Nonregular languages

Shakir Ullah Shah

Nonregular languages

Definition:

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

Notes:

  - By Kleene's theorem, a non-regular language can also **not** be accepted by any FA or TG.

  - All languages are either regular or non-regular; none are both.

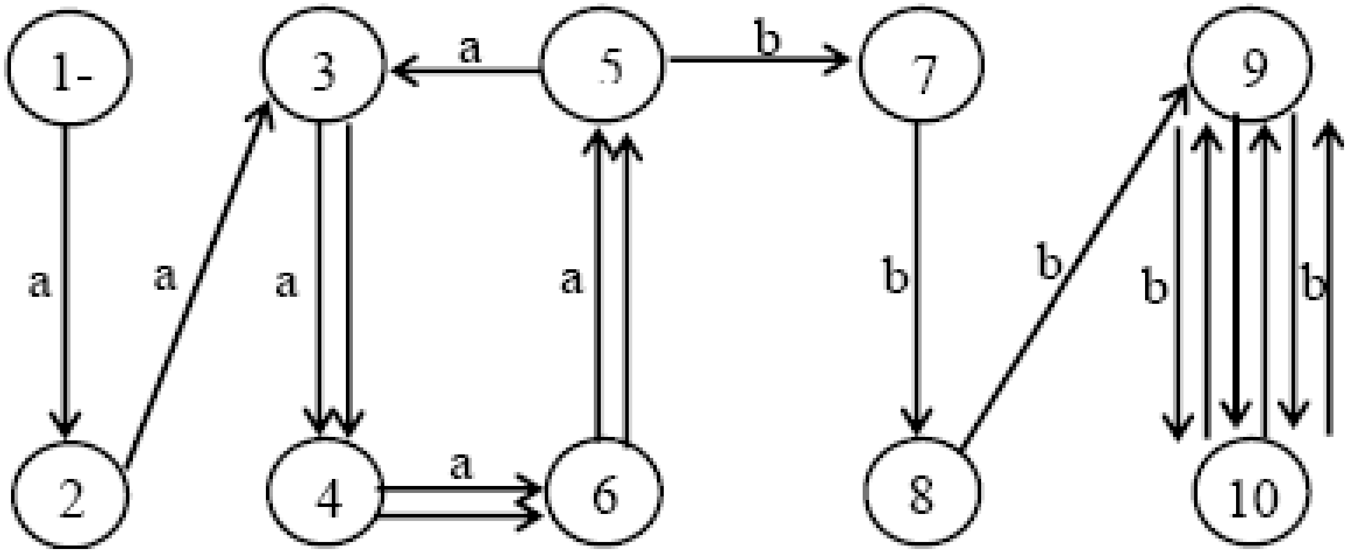
- The languages **PALINDROME** and **PRIME** are the examples of nonregular languages.

Example

- Consider the language L = {Λ, ab, aabb, aaabbb, …} *i.e.* {$\mathbf{a}^n$ $\mathbf{b}^n$: n=0,1,2,3,…}
- Suppose, it is required to prove that this language is nonregular.
  - Let, contrary, L be a regular language then by Kleene's theorem it must be accepted by an FA, say, F.
  - Since every FA has finite number of states then the language L (being infinite) accepted by F must have words of length more than the number of states. Which shows that, F must contain a circuit.

Example Cont….

- For the sake of convenience suppose that F has 10 states. Consider the word $a^9 b^9$ from the language L and let the path traced by this word be shown as under.



Example Cont….

- But, looping the circuit generated by the states 3,4,6,5,3 with a-edges once more, F also accepts the word $a^{9+4} b^9$, while $a^{13} b^9$ is not a word in L.
- It may also be observed that, because of the circuit discussed above, F also accepts the words $a^9 (a^4)^m b^9$, m=1,2,3, …
- Moreover, there is another circuit generated by the states 9,10,9. Including the possibility of looping this circuit, F accepts the words $a^9 (a^4)^m b^9 (b^2)^n$ where m,n=0,1,2,3,…(m and n not being 0 simultaneously). Which shows that F accepts words that are not belonging to L.

Example Cont….

- Similarly for finding FAs accepting other words from L, they will also accept the words which do not belong to L.
- Thus there is no FA which accepts the language L. which shows, by Kleene's theorem, that the language L can't be expressed by any regular expression.
- It may be noted that apparently $a^n b^n$ seems to be a regular expression of L, but in fact it is not.

