On beyond Regular!

NON-REGULAR LANGUAGES
Next we’ll be talking about non-Regular Languages

• All part of becoming familiar with the boundaries that define these language classes
• **Closure properties** are one way of doing exploring transformations that keep us *in* the boundaries of the class of Regular Languages
• Now we’re bursting through the boundary!
Next we’ll be talking about non-Regular Languages

• All part of becoming familiar with the boundaries that define these language classes

• **Closure properties** are one way of doing exploring transformations that keep us *in* the boundaries of the class of Regular Languages

• Now we’re bursting through the boundary!
Next we’ll be talking about non-regular languages

• All part of becoming familiar with the boundaries that define these language classes
• **Closure properties** are one way of doing exploring transformations that keep us *in* the boundaries of the class of Regular Languages
• Now we’re bursting through the boundary!

reversing
For the birds

PIGEONHOLE

PRINCIPLE
What is the longest string this DFA can accept without visiting any state more than once?

a) 1
b) 3
c) 4
d) 5
e) None of the above
What is the longest string this DFA can accept without visiting any state more than once?

a) 1  
b) 2  
c) 3  
d) 4  
e) None of the above
Generalizing:

• Given a string $s$ in $L$, and a DFA $M$ that recognizes $L$:
  – If $|s| > |Q|$, then when $M$ processes $s$, it must visit one (or more) state(s) more than once
    • For a second, let’s just consider one state being twice visited
  – Let $t$ be the substring of $s$ that is read between the first and second times the twice-visited state is visited
  – It must be the case that $t$ could appear repeatedly in $s$:
    • $s = [\text{first part of } s]t[\text{last part of } s]$, then also possible in this DFA:
      • $[\text{first part of } s]tt[\text{last part of } s]
      • $[\text{first part of } s]tttttttttt[\text{last part of } s]
  – WHY?? (Discuss in your groups)
On beyond Regular!

THE PUMPING LEMMA
PROVING A LANGUAGE IS NOT REGULAR
Pumping Lemma

• The pumping lemma starts with “If $L$ is a regular language, then...”

• We use the pumping lemma to show that a language is:
  
  – a) Regular
  – b) Not Regular
Pumping Lemma

• The pumping lemma starts with “If L is a regular language, then...”
  – The pumping lemma is a guarantee about regular languages.
  – We use it to show that if a language does not satisfy this guarantee, then it is NOT regular.
Your Script
- “I’m giving you a language L.”

- “Uh...sure...let’s just say it’s Regular.”

- “Excellent. I’m giving you this string s that I made using your p. It is in L and |s| >= p. I think you’ll really like it.”

- “Hm. I followed your directions for xyz, but when I [copy y N times or delete y], the new string is NOT is L! What happened to your 100% Guarantee??!”

Pumping Lemma’s Script
- “Is L Regular? In this shop I only work on Regular Languages.”

- “Thanks. For the language L that you’ve given me, I pick this nice pumping length I call p.”

- “Great string, thanks. I’ve cut s up into parts xyz for you. I won’t tell you what they are exactly, but I will say this: |y| > 0 and |xy| <= p. Also, I make you this 100% Guarantee: you can remove y, or copy it as many times as you like, and the new string will still be in L, I promise!”

- “Well, then L wasn’t a Regular Language. Since you lied, the Guarantee was void. Thanks for playing.”
How to use the Pumping Lemma Script to write a Pumping Lemma Proof (1)

• Let’s look at the parts of your script:
  – **Picking language L:**
    • Done—given to you in the homework/exam problem
  – **Start the proof:**
    • Proof by contradiction: Assume L is regular
  – **Picking p?** a) TRUE, b) FALSE
  – **Picking s:**
    • Here you need to get creative
    • Try several things, remember:
      – s must be in L
      – |s| must be >= p  (often do this by making some pattern repeat p times, e.g., s = “0^p1^p” is clearly of length >= p)
The Pumping Lemma: A One-Act Play

Your Script

• “I’m giving you a language L.”

• “Uh…sure…let’s just say it’s Regular.”

• “Excellent. I’m giving you this string s that I made using your p. It is in L and |s| >= p. I think you’ll really like it.”

