CSE 322: Midterm Review

- **Basic Concepts** (Chapter 0)
  - Sets
    - Notation and Definitions
      - A = {x | 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 = Pow(A) or 2^A = set of all subsets of A
    - E.g. A = {0,1} → $2^A = \{\emptyset, \{0\}, \{1\}, \{0,1\}\}$
  - Cartesian Product $A \times B = \{(a,b) \mid a \in A \text{ and } b \in B\}$
- Functions:
  - f: Domain $\rightarrow$ Range
    - Add(x,y) = x + y → Add: $Z \times Z \rightarrow Z$
  - Definitions of 1-1 and onto (bijection if both)

Strings

- Alphabet Σ = finite set of symbols, e.g. $\Sigma = \{0,1\}$
- String w = finite sequence of symbols $\in \Sigma$
  - $w = w_1 w_2 \ldots w_n$
- String properties: Know the definitions
  - Length of w = |w| ($|w| = n$ if $w = w_1 w_2 \ldots w_n$)
  - Empty string = ε (length of ε = 0)
  - Substring of w
  - Reverse of w = $w^R = w_n w_{n-1} \ldots w_1$
  - Concatenation of strings x and y (append y to x)
  - y^k = concatenate y to itself to get string of *k* y'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. $L = \{0^n 1^n \mid n \geq 0\}$ over $\Sigma = \{0,1\}$
  - E.g. L = {p | p is a syntactically correct C++ program} over Σ = ASCII characters
- Proof Techniques: Look at lecture slides, handouts, and notes
  - Proof by counterexample
  - Proof by contradiction
  - Proof of set equalities (A = B)
  - Proof of "iff" ($X \Leftrightarrow Y$) statements (prove both $X \Rightarrow Y$ and $X \Leftarrow Y$)
  - Proof by construction
  - Proof by induction
  - Pigeonhole principle
  - Dovetailing to prove a set is countably infinite E.g. Z or $N \times N$
  - 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 $M = (Q, \Sigma, q_0, \delta, F)$
  - Set of states Q, alphabet ∑, start state q_0, accept ("final") states F, transition function $\delta: Q \times \Sigma \rightarrow Q$
  - 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*
  - $\Sigma = \{0,1\} \rightarrow \Sigma^* = \{\varepsilon, 0, 1, 00, 01, \ldots\}$
- Regular languages are closed under the regular operations
  - Means: If A and B are regular languages, we can show A ∪ 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 $\overline{A}$
  - Are regular languages closed under these operations?

NFAs, Regular expressions, and GNFAs

- NFAs vs DFAs
  - DFA: δ(state,symbol) = next state
  - NFA: δ(state,symbol or ε) = set of next states
    - Features: Missing outgoing edges for one or more symbols, multiple outgoing edges for same symbol, ε-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 ε, ∅, $a \in \Sigma$, and $R1 \cup R2$, $R1 \circ R2$ or $R1^*$
- 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→DFA: subset construction (1 DFA state=subset of NFA states)
  - Reg Exp→NFA: combine NFAs for base cases with ε-transitions
  - DFA→GNFA→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 ≥ 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 $\Rightarrow \exists$p s.t. ($\forall s \in$ L s.t. |s| ≥ p ($\exists$x,y,z$\in \Sigma^*$ s.t. (s = xyz) and (|y| ≥ 1) and (|xy| ≤ p) 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 $xy^iz \notin L$ (for all x,y,z as in 4)

  This contradicts the pump. lemma. Therefore, L is not regular
- Typical Examples: $\{0^n1^n | n \geq 0\}$, $\{ww | w \in \Sigma^*\}$, $\{ww^R | w \in \Sigma^*\}$, $\{0^n | n \text{ is prime}\}$
- 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*(ε|E d d*)`
  - The symbol | stands for "or" (= union)