CSE 105
Theory of Computation

- Professor Jeanne Ferrante

Mary gave a cat to John
Reminders and Announcements

• HW 3: Due Fri Apr 15 by 11:59 pm
• Review Session: Mon Apr 18, 8 pm - 9:50 pm Peterson 108
• Student seating chart for Exam 1 is on Ted
• Midterm Exam 1: Study Guide is out on Ted
  – Tips for Studying:
    • Start each problem by yourself first, e.g. outline your approach
    • Start hard, jump to easy
    • Write out solutions by hand
    • Go over with others
  – Exam Time: Wed, April 20, 8:00 pm - 9:50 pm
  – Exam Location: WLH 2001
  – Please look up in advance in the seating chart so you can sit in your designated seat!
Today’s Agenda

• More Non-Regular Languages!
  – Pumping Lemma Review
  – Practice Using the Pumping Lemma
  – Using Diagonalization (NOT on Exam 1!!)
Review: Regular Expression Language

Let $L$ be the language of this regular expression: $(1 \ (0 \cup 1^*))^*$

Which of the following is true of $L$?

A. $\varepsilon$ is in $L$
B. $L$ contains “00”
C. 1’s are never followed by 0’s in any string in $L$
D. More than one or none of the above
Review: Write a Regular Expression for L

Let $L = \{ w \in \{0,1\}^* \mid |w| \geq 3$ and its third symbol is 0}\}$

Let $\varepsilon = \{0,1\}^3$

$(0 \cup 1) (0 \cup 1) 0 (0 \cup 1)^*$
On beyond Regular!

**USING THE PUMPING LEMMA**

**PROVING A GIVEN LANGUAGE IS NOT REGULAR**
We’ll use the Pumping Lemma to show a given language \( L \) is **not** Regular

- For each regular language \( L \)
  - [We assume \( L \) regular and hope to get a contradiction]

- There is a pumping length \( p \) for \( L \)
  - [Pumping lemma gives you a number \( p \) for \( L \)]

- For every string \( s \) in \( L \) of length \( \geq p \)
  - [You wisely choose a string \( s \) in \( L \) at least as long as \( p \)]

- There are strings \( x, y, z \) with \( s = xyz, \mid y \mid > 0, \mid xy \mid \leq p \)
  - [Given by the pumping lemma]

- For each \( i \geq 0 \), \( xy^i z \) is in \( L \).
  - [You choose an \( i \) that leads to contradiction! Therefore the assumption was false, \( L \) is NOT regular]
The Pumping Lemma: A One-Act Play

Your Script

- “I’m giving you a language L.”

- “Uh...let’s just say it’s Regular.”

- “Excellent. I’m giving you this string s that I made using the pumping length p you gave me. 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% Warranty??!”

Pumping Lemma’s Script

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

- “Good. For the Regular 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% Lifetime Warranty: 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 Warranty was void. Thanks for playing.”
The Pumping Lemma: A One-Act Play

Your Script

• “I’m giving you a language L.”

• “Uh...let’s just say it’s Regular.”

• “Excellent. I’m giving you this string s that I made using the pumping length p you gave me. 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% Warranty??!”

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% Lifetime Warranty: 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 Warranty 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 s:
    • Here you need to get creative
    • Try several things, remember:
      • s must be in L
      • |s| must be $\geq p$  (often do this by making some pattern repeat p times, e.g., $s = \text{"0}^p1^p\text{"}$ is clearly of length $\geq p$)
How to use the Pumping Lemma Script to write a Pumping Lemma Proof (2)

• Let’s look at the parts of your script:
  – Picking $i$ (the number of times to copy part $y$):
    • For many problems, two main ways to go:
      • $i = \text{big}$: (say, 3 or $p$), or
      • $i = \text{small}$: 0 (delete $y$) *Never* choose 1!
    • Try both and see if either “breaks” membership in $L$ (produces a string not in $L$)
    • If several tries don’t work, you may need to design a different $s$
  – 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.
$L = \{w \mid w = a^n b^n \text{ for } n \geq 0\}$ is not regular.

**Proof:** Assume $L$ is regular. Then the pumping lemma holds for $L$.

- Let $p$ be the pumping length for $L$ given by the pumping lemma.
- Consider the string $s = a^p b^p$ in $L$ of length $> p$.
- Because $s \in L$ and is of length $\geq p$, by the pumping lemma, $s$ can be split into 3 pieces, $s = xyz$, with:
  1. $|xy| \leq p$ and
  2. $|y| > 0$
  3. for $i \geq 0$, the string $xy^iz \in L$. 
L = \{ w \mid w = a^n b^n \text{ for } n \geq 0 \} \text{ not regular, Cont’d.}

- But since \( s = a^p b^p = xyz \) and \( |xy| \leq p \), it must be that \( xy \) consists only of a’s, and since \( |y| > 0 \),
- \( y \) consists of one or more a’s.

