Biçimsel Diller ve Otomata Teorisi

Sunu X
Düzenli Olmayan Diller

İZZET FATİH ŞENTÜRK

The Pumping Lemma

- Some languages can be described in English but they cannot be defined by an FA
  - Such as the language PALINDROME or PRIME (of all words a^p, where p is a prime number)
- A language that cannot be defined by a regular expression is called a **nonregular** language
- By Kleene's theorem, a nonregular language can also not be accepted by any FA or TG
- All languages are either regular or nonregular; none are both

The Pumping Lemma

- Let us define the language L
  - L = {Λ ab aabb aaabbb aaaabbbb aaaaabbbbb …}
- We could also define this language by the formula
  - L = {$a^n b^n$ for n = 0 1 2 3 4 5 …}
  - Or for short L = {$a^n b^n$}
- It is a subset of many regular languages, such as **a***b***
  - Note that $a^n b^n$ does not include *aab* or *bb*

The Pumping Lemma

- Suppose on the contrary that this language were regular
  - Then there would have to exist some FA that accepts it
  - Let us picture one of these FAs (there might be several)
- This FA might have many states
  - Let us say that it has 95 states
  - Yet, we know it accepts the word $a^{96}b^{96}$
  - The first 96 letters of this input string are all *a*'s and they trace a path through this machine

The Pumping Lemma

- The path cannot visit a new state with each input letter read
  - Because there are only 95 states
  - Therefore, at some point the path returns to a state that it has already visited
- The first time it was in that state it left by the *a*-road
  - The second time it is in that state it leaves by the *a*-road again
  - Even if it only returns once, we say that the path contains a circuit (A **circuit** is a loop that can contain several edges)

The Pumping Lemma

- First , the path wanders up to the circuit and then it starts to loop around the circuit maybe many times
  - It cannot leave the circuit until a *b* is read from the input
  - Then the path can take a different turn
  - For example, the path could make 30 loops around a three-state circuit before the first *b* is read
- After the first *b* is read, the path goes off and does some other stuff following *b*-edges and eventually winds up at a final state where the word $a^{96}b^{96}$ is accepted

The Pumping Lemma

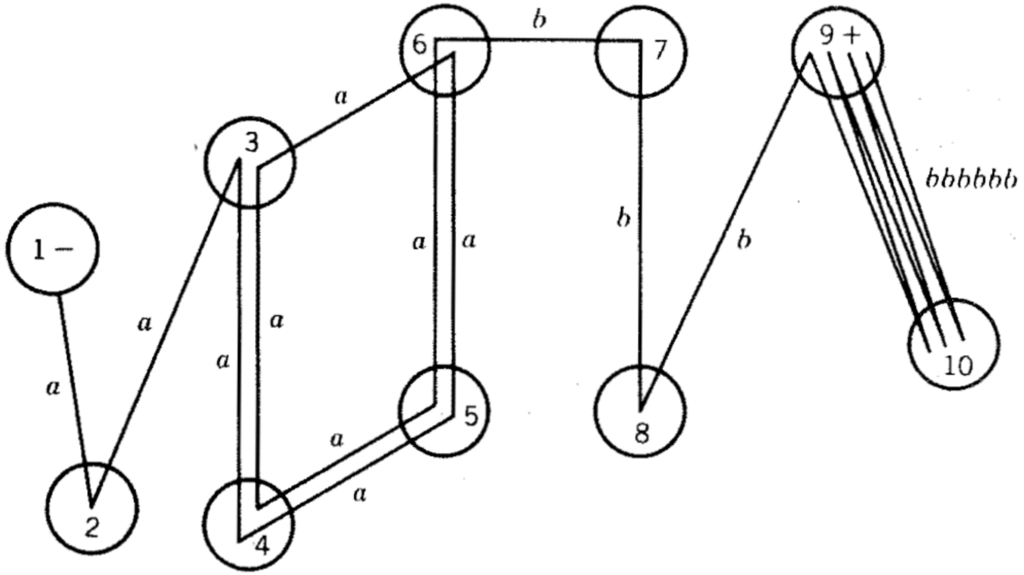
- Let us say that the circuit that the *a*-edge path loops around has seven states in it
  - The path enters the circuit
  - Loops around it
  - Then goes off on the *b*-line to a final state
- What would happen to the input string $a^{96+7}b^{96}$?
  - The path loops around this circuit one more time (precisely one extra time)
  - That string is not in the language L = $\{a^n b^n\}$
  - This is a contradiction, in other words, L is <u>nonregular</u>

