AUTOMATA FORMAL LANGUAGES AND LOGIC

Lecture notes on Pumping Lemma

Prepared by:
Prof.Divya S J
Assistant Professor

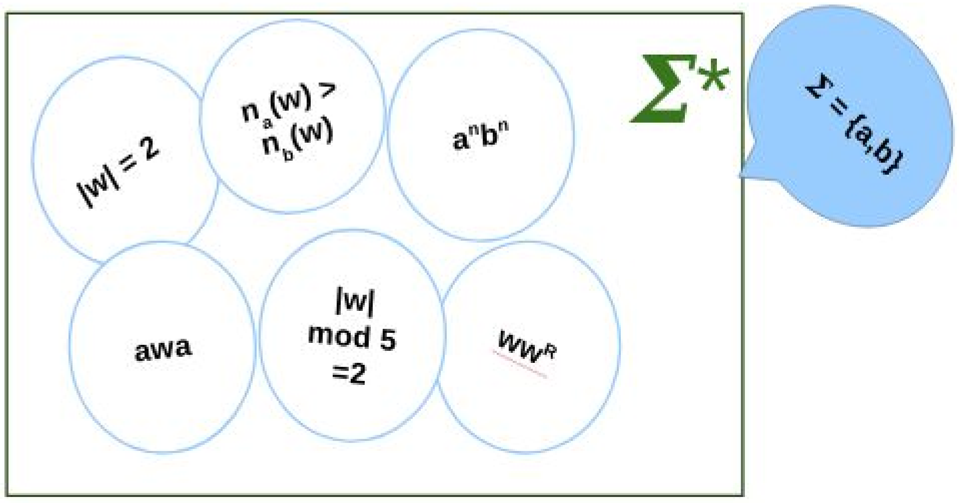
Department of Computer Science & Engineering

PES UNIVERSITY
(Established under Karnataka Act No.16 of 2013)
100-ft Ring Road, BSK III Stage, Bangalore – 560 085

Table of contents

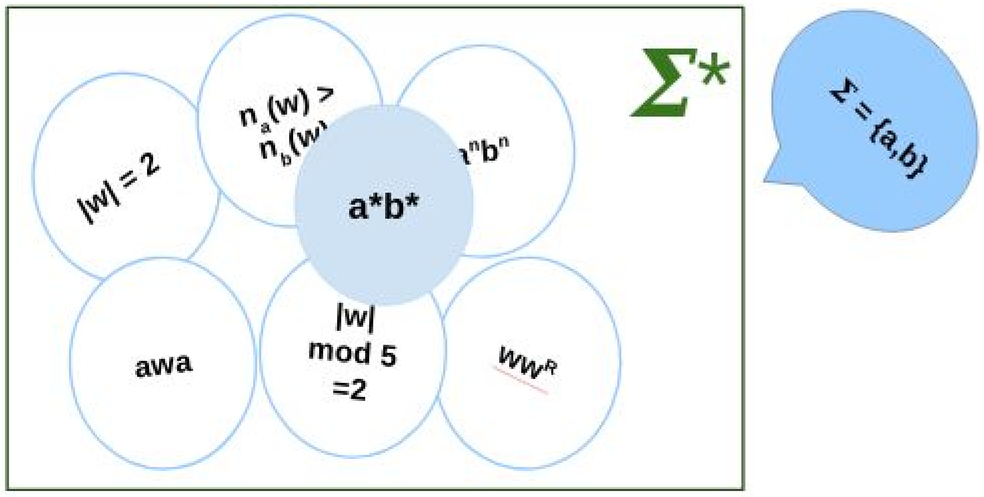
Pumping Lemma

We know that we can define a set of languages over the alphabet a,b. Each of these languages will be a subset of sigma *.Few of the languages are superset or subset of other languages. We see that in this set we have few languages which are finite for example length of string <=2 or exactly equal to 2.We also have infinite languages, for example language where the no of a’s >=2 or where the no of a’s is equal to no. of b’s in the string.



The question is can we construct a finite automata for each of the possible infinite languages. Alternatively we can say is there any language which is infinite but not regular.

Consider this example of the language expressed by the regex a*b*.



We can easily construct a finite automata for this, and also write regular grammar.Since we are able to construct a finite automata, regex and regular grammar for such a language, this language although infinite is definitely regular.

