12 Pumping Lemma

'Pumping the energy into the system regularises the activites, accepts the challenges, extends the benefits.'

Introduction

In this chapter, the concept of Pumping Lemma and its applications are discussed in detail. This concept is an important one, which introduces certain novel features into the subject. The study and analysis is based on the following discussion:

Suppose an automaton, over an alphabet Σ, has k states and suppose $w = a_1, a_2, a_3, \ldots, a_n$ is a word over Σ accepted by M, such that, $|w| = n > k$. Let $p = (s_0, s_1, \ldots, s_n)$ be the corresponding sequence of states determined by the word w. The condition that $n > k$ suggests that two of the states in p must be equal, say $s_i = s_j$ where $i < j$. Further, let

$$x = a_1, a_2 \cdots a_i, \quad y = a_{i+1} \cdots a_j, \quad z = a_{j+1} \cdots a_n.$$

Now, clearly from the figure 12.1, xy ends in $s_i = s_j$; hence xy^m also ends in s_i. In other words, for every m, $w_m = xy^m z$ ends in s_n, which is clearly an accepting state.

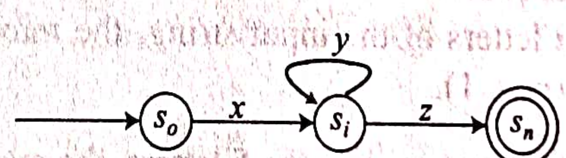

Figure 12.1.

The above discussion leads to the following important result, the details of which are discussed in the subsequent sections.

Suppose M is an automaton over Σ, such that:

a. M has k states
b. M accepts a word w from Σ where $|w| > k$.

Then $w = xyz$ where, for every positive m, $w_m = xy^m z$ is accepted by M.

12.1 Introduction to Non-Regular Languages

Consider a finite automata having 5 states, as shown below:

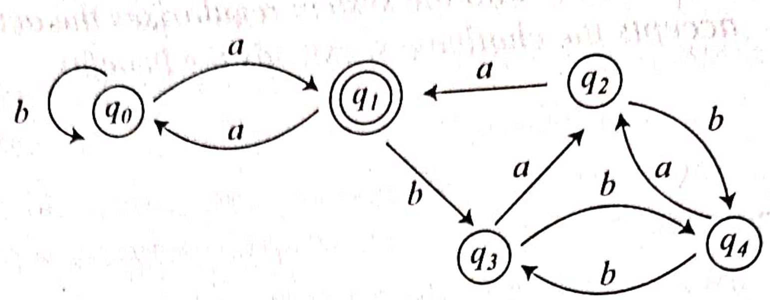

Figure 12.2. *A Finite Automaton with 5 States*

Let us process the string *ababbaa* on the *FA*:

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_3 \xrightarrow{a} q_2 \xrightarrow{b} q_4 \xrightarrow{b} q_3 \xrightarrow{a} q_2 \xrightarrow{a} q_1.$$

Since q_1 is the final state, the string *ababbaa* is accepted. In general,

a. we always start from the initial state.
b. after reading the first letter of the input string
   i. we may go to another state or return to the initial state,
   ii. the maximum number of different states, that can be visited after reading the first letter, is 2.
c. after reading the first two letters of the input string, the maximum number of different states that can be visited, is 3.
d. after reading the first m letters of the input string, the maximum number of different states visited would be $(m+1)$.

Thus, with the string *ababbaa*, after reading the 5 letters, the maximum number of different states that is visited is $5+1=6$. However, since *FA* has only 5 states, this means after reading the 5 letters, there is a state that is visited twice.

Consider the string *aaabaa*:

a. The string length is 6, which is more than the number of states in the above *FA*.
b. Let us process the string *aaabaa* on *FA*:

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_3 \xrightarrow{a} q_2 \xrightarrow{a} q_1$$

Since q_1 is the final state, so the string is accepted.
c. Here, the state q_0 is visited twice.

Thus, in general, for a FA with N states, with a string w of length $|w| \geq N$, there exists atleast one state that is visited atleast twice.

(i) Let u be the state that is visited twice.
(ii) Break up the string w as $w = xyz$ where x, y and z are 3 strings such that

- string x has those letters that are at the beginning of w and are read by the FA until the state u is hit for the first time.
- string y has those letters used by FA, starting from the time we are in the state u, to the time the state u is hit for the second time.
- string z contains the rest of the letters in w.