• “Hm. I followed your directions for xyz, but when I [copy y N times or delete y], the new string is NOT in L! What happened to your 100% Guarantee??!”

Pumping Lemma’s Script

• “Is L Regular? In this shop I only work on Regular Languages.”

• “Thanks. For the language L that you’ve given me, I pick this nice pumping length I call p.”

• “Great string, thanks. I’ve cut s up into parts xyz for you. I won’t tell you what they are exactly, but I will say this: |y| > 0 and |xy| <= p. Also, I make you this 100% Guarantee: you can remove y, or copy it as many times as you like, and the new string will still be in L, I promise!”

• “Well, then L wasn’t a Regular Language. Since you lied, the Guarantee was void. Thanks for playing.”
How to use the Pumping Lemma Script to write a Pumping Lemma Proof (2)

• Let’s look at the parts of your script:
  – Picking x, y, z? a) TRUE b) FALSE
  – Picking i (the number of times to copy part y):
    • For many problems, two main ways to go: i = big: (say, 3 or p), or i = 0 (delete y)
    • Try both and see if either “breaks” s (makes a string not in L)
    • If several tries don’t work, you may need to design a different s
  – End the proof:
    • Once you find an i/s pair that “breaks” the warranty, this is a contradiction, and so the assumption is false, and L is not Regular. QED.
Pumping Lemma Practice

• Thm. \( L = \{0^n1^m0^n \mid m,n \geq 0\} \) is not regular.
• Proof (by contradiction):
  • Assume (towards contradiction) that \( L \) is regular. Then the pumping lemma applies to \( L \). Let \( p \) be the pumping length. Choose \( s \) to be the string _________. The pumping lemma guarantees \( s \) can be divided into parts \( xyz \) s.t. for any \( i \geq 0 \), \( xy^i z \) is in \( L \), and that \( |y| > 0 \) and \( |xy| \leq p \). But if we let \( i = ____ \), we get the string XXXX, which is not in \( L \), a contradiction. Therefore the assumption is false, and \( L \) is not regular. Q.E.D.

a) \( s = 00000100000, i = 5 \)
b) \( s = 0^p10^p, i=0 \)
c) \( s = (010)^p, i=5 \)
d) None or more than one of the above
e) I don’t understand this at all
Pumping Lemma Practice

• Thm. $L = \{0^n1^n \mid n \geq 0\}$ is not regular.
• Proof (by contradiction):
  • Assume (towards contradiction) that $L$ is regular. Then the pumping lemma applies to $L$. Let $p$ be the pumping length. Choose $s$ to be the string _______. The pumping lemma guarantees $s$ can be divided into parts $xyz$ s.t. for any $i \geq 0$, $xy^iz$ is in $L$, and that $|y| > 0$ and $|xy| \leq p$. But if we let $i = ____$, we get the string XXXX, which is not in $L$, a contradiction. Therefore the assumption is false, and $L$ is not regular. Q.E.D.

a) $s = 010101$, $i = 0$
b) $s = 000000111111$, $i = 6$
c) $s = 0^p1^p$, $i = 1$
d) $s = 0^i1^i$, $i = 5$
e) None or more than one of the above
Pumping Lemma Practice

• Thm. \( L = \{ww_r \mid w_r \text{ is the reverse of } w\} \) is not regular.
• Proof (by contradiction):
  • Assume (towards contradiction) that \( L \) is regular. Then the pumping lemma applies to \( L \). Let \( p \) be the pumping length. Choose \( s \) to be the string _________. The pumping lemma guarantees \( s \) can be divided into parts \( xyz \) s.t. for any \( i \geq 0 \), \( xy^iz \) is in \( L \), and that \(|y| > 0 \) and \(|xy| \leq p\). But if we let \( i = _____ \), we get the string XXXX, which is not in \( L \), a contradiction. Therefore the assumption is false, and \( L \) is not regular. Q.E.D.

a) \( s = 000000111111, i=6 \)
b) \( s = 0^p0^p, i=2 \)
c) \( s = 0^p110^p, i = 2 \)
d) None or more than one of the above