- Let \( i = 0 \), obtaining the string \( xz \). \text{By condition 3 of the pumping lemma, } xz \in L.

- String \( xz \) has fewer a’s than \( s = xyz \), but has the same number of b’s. This is because \( y \) has at least 1 a, and \( y \) does not occur in \( xz \), but \( xz \) has the same number of b’s as \( xyz \). Therefore \textbf{xz is not in } L, \text{ by def. of } L, \text{ contradicting our assumption that } L \text{ was regular.}

- We conclude that \( L \) is not regular.
Non-Regular Languages

- Pumping Lemma gives a method for moving outside the boundary of Regular languages
- Closure properties of Regular Languages keep us in the boundaries of the class of Regular Language.
  - Union
  - Intersection
  - Concatenation
  - Star
  - Complement
  - Reversing
On beyond Regular!

PUMPING LEMMA PRACTICE

PROVING A GIVEN LANGUAGE IS NOT REGULAR
Pumping Lemma Practice

Thm. L = \{0^n1^m0^n | 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 ___________. ...

Qu: Which of these strings s satisfy s in L and |s| \geq p?
A. s = 00000100000
B. s = 0^p10^p
C. s = (010)^p
D. None or more than one of the above
Thm. $L = \{0^n1^m0^n | m,n \geq 0\}$ is not regular. (Continued)

Proof 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 __________, $|s| \geq p$. 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.

Qu: Which choice of $s$ in $L$, $|s| \geq p$, and $i$, complete the proof?

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
Thm. \( L = \{0^n1^n \mid n \geq 0\} \) is not regular.

Proof 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 \( \underline{__________} \), \(|s| \geq p\). 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 \( i = \underline{_____} \), we get the string \( \underline{XXXX} \), which is not in \( L \), a contradiction. Therefore the assumption is false, and \( L \) is not regular. Q.E.D.

Qu: Which choice of \( s \) in \( L \), \(|s| \geq p\) and \( i \) complete the proof?
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 __________, \(|s| \geq p\). 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.

Which \( s \) and \( i \) complete the proof?

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
Counting one of them could take forever!

COUNTABLE SETS
1-1 Correspondence

• If a function f: A → B is
  – 1-1 \((\text{if } f(a_1) = f(a_2), \text{ then } a_1 = a_2)\) AND
  – Onto B \((\text{for every } b \text{ in } B, \text{ there is an } a \text{ in } A \text{ with } f(a) = b)\)

Then f is a 1-1 correspondence.
Countable Sets

\[ \mathbb{N} \quad 1 \quad 2 \quad 3 \quad 4 \quad \ldots \]
\[ \mathbb{E} \quad 2 \quad 4 \quad 6 \quad 8 \quad \ldots \]

Take \( f(n) = 2n \) for \( n \) in \( \mathbb{N} \). It’s 1-1, and onto \( \mathbb{E} \), so a 1-1 correspondence!

**Def:** A set \( A \) is **countable** if
1. it is finite, or
2. there is a 1-1 correspondence from \( \mathbb{N} \) to \( A \).
\( \mathbb{Z} \) is countable

• \( \mathbb{Z} = \{0, -1, 1, -2, 2, -3, 3, \ldots\} \)

\begin{align*}
\text{f(1)} & \quad \text{f(2)} & \quad \text{f(3)} & \quad \text{f(4)} & \quad \text{f(5)} & \quad \text{f(6)} & \quad \text{f(7)} \\
\mathbb{N} & \quad 1 & \quad 2 & \quad 3 & \quad 4 & \quad 5 & \quad 6 & \quad 7
\end{align*}

• \( f(n) = \begin{cases} (n-1)/2 & \text{if } n \text{ is odd} \\ -(n/2) & \text{if } n \text{ is even} \end{cases} \)

• \( f \) is 1-1 and onto (1-1 correspondence)
Lemma: The set of strings over alphabet $\Sigma$ is countable

Proof: We can list the strings of $\Sigma^*$ in order of length, starting with the string of length 0 (i.e., lexicographical order). This provides the 1-1 correspondence with $\mathbb{N}$.

Example for $\{0,1\}^*$:

<table>
<thead>
<tr>
<th>$\mathbb{N}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(1)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(2)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(3)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(4)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(5)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(6)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(7)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Sigma^*$</td>
<td>$\varepsilon$</td>
<td>0</td>
<td>1</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>11</td>
<td>...</td>
</tr>
</tbody>
</table>


Lemma: The set of regular expressions $RE$ over alphabet $\Sigma$ is countable

Proof: We can list the regular expressions in order of length, starting with the strings of length 1, and this provides the 1-1 correspondence with $\mathbb{N}$.

