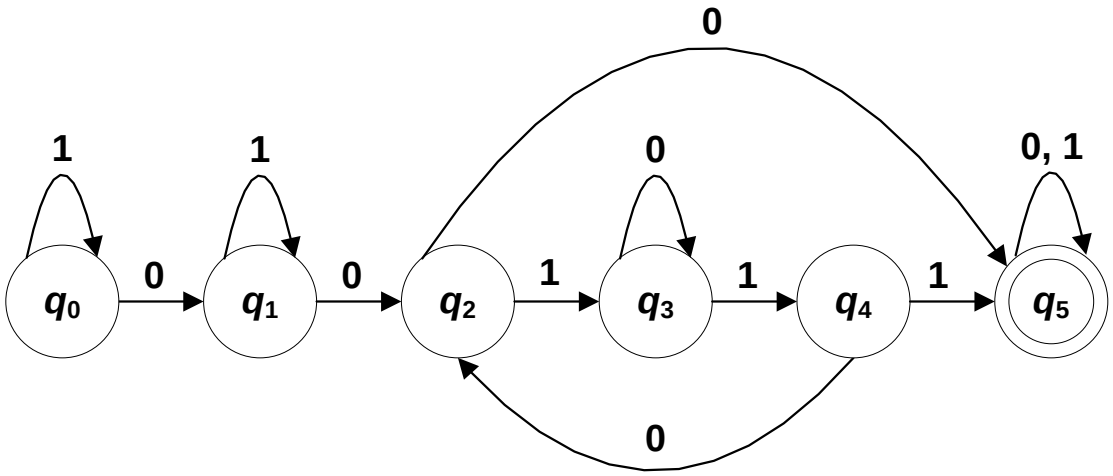
THE PUMPING LEMMA



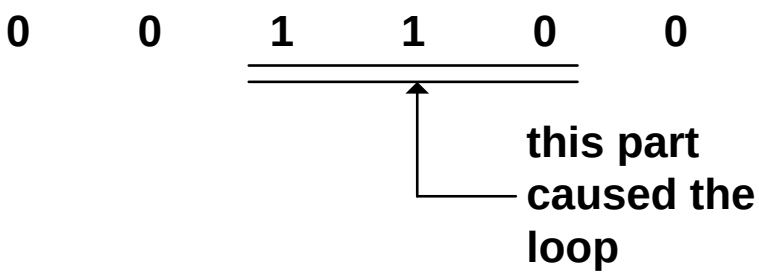
For w = 001100, state q_2 was traversed twice thereby forming a loop or cycle consisting of states q_2, q_3, and q_4. Specifically,



THE PUMPING LEMMA

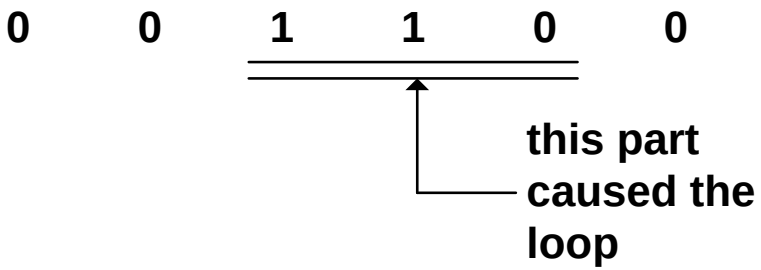


The part of the input string 001100 that caused the loop is the substring 110.



Take note that the first repeated state appeared within the first *m* symbols of the input string.

THE PUMPING LEMMA

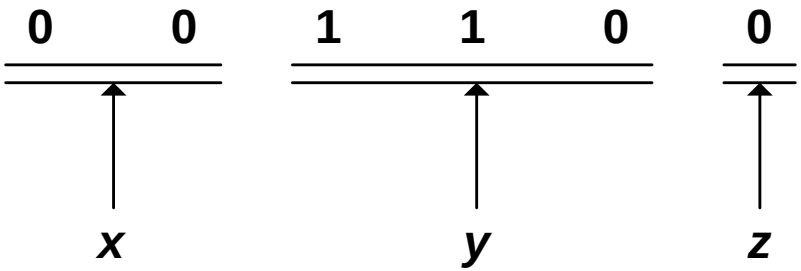


- So for *w* = 001100, the substring 110 can be repeated an arbitrary number of times (pumped) and the resulting strings will still be part of the language.

- These include strings such as 00110110, 00110110110, and 00110110110110.