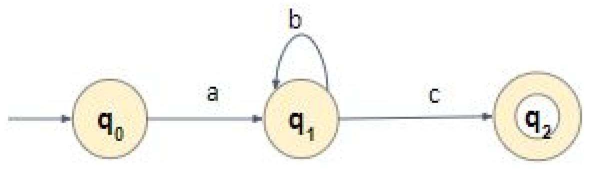
Consider a subset of this language anbn that is followed by equal no. of b's. Is it possible to construct a Finite Automata or write a regular grammar or a regex for this language?? Intuitively, in order to accept this set , an automaton scanning a string of the form a*b* would have to remember when passing the center point between the a'S and b's how many a's it has seen, since it would have to compare that with the number of b's and accept iff the two numbers are the same. Moreover, it would have to do this for arbitrarily long strings of a's and b's, much longer than the number of states. This is an unbounded amount of information, and there is no way it can remember this with only finite memory. All it "knows" at that point is represented in the state q it is in, which is only a finite amount of information. You might at first think there may be some clever strategy, such as counting mod 3, 5, and 7, or something similar. But any such attempt is doomed to failure: you cannot distinguish between infinitely many different cases with only finitely many states.

Basically,There is no way to remember how many a's you have seen to compare with the upcoming b's !The value of n could be anything! We cannot come up with a FA that takes care of all n!

Therefore this language a^nb^n is not regular suggesting that the finite automata has limits.

Finite automaton has a finite number of states. Hence, a finite automata can only "count" a finite number of input scenarios.(that is, maintain a counter, where different states correspond to different values of the counter) .The finite automata have only string pattern recognizing power. It can only recognize linear powers for example a^n where $n \geq 0$ is possible but $a^{2^\wedge n}$ is not or a^i where i is prime is not , $a^{n!}$ is not .

So a language which is finite is definitely regular, we can always construct an acceptor or a regex or a regular grammar for a finite language.However if the language is infinite, it may or may not be regular.In order to understand what this pumping property is Let's take an example of an infinite language ab*c we have constructed a finite automata for this.



Let's traverse this automata on the string abbbc. We start with the start state q0 which on seeing a will move to q1, q1 will consume all the b's and on seeing a c will move to the state q2 which is a final state. Since the string ends in a final state we accept the string.Here we have taken a string whose length was greater than the no of states in the machine. Traversing such a string through the automata is possible only when we are visiting some state(s) more than once.

This is kind of similar to the pigeonhole principle, which states that if the number of pigeons are greater than the number of pigeon holes then there exists at least one pigeonhole with more than one pigeon in it.

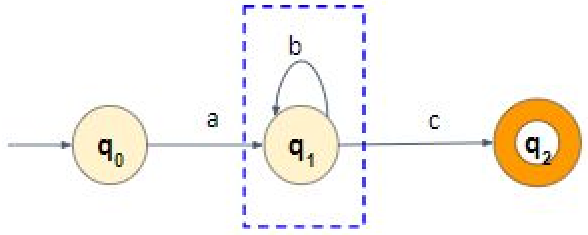
So here we can imagine that the pigeons are nothing but the symbols in the string and pigeonholes are the states in the automata.

So if the length of the string is greater than the number of states in the automata this suggests that at least one of the state was visited more than once.

Visiting a state more than once indicates there exists a loop in our Automata (within these n states) So if we pump that loop 0 or more no. of times, the resultant string will always be in the language

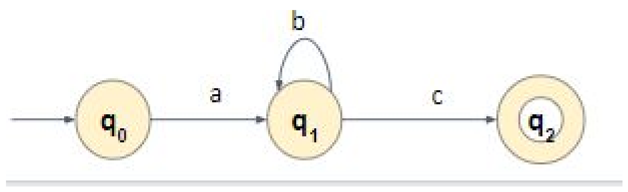
This indicated there exist three parts to a string w which is xyz, x is the part of the string matched before the loop and z is the part matched after the loop and y is the loop.

Let's take an example of infinite regular language ab*c



If |w| >= n we visit a set of states more than once which means,there exists a loop in our Automata (within these n states).If we pump that loop 0 or more no. of times,the resultant string will always be in the language. There exists 3 parts to a string w:w = x y z where x is before the loop, y is within the loop and z is after the loop.y $\neq \epsilon$ that is|y| >=1
The Pumping property States,
For every Regular language L, (infinite) there exists n where n is the # states in Finite Automata for L .For every string w that belongs to L such that, |w| >= n.There exists a break up of the string in three parts w = xyz such that |y| >=1 and |xy|<=n,for every i >= 0,xy^iz belongs to L.

