# Theory of Automata
# Pumping Lemma

Week 6

# Contents

- Definition of non-Regular Languages
- Theorem 13
  - Proof & Example
- Theorem 14
  - Proof & Example

# Introduction

- By using FAs and regular expressions, we have been able to define many languages. Although these languages have many different structures, they take only a few basic forms:
  - Languages with required substrings
  - Languages that forbid some substrings
  - Languages that begin or end with certain substrings
  - Languages with certain even (or odd) properties, and so on.
- We now turn our attention to some new forms, such as the language PALINDROME, or the language PRIME of all words $a^p$, where p is a prime number.
- We shall see that neither of these is a regular language. We can describe them in English, but they can not be defined by an FA. We need to build more powerful machines to define them.

# Definition of Non-Regular 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.

# Case Study

- Consider the language

  L = {Λ; ab; aabb; aaabbb; aaaabbbb; aaaaabbbbb; …}
- The language L can also be written as

  L = {$a^n b^n$ for n = 0; 1; 2; 3; …}

  or for short

  L = {$a^n b^n$}
- Note that although L is a subset of many regular languages, such as a*b*; the language defined by ab also includes such strings as aab and bb that are not in L.

# Example

- We shall show that the language L = $\{a^n b^n\}$ is non-regular.
- Suppose on the contrary that L were regular. Then there must exist some FA that accepts L.
- Just for the sake of argument, let us assume that this FA has 95 states.
- We know that this FA must accept the word $a^{96}b^{96}$.
- The first 96 letter a's of this input string trace a path through this machine.
- The path cannot visit a new state when each input letter is read, because there are only 95 states. Therefore, at some point the path returns to a state that it has already visited.
- In other words, the path must contain a **circuit** in it. (A circuit is a loop that can be made of several edges).

- So, the path first wanders up to the circuit and then starts to loop around the circuit (maybe many times) until a b is read from the input.
- At this point, the path can take a different turn following the b-edges, and eventually end up at a final state where the word $a^{96}b^{96}$ is accepted.
- Just for the sake of argument again, let us say that the circuit that the a-edge path loops around has 7 states in it.
- Then what would happen to the input string $a^{96+7}b^{96}$?
- Just as in the case of the input string $a^{96}b^{96}$, the input string $a^{96+7}b^{96}$ would produce a path through the machine and loop around the circuit exactly in the same way, but **one more time** than the path produced by the input string $a^{96}b^{96}$.
- Both paths, at exactly the same state in the circuit, begin to branch off on the b-road.

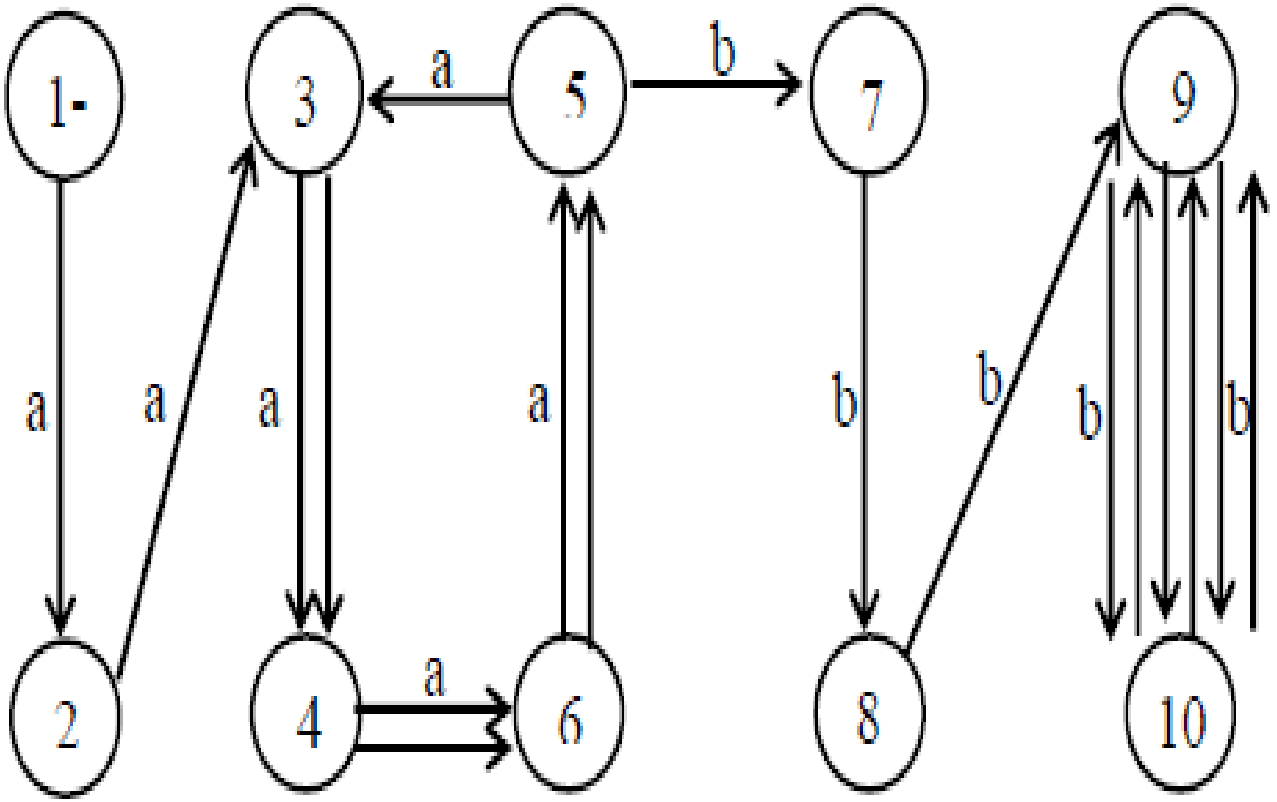
- Once on the b-road, they both go the same 96 b-steps to arrive at the same final state.