The Pumping Lemma

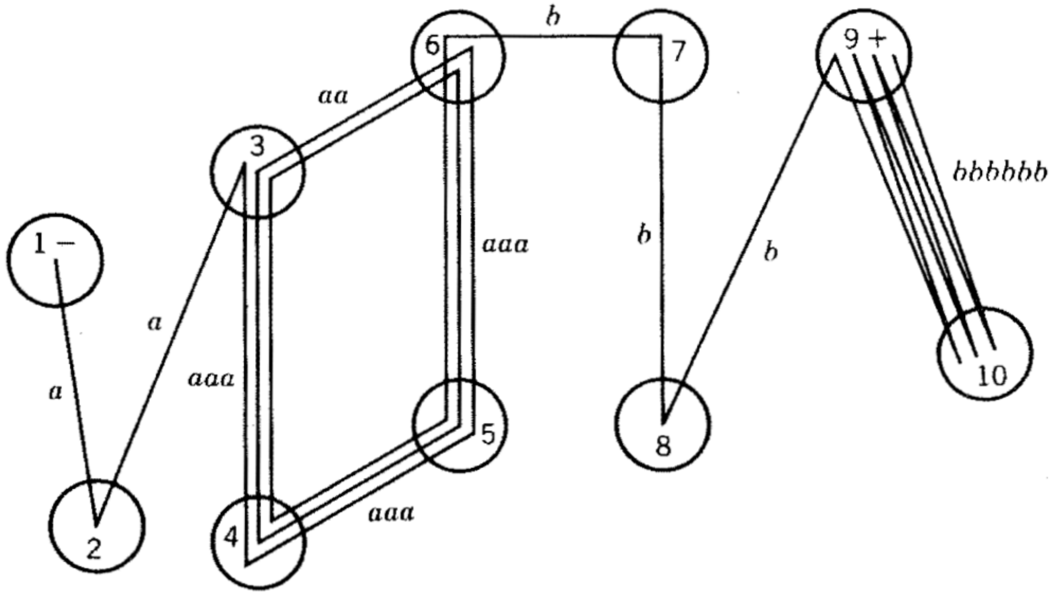
- Let us review what happened
- We choose a word in L that was so large (had so many letters) that its path through the FA had to contain a circuit
- Once we found that some path with a circuit could reach a final state, we ask..
  - What happends to a path that is just like the first one, but that loops 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 – an input string not in the language L

The Pumping Lemma

- Let the path a^9b^9 be



- The path for $a^{13}b^9$



The Pumping Lemma – Theorem

- Let L be any regular language that has infinitely many words
- There exist some three strings x, y, and z (where y is not the null string)
  - xy^nz for $n = 1\ 2\ 3 \ldots$

  are words in L

The Pumping Lemma – Proof

- If L is a regular language, then there is an FA that accepts exactly word in L
  - The machine has only finitely many states
  - L has infinitely many words in it (there are arbitrarily long words in L)
- Let w be some word in L that has more letters in it that there are states in the machine
- When this word generates a path through the machine, the path cannot visit a new state for each letter
  - Because there are more letters than states
  - It must at some point revisit a state that it has been to before

The Pumping Lemma – Proof

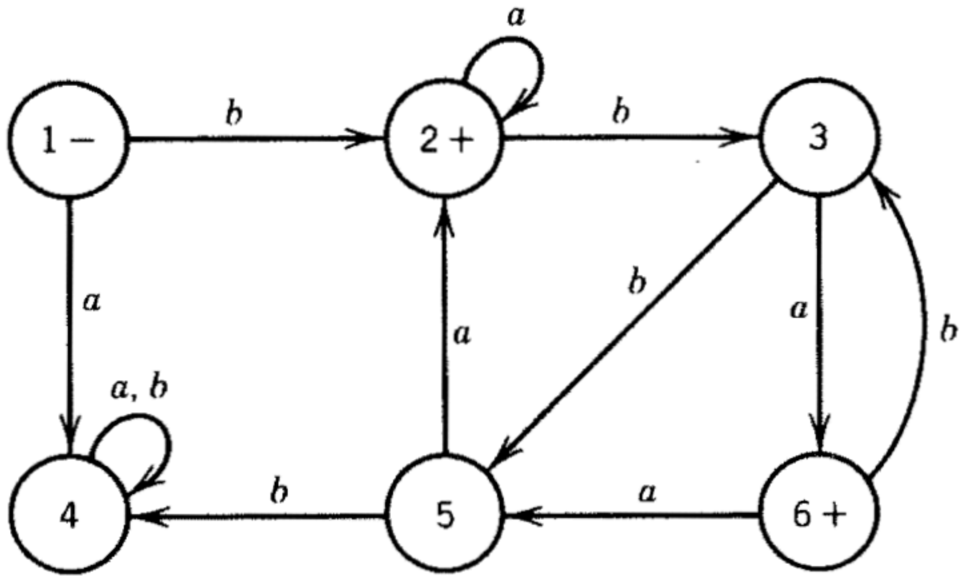
- Let us break the word w up into three parts
  - **Part 1** (*x*) All the letters of w starting at the beginning that lead up to the first state that is revisited. <u>x may be the null string</u> if the path for w revisits the start state as its first revisit
  - **Part 2** (*y*) Starting at the letter after x, y travels around the circuit coming back to the same state the circuit began. Because there must be a circuit, <u>y cannot be null</u>
  - **Part 3** (*z*) The rest of w. <u>z could be null</u>
- *w = xyz*