In our example of infinite regular language ab*c



n =3
w = abbbc
|w| = 5 > n
w = abc
x = a
y = b
z = c
for i>=0,
ab^ic is in lang ab*c
To prove language is not regular we will take a negation of pumping property i.e.,
~ ∀ = ∃
~ ∃ = ∀
∀ Regular language L,
    ∃ n where n is the # states in Finite Automata for L
        ∀ string w ∈ L such that,|w| >= n
            ∃ w = xyz such that |y| >= 1 and |xy|<=n,
                ∀ i >= 0, xy^iz ∈ L

∃ a language L which is claimed to be regular,
    ∀ n where n is the # states in Finite Automata for L
        ∃ string w ∈ L such that,|w| >= n
            ∀ w = xyz such that |y| >= 1 and |xy|<=n,
                ∃ i >= 0,xy^iz ∉ L
This contradicts the claim made, hence proving that the language is not regular.This is called Proof by contradiction.
Procedure to prove a language is Not regular :

- Assume the opposite: L is regular
- Use Pumping Lemma to obtain a contradiction
  It suffices to show that only one *string* gives a contradiction
- There by proving L is not regular

For regular languages pumping property always passes, but for non regular languages pumping property may fail or pass.If it fails indicates the language is definitely non regular.If it pass indicates the language may or may not be regular. Therefore pumping lemma is a necessary but not sufficient condition for a language to be regular.

Pumping Lemma as game

The correct argument can be visualized as a game we play against an opponent. Our goal is to win the game by establishing a contradiction of the pumping lemma, while the opponent tries to foil us. There are four moves in the game.

1. The opponent picks n.

2. Given m, we pick a string w in L of length equal or greater than n. We are free to choose any w, subject to $w \in L$ and $|w| \geq n$.

3. The opponent chooses the decomposition xyz, subject to $|xy| \leq n$, $|y| \geq 1$. We have to assume that the opponent makes the choice that will make it hardest for us to win the game.

4. We try to pick i in such a way that the pumped string w ,the equation $w = xy^iz$ in L. If we can do so, we win the game.

A strategy that allows us to win whatever the opponent's choices is tantamount to a proof that the language is not regular. In this, Step 2 is crucial. While we cannot force the opponent to pick a particular decomposition of w, we may be able to choose w so that the opponent is very restricted in Step 3, forcing a choice of x, y, and z that allows us to produce a violation of the pumping lemma on our next move.

Examples

1.Using pumping lemma proves that the language a^nb^n is not regular.

Proof by contradiction:

– Suppose it is regular, then there must be an associated k such that $|xy| \leq n$

– Pick any string in L: let's pick $w = a^n b^n$

– Since $|xy| \leq n$, both x and y must consist of only a's

– Pump y up by choosing i = 2

• Then xyyz should be in L, but it is not because this string has more a's than b's

• For example, if w = aaabbb and i = 2, then aaaabbb should also be in L, but it is not because it has more a's than b's

What if we made a different choice for w?

– We can still prove that $L = \{a^n b^n : n > 0\}$ is not regular

– If L were regular, then there would exists some k such that any string w, where |w| n, must satisfy the conditions of the lemma

– Let $w = a^{n/2} b^{n/2}$

– Since $|w| \geq n$ and w is in L, w must satisfy the conditions of the pumping lemma:

• There must exist an x, y, and z such that $w = xyz$,$|xy| \leq n$, $y \neq \varepsilon$, and $\forall i \geq 0$ ($xy^i z$ is in L)

• We now show that no such y can exist.Divide w into two regions:

```
        aaaaa.....aaaaaa | bbbbb.....bbbbbb
                1        |      2
```

There are 3 places where y could occur – in the a region, in the b region, or across the boundary between the a's and the b's.

• Case 1: $y = a^p$ for some $p \geq 1$. The resulting string is $a^{n/2+p} b^{n/2}$ .This string is not in L, since it has more a's than b's.

• Case 2: $y = b^p$ for some $p \geq 1$. The resulting string is $a^{n/2} b^{n/2+p}$ .This string is not in L, since it has more b's than a's.

