Formal Statement of the Pumping Lemma

- **Pumping Lemma**: If $L$ is regular, then $\exists p$ such that $\forall s \in L$ with $|s| \geq p$, $\exists x, y, z$ with $s = xyz$ and:
  1. $|y| \geq 1$, and
  2. $|xy| \leq p$, and
  3. $xy^iz \in L \ \forall \ i \geq 0$

- Proof on board last time…(also in the textbook)
- Proved in 1961 by Bar-Hillel, Peries and Shamir
Pumping Lemma in Plain English

Let L be a regular language and let $p$ = “pumping length” = no. of states of a DFA accepting L

Then, any string $s$ in L of length $\geq p$ can be expressed as $s = xyz$ where:

- $y$ is not empty ($y$ is the cycle)
- $|xy| \leq p$ (cycle occurs within p state transitions), and
- any “pumped” string $xy^iz$ is also in L for all $i \geq 0$ (go through the cycle 0 or more times)

I liked the formal statement better…

That’s more like it…
Using The Pumping Lemma

**In-Class Examples**: Using the pumping lemma to show a language $L$ is *not regular*

1. Assume $L$ is regular.
2. Let $p$ be the pumping length given by the pumping lemma.
3. Choose cleverly an $s$ in $L$ of length at least $p$, such that
4. For *all ways* of decomposing $s$ into $xyz$, where $|xy| \leq p$ and $y$ is not null,
5. There is an $i \geq 0$ such that $xy^iz$ is not in $L$. 

Can’t wait to use it!
An alternate view: Think of it as a game between you and an opponent (JS):
1. You: Assume $L$ is regular
2. JS: Chooses some value $p$
3. You: Choose cleverly an $s$ in $L$ of length $\geq p$
4. JS: Breaks $s$ into some $xyz$, where $|xy| \leq p$ and $|y| \geq 1$,
5. You: Need to choose an $i \geq 0$ such that $xy^iz$ is not in $L$ (in order to win (the prize of non-regularity)!) 
(Note: Your $i$ should work for all possible $xyz$ that JS chooses, given your $s$)
Examples: Show the following are not regular

- $L_1 = \{0^n1^n \mid n \geq 0\}$ over the alphabet $\{0, 1\}$
- $L_2 = \{ww \mid w \text{ in } \{0, 1\}^*\}$
- $\text{PRIMES} = \{0^n \mid n \text{ is prime}\}$ over the alphabet $\{0\}$
- $L_3 = \{w \mid w \text{ contains an equal number of 0s and 1s}\}$ over the alphabet $\{0, 1\}$
- $\text{DISTINCT} = \{x_1#x_2#...#x_n \mid x_i \text{ in } 0^* \text{ and } x_i \neq x_j \text{ for } i \neq j\}$ (last two can be proved using closure properties of regular languages)
If $\{0^n1^n \mid n \geq 0\}$ is not Regular, what is it?

Enter...the world of Grammars (after midterm)
Basic Concepts (Chapter 0)

Sets

- Notation and Definitions
  - $A = \{x \mid \text{rule about } x\}$, $x \in A$, $A \subseteq B$, $A = B$
  - $\exists$ (“there exists”), $\forall$ (“for all”)

- Finite and Infinite Sets
  - Set of natural numbers $N$, integers $Z$, reals $R$ etc.
  - Empty set $\emptyset$

- Set operations: Know the definitions for proofs
  - Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
  - Intersection $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$
  - Complement $\overline{A} = \{x \mid x \notin A\}$
Basic Concepts (cont.)

- Set operations (cont.)
  - Power set of $A = \text{Pow}(A)$ or $2^A = \text{set of all subsets of } A$
    - E.g. $A = \{0,1\} \implies 2^A = \{\emptyset, \{0\}, \{1\}, \{0,1\}\}$
  - Cartesian Product $A \times B = \{(a,b) \mid a \in A \text{ and } b \in B\}$

- Functions:
  - $f: \text{Domain} \to \text{Range}$
    - $\text{Add}(x,y) = x + y \implies \text{Add}: \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$
    - Definitions of 1-1 and onto (bijection if both)
Strings

- Alphabet $\Sigma = \text{finite set of symbols, e.g. } \Sigma = \{0,1\}$
- String $w = \text{finite sequence of symbols } \in \Sigma$
  $w = w_1w_2...w_n$

- String properties: Know the definitions
  - Length of $w = |w|$ (if $w = w_1w_2...w_n$)
  - Empty string $= \epsilon$ (length of $\epsilon = 0$)
  - Substring of $w$
  - Reverse of $w = w^R = w_nw_{n-1}...w_1$
  - Concatenation of strings $x$ and $y$ (append $y$ to $x$)
  - $y^k = \text{concatenate } y \text{ to itself to get string of } k \text{ } y$’s
  - Lexicographical order $= \text{order based on length and dictionary order within equal length}$
Languages and Proof Techniques