The Pumping Lemma

- What is the path through this machine of input string
  - xyyz?
  - xyyyz?
  - xyyyyyyyyyyz?
- All these must be accepted by the mahine and therefore are all in L
- L must contain all strings of the form
  - xy^nz for n = 1 2 3 …

Example

- The machine below accepts an infinite language and has only six states
- Any word with six or more letters must correspond to a path that includes a circuit (also some words with fewer than six letters such as *baaa*)
- We will consider in detail: w = *bbbababa*

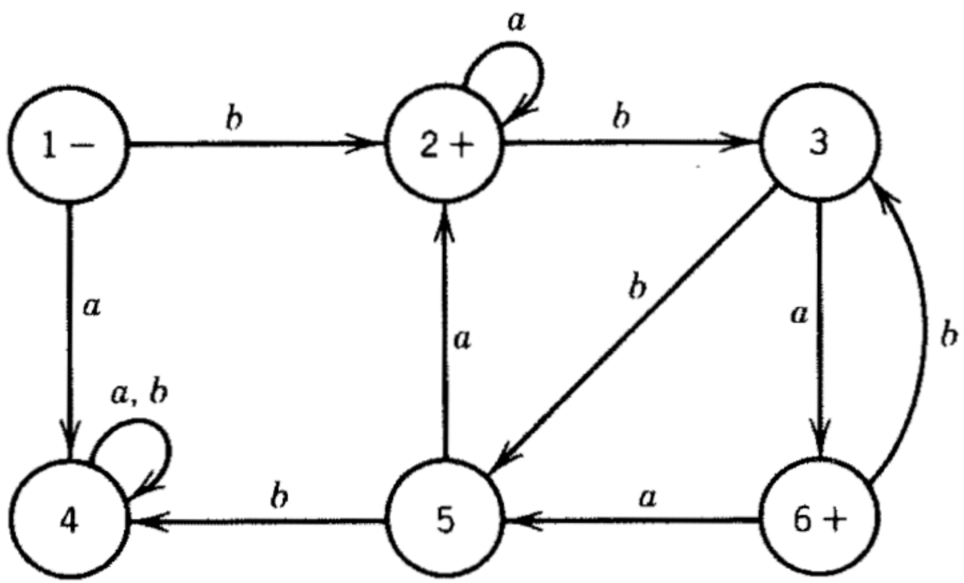


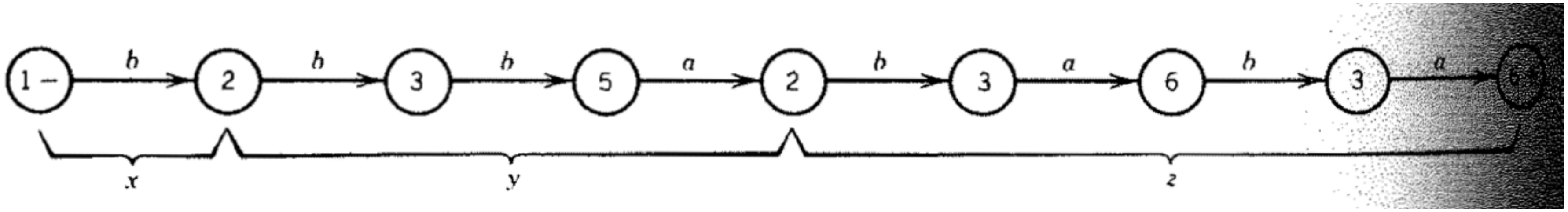
Example

- w = *bbbababa* has more than six letters (includes a circuit)
- w = b   bba baba

      x   y   z

What would happen to *xyyz?*
x y y z = b bba bba baba





Example

- The pumping lemma says that there must be strings *x*, *y*, and *z* such that all words in the form xy^nz are in *L*
  - Is this possible?
- A typical word of *L*
  - *aaa … aaaabbbb … bbb*
- How do we break this into three pieces as *x*, *y*, and *z*?
  - *y* is made entirely of *a*’s (*xyyz*, more *a*’s than *b*’s)
  - y is made entirely of *b*’s (*xyyz*, more *b*’s than *a*’s)
  - *y* contains some *a*’s and some *b*’s (*xyyz* would have two copies of substring *ab*)
  - *xyyz* cannot be a word in *L*, <u>*L* is not regular</u>

