Pumping Lemma Recap

✦ **Formal Statement of the 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. \( xy^iz \in L \forall i \geq 0 \), and
   2. \( |y| \geq 1 \), and
   3. \( |xy| \leq p \).

✦ Proof on board last time…(see also page 79 in 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

✧ 5 steps for a proof by contradiction:
1. Assume $L$ is regular. Then, $L$ satisfies the P. Lemma.
2. Let $p$ be the pumping length given by the P. 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 empty,
5. There is an $i \geq 0$ such that $xy^iz$ is not in $L$.

Can’t wait to use it…

Proving non-regularity as a Two-Person game

✦ An alternate view: Think of it as a game between you and an opponent (JC):
1. You: Assume $L$ is regular
2. JC: Chooses some value $p$
3. You: Choose cleverly an $s$ in $L$ of length $\geq p$
4. JC: Breaks $s$ into some $xyz$, where $|xy| \leq p$ and $y$ is not empty,
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 $xyz$ that JC chooses, given your $s$)
Proving Non-Regularity using the Pumping Lemma

✦ Examples: Show the following are not regular

✓ $L_1 = \{0^n1^n \mid n \geq 0\}$ over the alphabet \{0, 1\}
✓ $L_2 = \{w \mid w \text{ contains equal number of 0s and 1s}\}$ over the alphabet \{0, 1\}
✓ $L_3 = \{0^n1^m \mid n > m\}$ over the alphabet \{0, 1\}
✓ $ADD = \{x=y+z \mid x, y, z \text{ are binary numbers and } x \text{ is the sum of } y \text{ and } z\}$ over the alphabet \{0, 1, =, +\}
✓ $SQUARES = \{0^m \mid m = n^2 \text{ for some } n \geq 0\}$ over alphabet \{0\}
(see textbook for the proof)

Da Pumpin’ Lemma
(Orig. lyrics: Harry Mairson)

Any regular 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 is a segment you can find
Whoz repetition or omission leaves $s$ amongst its kind.

So if ya find a language $L$ which fails dis acid test,
And some long word ya pump becomes distinct from all da rest,
By contradiction you have shown dat language $L$ is not
A regular homie, resilient to the damage you’ve caused.

But if, upon the other hand, $s$ stays within its $L$,
Then either $L$ is regulah, or else you chose not well.
For $s$ is $xyz$, where $y$ cannot be empty,
And $y$ must come before da $p+j^{th}$ symbol is read.
If \( \{0^n1^n \mid n \geq 0\} \) is not Regular, what is it?

Enter... the world of Grammars
(after the Midterm)

CSE 322: Midterm Review

- **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 $2^A$ = set of all subsets of $A$
    - E.g. $A = \{0,1\} \Rightarrow 2^A = \{\emptyset, \{0\}, \{1\}, \{0,1\}\}$
  - Cartesian Product $A \times B = \{(a,b) \mid a \in A \text{ and } b \in B\}$

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

Strings

* Alphabet $\Sigma$ = finite set of symbols, e.g. $\Sigma = \{0,1\}$

* String $w$ = finite sequence of symbols $\in \Sigma$
  - $w = w_1w_2\ldots w_n$

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

✦ Language \( L \) = 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 \( \Sigma = \text{ASCII characters} \)

✦ 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 \( X \implies Y \) and \( X \iff Y \))
  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)

R. Rao, CSE 322

Chapter 1 Review: Languages and Machines

R. Rao, CSE 322
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, Σ, δ, q₀, F)
  ➔ Set of states Q, alphabet Σ, start state q₀, accept ("final")
  states F, transition function δ: Q × Σ → 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.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: \( \delta(\text{state}, \text{symbol}) = \text{next state} \)
  - NFA: \( \delta(\text{state}, \text{symbol} \text{ or } \varepsilon) = \text{set of next states} \)
  - Features: Missing outgoing edges for one or more symbols, multiple outgoing edges for same symbol, \( \varepsilon \)-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 \( L = \{ w \mid w = x1a, x \in \Sigma^* \text{ and } a \in \Sigma \} \)

- **Regular expressions**: Base cases \( \varepsilon, \emptyset, a \in \Sigma \), and \( R1 \cup R2, R1^*R2 \) or \( R1^* \)

- **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→DFA: subset construction (1 DFA state=subset of NFA states)
  - DFA→GNFA→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→NFA: combine NFAs for base cases with \( \varepsilon \)-transitions
Other Results

✦ Using NFAs to show that Regular Languages are closed under:
  ➔ Regular operations $\cup$, $\cdot$, and $\ast$

✦ Are Regular Languages closed under:
  ➔ intersection?
  ➔ complement (Exercise 1.10)?

✦ 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 (⊆)?
  - superset (⊇)?
  - Prefix?
    Prefix(L) = \{ w ∈ Σ* and wx ∈ L for some x ∈ Σ* \}
  - NoExtend?
    NoExtend(L) = \{ w ∈ L but wx ∉ L for all x ∈ Σ*-{ε} \}
    (see also Problem 1.32 in the text)

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 ≥ p can be expressed as s = xyz where y is not null (y is the loop in the DFA), |xy| ≤ p (loop occurs within p state transitions), and any “pumped” string xy^iz is in L for all i ≥ 0 (go through the loop 0 or more times).

- **Pumping lemma in plain Logic:**
  L regular ⇒ ∃ p s.t. (\∀ s ∈ L s.t. |s| ≥ p (∃ x,y,z ∈ Σ* s.t. (s = xyz) and (|y| ≥ 1) and (|xy| ≤ p) and (\∀ i ≥ 0, xy^iz ∈ L))

- Is the other direction ⇐ also true?
  No! See Problem 1.37 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 $xy^iz \notin L$ (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=n^2\}, ADD = \{x=y+z | x, y, z are binary numbers and x is sum of y and z\}

✦ Can sometimes also use closure under $\cap$ (and/or complement)
  \- E.g. If $L \cap B = L_1$, and $B$ is regular while $L_1$ 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* (e | E d d*)
  \- The symbol | stands for “or” (= union)
Good luck on the midterm on Wednesday!

✦ 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 of you who don’t believe in 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!