♦ Language $L = \text{set of strings over an alphabet (i.e. } L \subseteq \Sigma^*)$
  ➤ E.g. $L = \{0^n1^n \mid n \geq 0\}$ over $\Sigma = \{0,1\}$
  ➤ E.g. $L = \{p \mid p \text{ is a syntactically correct C++ program}\}$ over $\Sigma =$ ASCII characters

♦ Proof Techniques: Look at lecture slides, handouts, and notes
  1. Proof by counterexample
  2. Proof by contradiction
  3. Proof of set equalities ($A = B$)
  4. Proof of “iff” ($X \iff Y$) statements (prove both $X \Rightarrow Y$ and $X \Leftarrow Y$)
  5. Proof by construction
  6. Proof by induction
  7. Pigeonhole principle
  8. Dovetailing to prove a set is countably infinite E.g. $\mathbb{Z}$ or $\mathbb{N} \times \mathbb{N}$
  9. Diagonalization to prove a set is uncountable E.g. $2^\mathbb{N}$ or Reals
Chapter 1 Review: Languages and Machines
Languages and Machines (Chapter 1)

- **Language** = set of strings over an alphabet
  - Empty language = language with no strings = ∅
  - Language containing only empty string = {ε}

- **DFAs**
  - Formal definition M = (Q, ∑, δ, q₀, F)
  - Set of states Q, alphabet ∑, start state q₀, accept (“final”) states F, transition function δ: Q × ∑ → Q
  - M recognizes language L(M) = {w | M accepts w}
  - In class examples:
    - E.g. DFA for L(M) = {w | w ends in 0}
    - E.g. DFA for L(M) = {w | w does not contain 00}
    - E.g. DFA for L(M) = {w | w contains an even # of 0’s}
    - Try: DFA for L(M) = {w | w contains an even # of 0’s and an odd number of 1’s}
Languages and Machines (cont.)

- Regular Language = language recognized by a DFA

- Regular operations: Union $\cup$, Concatenation $\circ$ and star $^*$
  - Know the definitions of $A \cup B$, $A \circ B$ and $A^*$
  - $\Sigma = \{0,1\} \rightarrow \Sigma^* = \{\varepsilon, 0, 1, 00, 01, \ldots\}$

- Regular languages are closed under the regular operations
  - Means: If $A$ and $B$ are regular languages, we can show $A \cup B$, $A \circ B$ and $A^*$ (and also $B^*$) are regular languages
  - Cartesian product construction for showing $A \cup B$ is regular by simulating DFAs for $A$ and $B$ in parallel

- Other related operations: $A \cap B$ and complement $\overline{A}$
  - Are regular languages closed under these operations?
NFAs, Regular expressions, and GNFAs

- **NFAs vs DFAs**
  - DFA: $\delta$(state,symbol) = next state
  - NFA: $\delta$(state,symbol or $\epsilon$) = set of next states
    - Features: Missing outgoing edges for one or more symbols, multiple outgoing edges for same symbol, $\epsilon$-edges
  - Definition of: NFA $N$ accepts a string $w \in \Sigma^*$
  - Definition of: NFA $N$ recognizes a language $L(N) \subseteq \Sigma^*$
  - E.g. NFA for $L = \{w \mid w = xla, x \in \Sigma^* \text{ and } a \in \Sigma\}$

- **Regular expressions**: Base cases $\epsilon$, $\emptyset$, $a \in \Sigma$, and $R_1 \cup R_2$, $R_1^*R_2$ or $R_1^*$

- **GNFAs = NFAs with edges labeled by regular expressions**
  - Used for converting NFAs/DFAs to regular expressions
Main Results and Proofs

- L is a Regular Language iff
  - L is recognized by a DFA iff
  - L is recognized by an NFA iff
  - L is recognized by a GNFA iff
  - L is described by a Regular Expression

Proofs:
- NFA $\rightarrow$ DFA: subset construction (1 DFA state=subset of NFA states)
- DFA $\rightarrow$ GNFA $\rightarrow$ Reg Exp: Repeat two steps:
  1. Collapse two parallel edges to one edge labeled $(a \cup b)$, and
  2. Replace edges through a state with a loop with one edge labeled $(ab^*c)$
- Reg Exp $\rightarrow$ NFA: combine NFAs for base cases with $\varepsilon$-transitions
Other Results

- Using NFAs to show that Regular Languages are closed under:
  - Regular operations $\cup$, $\circ$ and $*$

- Are Regular Languages closed under:
  - intersection?
  - complement?