Example

- The language EQUAL, of all words with the same total number of *a*’s and *b*’s is also <u>nonregular</u>
- EQUAL = {Λ ab ba aabb abab abba baab baba bbaa aaabbb …}
- The language $a^n b^n$ is the intersection of all words defined by the RE **a*****b*** and the language EQUAL
  - {$a^n b^n$ = **a*****b*** ∩ EQUAL}
- If EQUAL were a regular language, then $a^n b^n$ would be the intersection of two regular languages. Because $a^n b^n$ is not regular, EQUAL cannot be

Example

- Consider the language $a^n b a^n$ = {*b aba aabaa* …}
- If this language were regular, there would exist three strings x, *y*, and z such that
  - *xyx* and *xyyz* were both words in this language
  - We can show that this is impossible

Example

- Consider the language $a^n b a^n$ = {*b aba aabaa* …}
- Observations
  1. If the *y* string contained *b*, then *xyyz* would contain two *b*'s (no word in this language can have)
  2. If the *y* string is all *a*'s then the *b* in the middle of the word *xyz* is in the *x*-side or *z*-side. In either case, *xyyz* has increased the number of *a*'s either in front of the *b* or after the *b* (but not both)
- Conclusions
  1. *xyyz* does not have its *b* in the middle and is not in the form $a^n b a^n$
  2. This language cannot be pumped and is therefore not regular

Example

- Consider the language $a^n b^n a b^{n+1}$ for n = 1, 2, 3, …
  - Show that if *xyz* is in this language, then *xyyz* is not
- Observations
  1. If we know the total number of *a*’s we can calculate the number of *b*’s and vice versa. <u>No two different words have the same number of *a*’s or *b*’s</u>
  2. All words have exactly two substrings equal to *ab* and one equal to *ba*
  3. If *xyz* and *xyyz* are both in this language, then *y* cannot contain either *ab* or *ba* because then *xyyz* would have too many

Example

- Consider the language $a^n b^n a b^{n+1}$ for n = 1, 2, 3, …
- Conclusions
  1. Because *y* cannot be **Λ**, it must contain either only *a*'s or *b*'s, any mixture contains the forbidden substrings (observation 3)
  2. If *y* is solid *a*'s, then *xyz* and *xyyz* are different words with the same total *b*'s, violating observation 1. If *y* is solid *b*'s, then *xyz* and *xyyz* are different words with the same number of *a*'s violating observation 1
  3. It is impossible for both *xyz* and *xyyz* to be in this language for any strings x, *y*, and z. The language is unpumpable and not regular

Theorem

- 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 in the form xy^nz (for n = 1 2 3 …) are in L

Example

- We shall show that the language PALINDROME is nonregular
  - We cannot use the first version of the pumping lemma because the strings x=a, y=b, z=a satisfy the lemma and do not contradict the language
  - All words of the form xy^nz = ab^na are in PALINDROME
- Let us consider one of the FAs that might accept this language
  - The machine has 77 states
  - w = $a^{80}ba^{80}$
  - Because it has more letters than the machine has states, we can break *w* into the three parts: *x*, *y*, and *z*

Example

- Because the length of x and y must be in total 77 or less
  - They must both be made of solid *a*'s (because the first 77 letters of *w* are all *a*'s)
  - When we form the word *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* because all the rear *a*'s are in the z-part, which stays fixed at 80 *a*'s)
  - The string *xyyz* is not a palindrome because it will be of the form $a^{\text{more than } 80}ba^{80}$
- But the second version of the pumping lemma says that PALINDROME has to include this string
  - PALINDROME is nonregular

Example

- Let us consider the language PRIME = {a^p where p is a prime}
- Is PRIME a regular language?
  - If it is, then there is some FA that accepts exactly these words
  - Let us suppose that it has 345 states
  - Let us choose a prime number bigger than 345 – for example, 347
- a^{347} can be broken into parts x, y, and z such that xy^nz is in PRIME for any value of n
  - x, y, z are all just a's
  - Let us take n = 348, the word $xy^{348}z$ must be in PRIME

Example

- $xy^{348}z = xyzy^{347}$ (all a's, order doesn't matter)
- $xyzy^{347} = a^{347}y^{347}$ (x, y, and z came originally from a^{347}
- We also know that y is some (nonempty) string of a's
  - Let us say that $y = a^m$
- $a^{347}y^{347} = a^{347}(a^m)^{347}$

  $= a^{347+347m}$

  $= a^{347(m+1)}$
- Because m ≠ 0, we know that 347(m+1) is not a prime number
  - This is a contradiction
  - Therefore, PRIME is nonregular