Summary

- In summary, we can always choose a word in L that is large enough so that its path through the FA has to contain a circuit.
  - Once we find that some path with a circuit can reach a final state, we ask ourselves what happens to a path that is just like the first one, but that loops around the circuit one extra time and then proceeds identically through the machine.
  - The new path also leads to the same final state, but it is generated by a different input string which is **not** in the language L.
  - We then can conclude that there is no FA that can accepts all the words in L and only the words in L. Therefore, L is non-regular.
- This idea is called the **pumping lemma** discovered by Bar-Hillel, Perles, and Shamir in 1961. It is called “pumping” because we pump more letters into the middle of the words. It is called “lemma” because it is used as a tool to prove other results (i.e., certain specific languages are non-regular).

Theorem 13

- **Let** L **be any regular language that has infinitely many words. Then there exist some three strings** x, y, **and** z **(where** y **is not the null string) such that all the strings of the form**

$$xy^nz \textbf{ for } n = 1, 2, 3, \ldots$$

are words in L**.**

Proof of theorem 13

- Since L is regular, there is an FA that accepts exactly the words in L.

- Let *w* be some word in L that has more letters than there are states in FA.

- When *w* generates a path through the machine, the path cannot visit a new state for each letter read, because there are more letters than states. Therefore, the path must at some point revisit a state that it has already visited. In other words, the path contains a circuit in it.

Proof of theorem 13 contd.

Let's break the word *w* up into 3 parts:

1. Part 1: Starting at the beginning, let *x* denote all the letters of *w* that lead up to the **first** state that is revisited. Note that *x* may be the null string if the path revisits the start state as its first revisit.

2. Part 2: Starting at the letter after the substring *x*, let *y* denote the substring of w that travels around the circuit coming back to the same state the circuit began with. Because there must be a circuit, y cannot be the null string, and y contains the letters of w for exactly one loop around this circuit.

3. Part 3: Let *z* be the rest of *w*, starting at the letter after *y* and going to the end of the string *w*. Note that *z* could be null, or the path for *z* could also loop around the *y*-circuit or any other. That means that what *z* does is arbitrary.

- Clearly, from the definitions of these substrings, we have *w* = *xyz*. Recall that *w* is accepted by the FA.

Proof of Theorem 13 contd.

- What is the path for the input string *xyyz*?
- This path follows the path for *w* in the first part *x* and leads up to the beginning of the place where w looped around a circuit.
- Then like *w*, it inputs the substring *y*, which causes the machine to loop back to this same state again.
- Then, again like *w*, it inputs the substring *y*, which causes the machine to loop back to this same state another time.
- Finally, just like *w*, it proceeds along the path dictated by the substring *z* and ends at the same final state that *w* did.
- Hence, the string *xyyz* is accepted by this machine and therefore must be in the language L.

Proof of Theorem 13 contd.

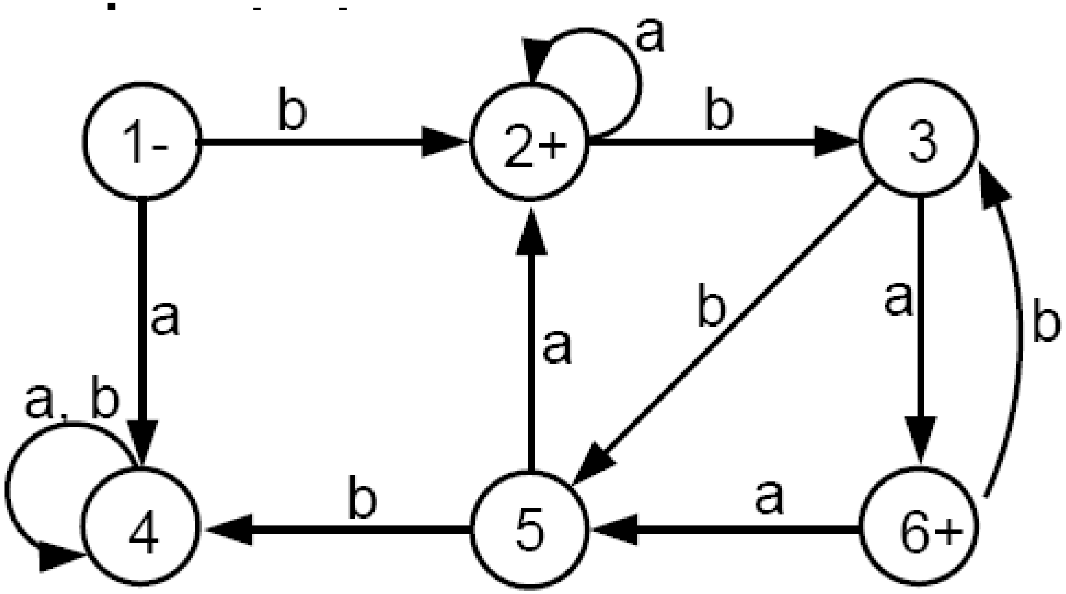
- Similarly, the strings xyyyz, xyyyyz, ... must also be in L.