For the string $w = ababbaa$ processed on the above FA,

$$u = q_1, \quad x = ab, \quad y = abb \text{ and } z = aa.$$

For the string $w = aaabaa$ processed on the above FA,

$$u = q_0, \quad x = \in, \quad y = aa, \quad z = abaa.$$

Now, consider a language $L = \{\epsilon, ab, aabb, aaabbb, \ldots,\}$ i.e.,

$$L = \{a^n b^n : n = 0, 1, 2, \ldots\}$$

which is regular.

The following are the observations made:

a. Consider the FA (figure 12.3) with 5 finite states and $w = a^6 b^6$.
b. The first 6 letters of the word are a's. While processing these six letters, the FA visits some state u atleast twice, since there are only 5 states in the FA.

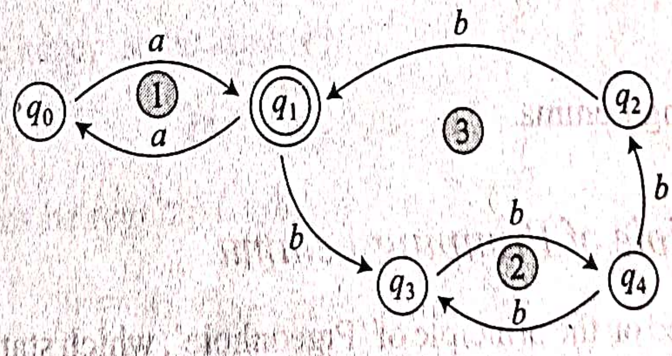

Figure 12.3. A Finite Automaton with 5 States

c. We can say that the path has a circuit ① ② and ③, which consists of edges that are taken from the time u is visited for the first time to the time corresponding to the next visit to u.
d. After the first b is read, the path goes elsewhere and eventually ends up in the final state, where $w = a^6 b^6$ is accepted.
e. Now consider the string $w = a^{6+3} b^6$. While processing this string, we again end up in the final state and hence $w = a^{6+3} b^6$ is also accepted.

f. But $a^{6+3}b^6$ is not in $L(a^6(a^3)^k b^6, k \geq 0)$, since it does not have an equal number of a's and b's.

g. Thus L is not a regular language, which means that it is a non-regular language.

Definition:

a. A language that cannot be defined by a regular expression is called a non-regular language.

b. The languages which are not regular are called non-regular languages.

12.1.1 *Examples of Non-Regular Languages*

a. $L = \{a^n b^n : n \geq 0\}$

b. $L = \{ww^R : w \in \Sigma^*\}$

c. $L = \{a^n b^l c^{n+l} : n, l \geq 0\}$

d. $L = \{a^{n!} : n \geq 0\}$

e. $L = \{a^i : i^r\}$

f. $L = \{O^{n^2} : n \geq 0\}$

g. $L = \{O^i 1^i 2^k : O \leq i < j < k\}$

12.1.2 *Pumping Lemma*

The fundamental tool for proving that a language is

- not regular
- not context-free

is known as the Pumping Lemma.

12.1.3 *The Principle of Pumping Lemma*

Pumping Lemma is based on the principle of 'Pigeonhole', which states that if '*n*' pigeons are placed in '*m*' holes, and if $n > m$, then atleast one hole must have more than one pigeon in it.

12.2 Pumping Lemma for Regular Languages

Pumping Lemma for regular languages is used to recognise all non-regular languages. It gives a necessary condition for an input string to belong to a regular set and also states a method of pumping (generating) many input strings from a given string, such that all of them are in the language, if the language is regular.

Pumping Lemma connot be used to establish that a given language is regular, but it can be used to prove that a language is not regular by showing that the language does not obey the lemma.

12.2.1 Pigeonhole Principle

Pigeonhole Principle of Lemma for regular languages states that 'in a transition diagram with n states, any string of length greater than or equal to n must repeat some state'.

Theorem I: Pumping Lemma for RL's

If L is a regualr set, accepted by some finite automaton D (with n number of states), with a string w in L written as $w = xyz$, then

a. $|y| \geqslant 1$
b. $|xy| \leqslant n$
c. $xy^iz \in L\ \forall i \geq 0$, where y^i denotes y repeated i times and $y^0 = \epsilon$.

In other words, any sufficiently long string accepted by a finite automaton, can be broken into three parts (x, y and z) in such a way that an arbitrary number of repetitions of the middle part (y) yields another string in L. In that case, we say that the middle substring is pumped and hence the name, Pumping Lemma.

Proof. Let a language accepted by the DFA

$$D = (Q, \Sigma, \delta, q_0, F)$$

with n number of states be regular. Consider an input string w of length m, with $m \geqslant n$:

$$w = a_1, a_2, \ldots, a_m, \quad \text{where } m \geqslant n.$$

Let

$$\delta(q_0, a_1, a_2, \ldots a_i) = q_i.$$

Then it is not possible for each of the $n + 1$ states $q_0, q_1, \ldots, q_n$ to be distinct, since there are only n different states. This means that there must be atleast two states in Q, which must coincide.