THE PUMPING LEMMA

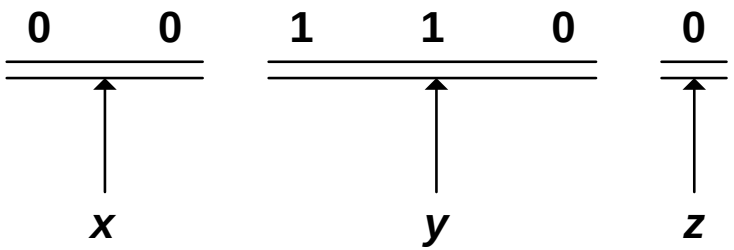
- The input string can actually be viewed as being composed of three parts or substrings (called *x*, *y*, and *z*) as shown below:

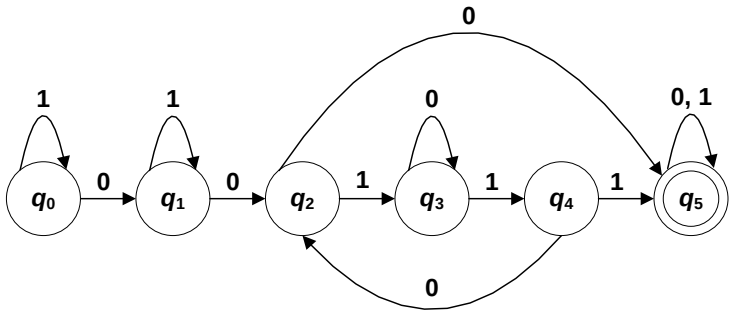


- The input string *w* can therefore be written as:

$$w = xyz$$

THE PUMPING LEMMA



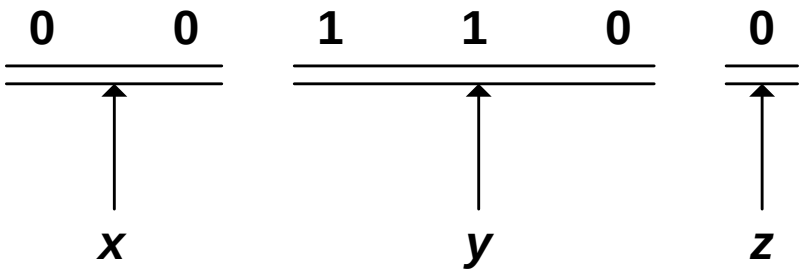


- Substring *x* is the first part of the input string that is composed of the symbols before the 1st appearance of the repeated state which is q_2.

- Substring *y* is the second part of the input string that contains the input symbols from the first occurrence of the repeated state up to its second occurrence. This is the loop within the input string.

- Substring *z* is composed of the remaining symbols of the input string that lead to the final state.

THE PUMPING LEMMA



- Since substring *y* can be repeated an arbitrary number of times and the resulting string is still part of the language, then *w* can be generalized as

$$w = x\, y^i\, z \text{ where } i \geq 0$$

Examples:

$w = xy^1z = 00\mathbf{110}0$
$w = xy^2z = 00\mathbf{1101100}$
$w = xy^3z = 00\mathbf{11011011}0$

THE PUMPING LEMMA

- To summarize this special property of regular languages:

1. Let *w* be any string of regular language *L* whose length is greater than or equal to the number of states of the DFA that recognizes language *L* ($|w| \geq m$).

2. The string *w* can be divided into three parts *xyz* where *y* contains the input symbols from the first occurrence of a repeated state to its second occurrence.

THE PUMPING LEMMA

3. The length of substring *xy* will be less than or equal to the number of states of the DFA ($|xy| \leq m$).

4. The length of $y \geq 1$.

5. Substring *y* can be repeated or pumped any number of times and the resulting string is still a member of language *L*.

- The summary given is an informal statement of the pumping lemma.

FORMAL DEFINITION OF THE PUMPING LEMMA

- If L is a regular language, then there is a number p (the pumping length) where, if w is any string in L whose length is greater than or equal to p, then w may be divided into three parts, $w = xyz$, satisfying the following conditions:

    1. for each $i \geq 0$, $x\, y^i\, z \in L$,
    2. $|y| \geq 1$, and
    3. $|xy| \leq p$.

Take note that the pumping length p refers to the number of states of the DFA that recognizes the given language L. Hence, $|w| \geq p$ in order for the pumping lemma to be used.

FORMAL DEFINITION OF THE PUMPING LEMMA

- Major Points:

1. The string $w = x\, y^i\, z \in L$ for each $i \geq 0$. This states that the loop or cycle within the input string w can be pumped any number of times and w will still be a member of L.

2. $|y| \geq 1$ refers to that fact that y cannot be an empty string. There has to be a loop or cycle that can be pumped.

3. $|xy| \leq p$ refers to the fact that the loop or cycle that will be pumped is the first one encountered from the start of the input string. In other words, the part to be pumped is within the first p symbols of the input string.

USING THE PUMPING LEMMA

- The pumping lemma can be used to prove the non-regularity of languages.