• Case 3: $y = (ab)^p$ for some $p \geq 1$. The resulting string is $a^{n/2} (ab)^p b^{n/2}$ has interleaved a's and b's, and so is not in L.

Therefore since there exists at least one long string in L for which there is no way to split w into xyz, such that the required properties are preserved, L is not regular.

2.Using pumping lemma proves that the language palindrome is not regular.

If L were regular, then there would exist some n such that any string w, where $|w| \geq n$, must satisfy the conditions of the pumping lemma

• Let $w = a^n b^n b^n a^n$

• Since $|w| = 4n$ and w is in L, w must satisfy the conditions of the pumping lemma

– There must exist an x, y, and z such that $w = xyz$, $|xy| \leq n$, $y \neq \varepsilon$, and $\forall i \geq 0$ ($xy^i z$ is in L)

– We now show that no such x, y, and z exist

• Since $|xy| \leq n$, y must occur within the first n characters and so $y = a^p$ for some p

• Since $y \neq \varepsilon$, p must be greater than 0

• Let i = 2 (in other words, pump in one extra copy of y)

• The resulting string is $a^{n+p} b^n b^n a^n$

• If p is odd, then this string is not in L because all strings in L have even length

• If p is even, then it is at least 2, so the first half of the string has more a's than the second half so it is not in L

• Therefore, L is not regular

3.L = {ww, w ∈ {a,b}*}

If L were regular, then there would exist some n such that any string w, where $|w| \geq n$, must satisfy the conditions of the pumping lemma

• Let $w = a^n b^n a^n b^n$

• Since $|w| = 4n$ and w is in L, w must satisfy the conditions of the pumping lemma

– There must exist an x, y, and z such that $w = xyz$, $|xy| \leq n$, $y \neq \varepsilon$, and $\forall i \geq 0$ ($xy^i z$ is in L)

– We now show that no such x, y, and z exist

• Since $|xy| \leq n$, y must occur within the first n characters and so $y = a^p$ for some p

• Since $y \neq \varepsilon$, p must be greater than 0

• Let i = 2 (in other words, pump in one extra copy of y)

- The resulting string is $a^{n+p} b^n a^n b^n$ must also be L in but it is not of the right form since the middle of the string would be in the middle of the b which prevents a match with the beginning of the string.
- Therefore, L is not regular

4.L = {0^n where is power of 2}

If L were regular, then there would exist some n such that any string w, where $|w| \geq n$, must satisfy the conditions of the pumping lemma

- Let $w = 0^{2^n}$
- Since $|w| = 2^n$ and w is in L, w must satisfy the conditions of the pumping lemma

– There must exist an x, y, and z such that w = xyz, $|xy| \leq n$, $y \neq \varepsilon$, and $\forall i \geq 0$ ($xy^i z$ is in L)

– We now show that no such x, y, and z exist

- Now consider $0^l 0^p 0^p 0^{n2 -l+p}$ (pumping twice) and show that it is not belong to the language
- Therefore, L is not regular

5.**L = {a^n where n is a prime number}**

If L were regular, then there would exist some k such that any string w, where $|w| \geq n$, must satisfy the conditions of the pumping lemma

- Let $w = a^j$, where j is the smallest prime number greater than n + 1 (e.g., w = aaaaa, n = 3, j = 5)
- Since $|w| > n$ and w is in L, w must satisfy the conditions of the pumping lemma

– There must exist an x, y, and z such that w = xyz, $|xy| \leq k$, $y \neq \varepsilon$, and $\forall i \geq 0$ (xy i z is in L)

– We now show that no such x, y, and z exist

- Since $|xy| \leq n$, y must occur within the first n characters and so $y = a^p$ for some p
- Recall that pumping lemma requires that $\forall i \geq 0$ ($xy^i z$ is in L), therefore $\forall i \geq 0$ ($a^{|x| + i \cdot |y| + |z|}$ must also be in L)
- This means that $|x| + i \cdot |y| + |z|$ must also be prime
- Let $i = |x| + |z|$
- Then $|x| + i \cdot |y| + |z| = |x| + (|x| + |z|) \cdot |y| + |z|$

$= (|x| + |z|) \cdot (1 + |y|)$

This is composite (i.e., not prime). So for at least one value of i, the resulting string is not in L. Therefore, L is not regular.