Example for $\Sigma = \{0,1\}$:

$$
\begin{align*}
\mathbb{N} & \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad \ldots \\
f(1) & \quad f(2) \quad f(3) \quad f(4) \quad f(5) \quad f(6) \quad \ldots \\
RE & \quad \varepsilon \quad 0 \quad 1 \quad \emptyset \quad 0^* \quad 1^* \quad \ldots 
\end{align*}
$$
Size of Sets

- **Def:** Sets A and B have the same size if there is a 1-1 correspondence $f$ between A and B.

<table>
<thead>
<tr>
<th>Size</th>
<th>Example Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>1</td>
<td>${a}$, ${\frac{1}{4}}$, ${\pi}$, ${\diamond}$</td>
</tr>
<tr>
<td>2</td>
<td>${0,1}$, ${17.5, 34}$, ${\diamond, \diamond}$</td>
</tr>
</tbody>
</table>

- *Infinity*  
  - $\mathbb{N}$  
  - $\mathbb{Z}$  
  - $\mathbb{E}$  
  - $\mathbb{Q}$  
  - $\mathbb{R}$

- Are all infinite sets the same size? ?
It’s all about the diagonal!

DIAGONALIZATION
Th 4.17: The Real Numbers $\mathbb{R}$ are not countable.

Proof: $\mathbb{R}$ is not finite. We show that $\mathbb{R}$ is not countable by contradiction.

Suppose there was a 1-1 correspondence, $f : \mathbb{N} \rightarrow \mathbb{R}$.

<table>
<thead>
<tr>
<th>Number n in $\mathbb{N}$</th>
<th>Real number $f(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3.12159...</td>
</tr>
<tr>
<td>4</td>
<td>55.55555...</td>
</tr>
<tr>
<td>5</td>
<td>0.2345678...</td>
</tr>
<tr>
<td>6</td>
<td>0.5223344...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

We can construct a real number $x$ whose $ith$ digit $\neq ith$ digit of $f(i)$. (We always choose a digit between 1 and 8 (not 0 or 9). Why?) Then $x$ is a real number, but $x \neq f(n)$ for any $n$. This yields our contradiction.

This technique is called **Diagonalization**.
Lemma: The set \( L \) of all languages over alphabet \{0,1\} is uncountable.

Proof: \( L \) is not finite. Suppose that \( L \) were countable. Then there must be a 1-1 onto function \( f : \mathbb{N} \rightarrow L \).

We define a new language \( A \) over \{0,1\} and show that it is \( \neq f(n) \) for any \( n \), obtaining a contradiction to \( f \) being onto.

We use Diagonalization:

We define the elements of \( A \) so that they differ from each of the languages \( f(n) \) for any \( n \) in \( \mathbb{N} \), by choosing the \( i \)th element of \{0,1\}* to be in \( A \) IFF the \( i \)th element of \{0,1\}* is NOT in \( f(i) \).
Lemma, continued

The elements of any language $L$ can be listed as follows:

$$s_1, s_2, s_3, s_4, s_5, s_6, s_7, \ldots \ldots$$

$$\Sigma^* \quad \varepsilon \quad 0 \quad 1 \quad 00 \quad 01 \quad 10 \quad 11 \quad 000 \quad 001 \quad \ldots$$

$L \quad 1 \quad 10 \quad 11 \quad \ldots$

Now consider the languages given by $f$ in order:

<table>
<thead>
<tr>
<th>A</th>
<th>Number $n$ in $\mathbb{N}$</th>
<th>Language $f(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>1</td>
<td>${s_1, s_2, s_3, \ldots \ldots}$</td>
</tr>
<tr>
<td>$s_2$</td>
<td>2</td>
<td>${s_1, s_2, s_3, \ldots \ldots}$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>3</td>
<td>${s_1, s_2, s_3, \ldots \ldots}$</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

We define $A$ so that: the $i$th element $s_i$ of $\Sigma^*$ is in $L$ IFF

- the $i$th element $s_i$ of $\Sigma^*$ is NOT in the language $f(i)$.

Therefore, $A \neq f(n)$ for any $n$, and so $f$ is not onto, as assumed, CONTRADICTION.
Th. There are languages over \{0,1\} that are not regular.

Proof: The set of regular languages over \{0,1\} is countable, since, as previously shown, the set RE of regular expressions over \{0,1\} is countable.

We have just shown that the set \(L\) of all languages over \{0,1\} is not countable. Therefore, there are more languages in \(L\) than there are regular languages, so a non-regular language must exist. QED
Summary

• Pigeon Hole Principle applied to DFA’s:
  – DFA’s can only “remember” finitely far in the past, so their languages can be “pumped”

• Pumping Lemma can be used to show a given language is non-regular
  – You will be asked to do so in HW and exams!
  – There is a standard “script” for these proofs

• Diagonalization can be used to show there is a non-regular language
  – We’ll come back to Diagonalization later!
  – NOT on Exam 1

• Both proofs proceed by contradiction