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: $A \cup B = \{x \mid x \in A \lor x \in B\}$
      • Intersection $A \cap B = \{x \mid x \in A \land x \in B\}$
      • 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. $A = \{0,1\} \Rightarrow 2^A = \{\emptyset, \{0\}, \{1\}, \{0,1\}\}$
  ➤ Cartesian Product $A \times B = \{(a,b) \mid a \in A \land b \in B\}$

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

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

✦ String $w = \text{finite sequence of symbols } \in \Sigma$
  $w = w_1w_2\ldots w_n$

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

Languages and Proof Techniques

✦ Language $L = \text{set of strings over an alphabet } (\text{i.e. } L \subseteq \Sigma^*)$
  $\Rightarrow$ E.g. $L = \{0^n1^n \mid n \geq 0\} \text{ over } \Sigma = \{0, 1\}$
  $\Rightarrow$ 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
  $\Rightarrow$ Proof by counterexample
  $\Rightarrow$ Proof by contradiction
  $\Rightarrow$ Proof of set equalities (A = B)
  $\Rightarrow$ Proof of “iff” (X$\iff$Y) statements (prove both X$\Rightarrow$Y and X$\Leftarrow$Y)
  $\Rightarrow$ Proof by construction
  $\Rightarrow$ Proof by induction
  $\Rightarrow$ Pigeonhole principle
  $\Rightarrow$ Dovetailing to prove a set is countably infinite (E.g. Z or N $\times$ N)
  $\Rightarrow$ Diagonalization to prove a set is uncountable (E.g. $2^N$ or Reals)
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, ∑, q0, δ, 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 ∪, Concatenation ° and star *
- Know the definitions of A ∪ B, A.B and A*
- ∑ = {0,1} → ∑* = {ε, 0, 1, 00, 01, …}

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

Other related operations: A ∩ B and complement A̅
- 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: $\delta(\text{state}, \text{symbol or } \epsilon) = \text{set of next states}$
    - 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 $L = \{w \mid w = x_1a, x \in \Sigma^* \text{ and } a \in \Sigma\}$

- **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 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)
  - Reg Exp$\rightarrow$NFA: combine NFAs for base cases with $\epsilon$-transitions
  - DFA$\rightarrow$GNFA$\rightarrow$Reg Exp: Collapse two parallel edges to one edge ($a \cup b$) and replace edges through a state with a loop with one edge ($ab^*c$)
Other Results

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

✦ Are Regular Languages closed under:
  ➤ intersection?
  ➤ complement (Exercise 1.10)?
  ➤ reversal (Problem 1.24)?
  ➤ subset $\subseteq$?
  ➤ superset $\supseteq$?
  ➤ no-prefix?  
  ➤ no-extend?

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 $\Rightarrow \exists p \text{ s.t. } (\forall s \in L \text{ s.t. } |s| \geq p \ (\exists x, y, z \in \Sigma^* \text{ s.t. } (s = xyz) \text{ and } (|y| \geq 1) \text{ and } (|xy| \leq p) \text{ and } (\forall i \geq 0, xy^iz \in L)))$
Proving Non-Regularity using the Pumping Lemma

- Proof by contradiction to show $L$ is not regular
  1. Assume $L$ is regular
  2. Let $p$ be some number (“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 $x,y,z$ as in 4)
     This contradicts the pump. lemma. Therefore, $L$ is not regular

- Typical Examples: $\{0^n1^n|n \geq 0\}$, $\{ww| w \in \sum^*\}$, $\{ww^R| w \in \sum^*\}$, $\{0^n|n \text{ is prime}\}$

- 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 not regular (if $L$ was regular, $L_1$ would have to 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* (\varepsilon | \varepsilon d d*)`
  - The symbol | stands for “or” (= union)