- But this would mean that the input string $a^{103}b^{96}$ is also accepted by this machine.

- This is a **contradiction** since we assume that this FA accepts exactly the words in L = $\{a^n b^n\}$.

- This contradiction means that the FA that accepts exactly the words in L does not exist. In other words, L is non-regular.

- 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).


1-
3
5
7
9
2
4
6
8
10
a
a
a
a
a
a
b
b
b
b
b

# 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**

$$xy^n z \text{ for } n = 1, 2, 3, \ldots$$

**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.

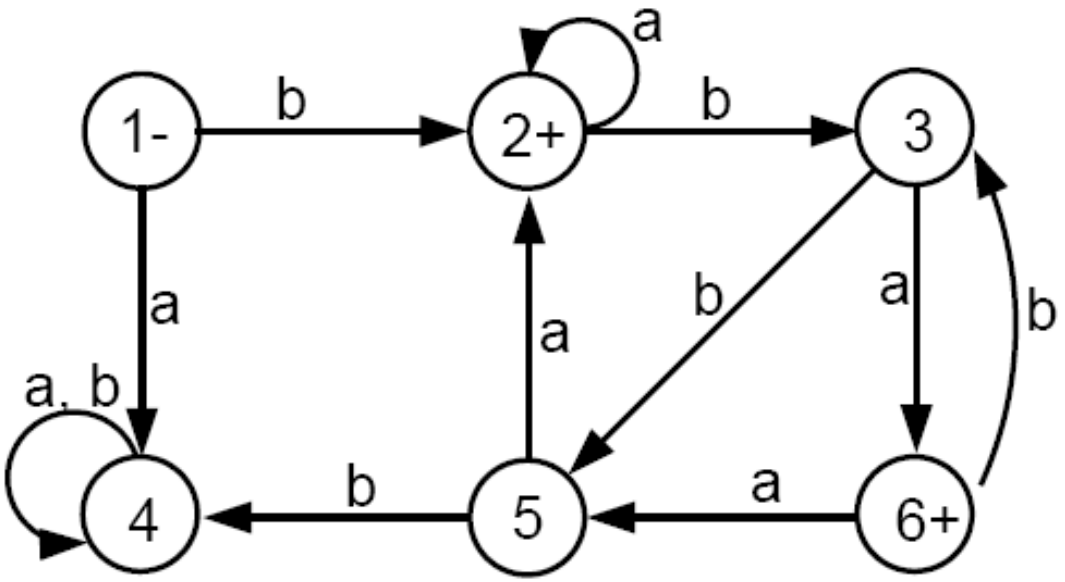
- Similarly, the strings xyyyz, xyyyyz, ... must also be in L.

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

  $xy^nz$ for n = 1; 2; 3; …

# Example

- Consider the following FA that accepts an infinite language and has only six states:




- 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 consisting of states 2, 3, 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 $xy^n z$ 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 have some a's followed by some b's then y must contain the substring *ab*.
  - So, *xyyz* must have 2 substrings *ab*'s.
  - But every word in L has exactly one substring *ab*.
  - 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) because 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) ≤ 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.

# Example

- Consider the language
  PRIME = {$a^p$ where p is a prime}

- Recall that *a prime is a positive integer greater than 1 whose only positive divisors are 1 and itself*, for example 2, 3, 5, 7 ...

- Hence,
  PRIME = {$a^p$ where p is a prime}
  = {aa; aaa; aaaaa; aaaaaaa; …}

- Can you show that PRIME is non-regular?