Thus, there are two integers j and k with $0 \leq j < k \leq n$, such that

$$q_j = q_k.$$

method of pumping (generating) many input strings from a given string, such that all of them are in the language, if the language is regular.

Pumping Lemma cannot be used to establish that a given language is regular, but it can be used to prove that a language is not regular by showing that the language does not obey the lemma.

12.2.1 *Pigeonhole Principle*

Pigeonhole Principle of Lemma for regular languages states that 'in a transition diagram with n states, any string of length greater than or equal to n must repeat some state'.

Theorem I: Pumping Lemma for RL's

If L is a regualr set, accepted by some finite automaton D (with n number of states), with a string w in L written as $w = xyz$, then

a. $|y| \geqslant 1$
b. $|xy| \leqslant n$
c. $xy^iz \in L \; \forall i \geq 0$, where y^i denotes y repeated i times and $y^0 = \epsilon$.

In other words, any sufficiently long string accepted by a finite automaton, can be broken into three parts (x, y and z) in such a way that an arbitrary number of repetitions of the middle part (y) yields another string in L. In that case, we say that the middle substring is pumped and hence the name, Pumping Lemma.

Proof. Let a language accepted by the DFA

$$D = (Q, \Sigma, \delta, q_0, F)$$

with n number of states be regular. Consider an input string w of length m, with $m \geqslant n$:

$$w = a_1, a_2, \ldots, a_m, \quad \text{where } m \geqslant n.$$

Let

$$\delta(q_0, a_1, a_2, \ldots a_i) = q_i.$$

Then it is not possible for each of the $n+1$ states $q_0, q_1, \ldots q_n$ to be distinct, since there are only n different states. This means that there must be atleast two states in Q, which must coincide.

Thus, there are two integers j and k with $0 \leq j < k \leq n$, such that

$$q_j = q_k.$$

The string w can now be written as

$$w = a_1, a_2, \ldots a_j, \quad a_{j+1}, a_{j+2}, \ldots a_k, \quad a_{k+1}, a_{k+2}, \ldots a_m.$$

Thus $w = xyz$,

$$\text{where } x = a_1, a_2, \ldots a_j$$
$$y = a_{j+1}, a_{j+2}, \ldots a_k$$
$$z = a_{k+1}, a_{k+2}, \ldots a_m.$$

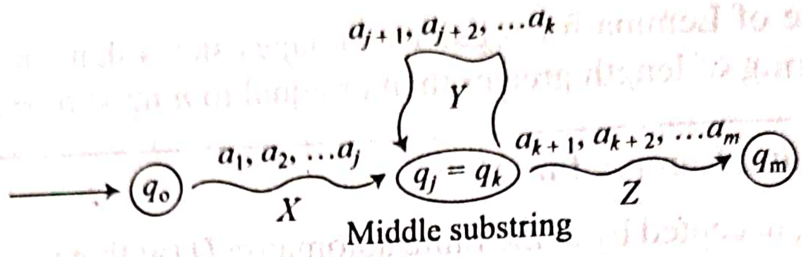

Figure 12.4. *Path in the Transitions for the DFA, D.*

Since there is a path from q_0 to q_m that goes through q_j, but not around the loop labelled $a_{j+1}, \ldots a_k$, the input string

$$a_1, a_2, \ldots a_j, a_{k+1}, a_{k+2}, \ldots a_m \text{ is in } L(D).$$

Consider

$$\begin{aligned}\delta(q_0, a_1, a_2, \ldots a_j, a_{k+1}, a_{k+2}, \ldots a_m) &= \delta(\delta(q_0, a_1, \ldots a_j), a_{k+1}, \ldots a_m)\\ &= \delta(a_j, a_{k+1}, \ldots a_m)\\ &= q_m\\ &\Rightarrow xyz \in L(D).\end{aligned}$$

The automaton starts from the initial state q_0, and with the string x, it reaches q_j. Then with the string y, it comes back to q_j again and finally with the string y^i, the automaton will be in the same state q_j

$$\text{i.e.} \quad \delta(q_0, xy^2) = q_j$$
$$\delta(q_0, xy^3) = q_j$$
$$\ldots$$
$$\delta(q_0, xy^i) = q_j.$$

By introducing the string z, the automaton reaches the final state q_m, i.e.,

$$xy^i z \in L(D).$$

Hence proved.