- Are there other operations that regular languages are closed under?
What about the **reversal** operation?

What about the **idon’tcare** operation?

What about the **subset** operation?
Other Results

- Are Regular languages closed under:
  - reversal?
  - subset ($\subseteq$) ?
  - superset ($\supseteq$) ?
  - Prefix?
    \[ \text{Prefix}(L) = \{ w \mid w \in \Sigma^* \text{ and } wx \in L \text{ for some } x \in \Sigma^* \} \]
  - NoExtend?
    \[ \text{NoExtend}(L) = \{ w \mid w \in L \text{ but } wx \not\in L \text{ for all } x \in \Sigma^*-\{\varepsilon\} \} \]
Pumping Lemma

- *Pumping lemma in plain English (sort of):* If $L$ is regular, then there is a $p$ (= number of states of a DFA accepting $L$) such that any string $s$ in $L$ of length $\geq p$ can be expressed as $s = xyz$ where $y$ is not null ($y$ is the loop in the DFA), $|xy| \leq p$ (loop occurs within $p$ state transitions), and any “pumped” string $xy^iz$ is in $L$ for all $i \geq 0$ (go through the loop 0 or more times).

- *Pumping lemma in plain Logic:*

  $L$ regular $\Rightarrow \exists p$ s.t. $(\forall s \in L \text{ s.t. } |s| \geq p \ (\exists x, y, z \in \Sigma^* \text{ s.t. } (s = xyz)$
  and $(|y| \geq 1) \text{ and } (|xy| \leq p) \text{ and } (\forall i \geq 0, xy^iz \in L)))$

- Is the other direction $\Leftarrow$ also true?
  
  **No! See Problem 1.54 for a counterexample**
Proving Non-Regularity using the Pumping Lemma

- Proof by contradiction to show L is not regular
  1. Assume L is regular. Then L must satisfy the P. Lemma.
  2. Let p be the “pumping length”
  3. **Choose a long enough string** \( s \in L \) **such that** \( |s| \geq p \)
  4. Let \( x,y,z \) be strings such that \( s = xyz, |y| \geq 1, \) and \( |xy| \leq p \)
  5. **Pick an** \( i \geq 0 \) **such that** \( xy^iz \not\in L \) **(for all possible** \( x,y,z \) **as in 4)**

This contradicts the P. lemma. Therefore, L is not regular

- Examples: \( \{0^n1^n|n \geq 0\}, \{ww|w \in \Sigma^*\}, \{0^m|m \text{ prime}\}, \text{SUB} = \{x=y-z | x, y, z \text{ are binary numbers and } x \text{ is diff of } y \text{ and } z\} \)

- Can sometimes also use closure under \( \cap \) (and/or complement)
  - E.g. If \( L \cap B = L_1 \) where B is regular and \( L_1 \) is not regular, then L is also not regular (if L was regular, \( L_1 \) would be regular)
Some Applications of Regular Languages

✦ Pattern matching and searching:
   ✦ E.g. In Unix:
     ✦  ls *.c
     ✦  cp /myfriends/games/*.* /mydir/
     ✦  grep ‘Spock’ *trek.txt

✦ Compilers:
   ✦ id ::= letter (letter | digit)*
   ✦ int ::= digit digit*
   ✦ float ::= d d*.d*(ε|E d d*)
   ✦ The symbol | stands for “or” (= union)
Good luck on the midterm!

✦ You can bring one 8 1/2" x 11" review sheet (double-sided ok)

✦ The questions sheet will have space for answers. We will also bring extra blank sheets for those not so fond of brevity.

Don’t sweat it!

• Go through the homeworks, lecture slides, and examples in the text (Chapters 0 and 1 only)

• Do the practice midterm on the website and avoid being surprised!
Da Pumpin’ Lemma  
(adapted from a poem by Harry Mairson)

Any regulah language $L$ has a magic numba $p$
And any long-enuff word $s$ in $L$ has da followin’ propa’ty:
Amongst its first $p$ symbols issa segment u can find
Whoz repetition or omission leaves $s$ amongst its kind.

So if ya find a lango $L$ which fails dis acid test,
And some long word ya pump becomes distinct from all da rest,
By contradixion ya have shown $L$ is not
A regular homie, resilient to da pumpin’ u’ve wrought.

But if, on da otha’ hand, $s$ stays within its $L$,
Then eitha $L$ is regulah, or else ya chose not well.
For $s$ is $xyz$, where $y$ is not empty,
And $y$ must come befo’ da $p+1^{th}$ symbol u see.