- Pumping lemma proofs often use proof by contradiction:

1. First, assume that the language *L* is regular.
2. Since *L* is regular, the pumping lemma states that all strings of *L* that are long enough can be pumped.
3. Find a string *w* that is long enough that cannot be pumped.
4. If there is such a string, then the pumping lemma was contradicted. Hence, language *L* is not regular.

USING THE PUMPING LEMMA

- Example 1: Prove that the language $L_1 = \{0^x1^x \mid x > 0\}$ is not regular.

  Assume first that L_1 is regular.

  Let p be the pumping length of L_1.

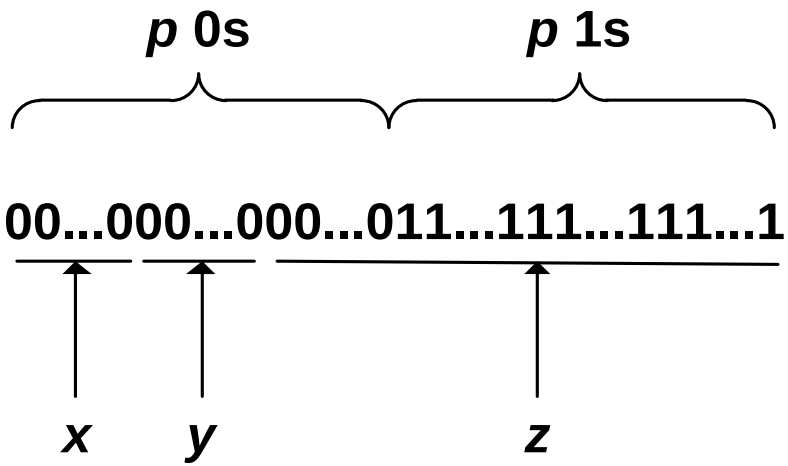
  Let the string to be pumped be $w = 0^p1^p$.

  Since $|w| \geq p$, the string chosen is long enough to be pumped.

  Consider now the various ways in which w can be divided into x, y, and z substrings.

USING THE PUMPING LEMMA

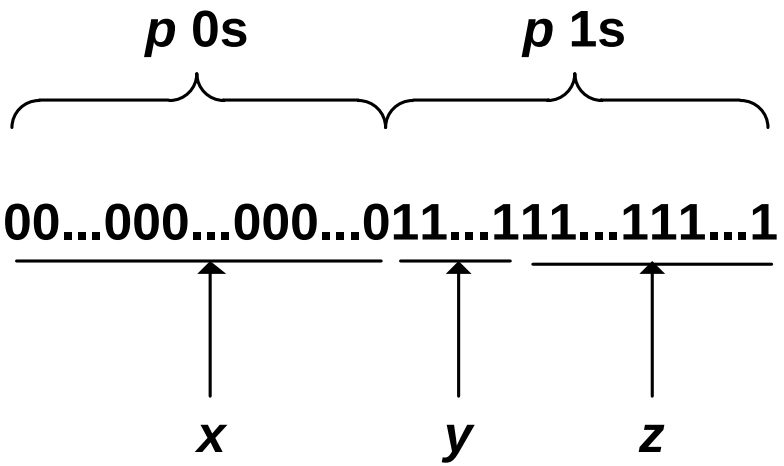
Option 1: Let *y* be all 0s.



If *y* = all 0s, pumping it would create a string in which there are more 0s than 1s. Hence, the resulting string will not be in L_1. This violates condition 1 of the pumping lemma.

USING THE PUMPING LEMMA

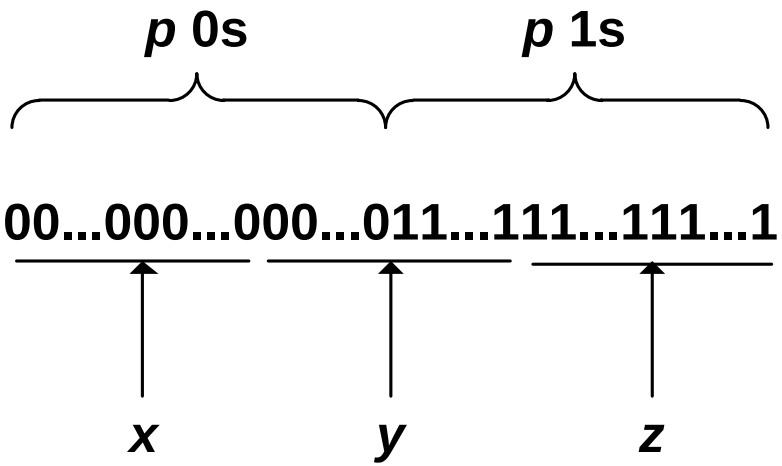
Option 2: Let *y* be all 1s.



If *y* = all 1s, pumping it would create a string in which there are more 1s than 0s. Hence, the resulting string will not be in L_1. This violates condition 1 of the pumping lemma.