- In other words, L must contain all strings of the form:

  xy^nz for $n = 1; 2; 3; \ldots$

Example

- Consider the following FA that accepts an infinite language and has onl

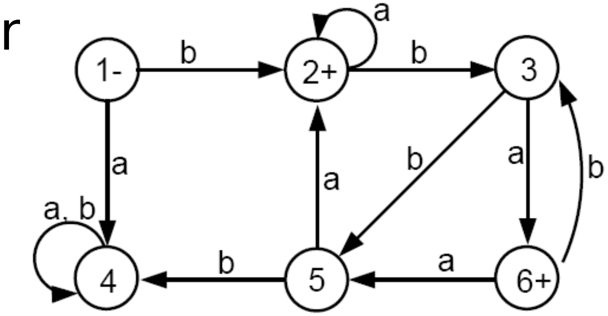


- Consider the word $w = bbbababa$

Example contd.

- The *x*-part goes from the - state up to the first circuit: substring *b*

- The y-part goes around the circuit cor and 5: substring *bba*



- The *z*-part is substring *baba*

- What happens to the input string *xyyz = (b)(bba)(bba)(baba)*?

- This string will loop twice around the circuit and is accepted.

- The same thing happens with *xyyyz*, *xyyyyz*, and in general, for xy^nz.

- Let us use Theorem 13 to show again that the language $L = \{a^n b^n\}$ is not regular.
- If L is regular then Theorem 13 says that there must be strings x, y, and z such that all words of the form xy^nz are in L.
- A typical word of L looks like this: aaa…aaaabbbb…bbb. How to break it into x, y, and z?
- If *y* contains entirely a's, then when we pump it to *xyyz*, this string will have more a's than b's, which is not allowed in L.
- Similarly, if y is composed of only b's then xyyz will have more b's than a's, and is not allowed in L either.

- If y consists of both **a**s and **b**s. In this case the substring xyyz may have the same number of **a**s and **b**s, but they will be out of order with some **a**s before **b**s.
  - Therefore, *xyyz* can not be a word in *L*.

- The above arguments show that the pumping lemma cannot apply to L and therefore L is not regular.

Example

- Let EQUAL be the language of all words (over the alphabet ∑ = {a; b}) that have the same total number of a’s and b’s:

- EQUAL = {Λ; ab; ba; aabb; abab; abba; baab; baba; bbaa; aaabbb; …}

- Can you show that EQUAL is not regular?

- Let L = $\{a^n b a^n\}$ = {b; aba; aabaa; …}

- Can you show that L is not regular?

Theorem 14

- **Let** L **be an infinite language accepted by a finite automaton with** N **states. Then, for all words** w **in** L **that have more than** N **letters, there are strings** x**,** y**, and** z**, where** y **is not null and** length(x) + length(y) **does not exceed** N**, such that** w = xyz **and all strings of the form** xy^nz **for** n = 1; 2; 3; … **are in** L**.**

- This is obviously just another version of Theorem 13 (the pumping lemma), for which we have already provided the proof.
- The purpose of stressing the issue of lengths is illustrated in the following example.

Example

- We will show that the language PALINDROME is not regular.

- We cannot use the first version of the pumping lemma (Theorem 13)vbecause the strings

  $$x = a; y = b; z = a$$

  satisfy the lemma and do not contradict the language, since all the strings of the form $xy^nz = ab^na$ are words in PALINDROME.

- So, we will use the second version of the pumping lemma (Theorem 14) to show that PALINDROME is non-regular.

Example contd.

- Suppose for the contrary that PALINDROME were regular, then there would exist some FA that accepts it.
- For the sake of argument, assume that this FA has 77 states.
- Then, the palindrome $w = a^{80}ba^{80}$ must be accepted by this FA.
- Because w has more letters than the FA has states, by Theorem 14 we can break w into three parts: x, y, and z.
- Since length(x) + length(y) $\leq$ 77 (by Theorem 14), the strings x and y must both be made of all a’s, since the first 77 letters of w are all a’s.

- Hence, when we form *xyyz*, we are adding more a’s to the front of w, but we are not adding more a’s to the back of w.

- Thus, the string *xyyz* will be of the form

  $a^{(\text{more than } 80)}ba^{80}$

  and obviously is NOT a palindrome.

- This is a contradiction, since Theorem 14 says that *xyyz* must be a palindrome. Hence, the language PALINDROME is NOT regular.