USING THE PUMPING LEMMA

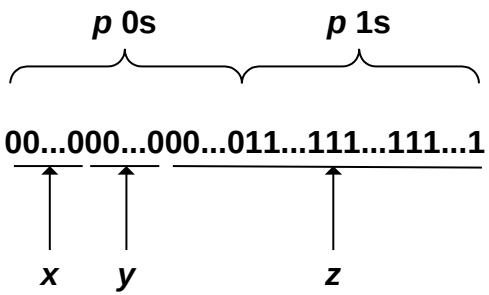
Option 3: Let *y* be composed of 0s and 1s.



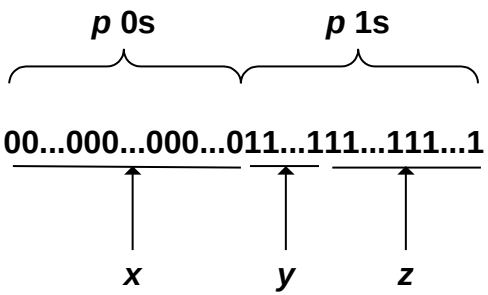
If *y* is composed of 0s and 1s, pumping it would create a string that may have the same number of 0s and 1s, but they will be out of order with some 1s before 0s. Hence, the resulting string will not be in L_1. This violates condition 1 of the pumping lemma.

USING THE PUMPING LEMMA

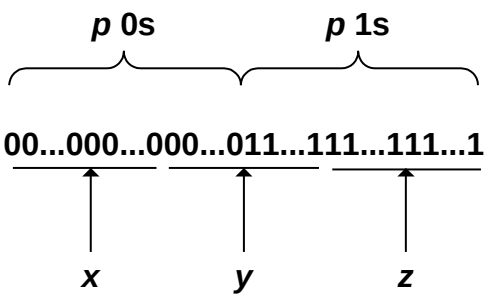
Option 1



Option 2



Option 3



Actually, Options 2 and 3 should not have been checked since they violate condition 3 which states that $|xy| \leq p$.

USING THE PUMPING LEMMA

Conclusion:

Since L_1 has a string (that is long enough) that cannot be pumped, then this contradicts the original assumption that L_1 is regular.

Therefore, L_1 is not regular.

USING THE PUMPING LEMMA

- Example 2: Prove that the language $L_2 = \{ww \mid w \in \{0,1\}^*\}$ is not regular.

  Assume first that L_2 is regular and let p be the pumping length of L_2.

  Let the string to be pumped be $w = 0^p10^p1$. Since $|w| \geq p$, the string can pumped.

  From condition 3, substring y can only be 0s that belong to the first half of the input string. If y is pumped, then there will be more 0s in the first half of the string compared to its second half, thereby creating an imbalance. This contradicts the original assumption that L_2 is regular. Therefore, L_2 is not regular.

USING THE PUMPING LEMMA

- Example 3: Prove that the language $L_3 = \{w \in \{0,1\}^* \mid w = w^R\}$ is not regular.

  Assume first that L_3 is regular and let p be the pumping length of L_3.

  Let the string to be pumped be $w = 0^p10^p$. Since $|w| \geq p$, the string can pumped.

  From condition 3, the substring y can only be 0s that belong to the first block of consecutive 0s (to the left of the single 1) of the input string. If y is pumped, then the string is no longer a palindrome. This contradicts the original assumption that L_3 is regular. Therefore, L_3 is not regular.

USING THE PUMPING LEMMA

- Example 4: Prove that the language $L_4 = \{0^i1^j \mid i > j\}$ is not regular.

  Assume first that L_4 is regular and let p be the pumping length of L_4.

  Let the string to be pumped be $w = 0^{p+1}1^p$. Since $|w| \geq p$, the string chosen is long enough to be pumped.

  From condition 3, substring y can only be 0s that belong to the block of consecutive 0s of the input string. If y is pumped, then there will still be more 0s than 1s. Hence, the string is still in language L_4.

  Is the language regular?

USING THE PUMPING LEMMA

Condition 1 of the pumping lemma states that $w = xy^iz \leq L$ for each $i \geq 0$. This means that i can be equal to 0 making $w = xy^iz = xy^0z = xz$. This is called ***pumping down***.

If y is pumped down, then there will be more consecutive 1s than 0s. Hence, the resulting string is not in L_4.

Since L_4 has a string that cannot be pumped, then this contradicts the original assumption that L_4 is regular. Therefore, L_4 is not regular.

EXERCISES

- Use the pumping lemma to prove that the language $L = \{a^n b^m c^{n+m}\}$, where n and $m \geq 0$ is nonregular.

  Assume the alphabet $\Sigma = \{a, b, c\}$. Show all possible cases.