Chapter 1

Finite Automata ($Q-$states, $\Sigma-$alphabet, $\delta-$transitions, q_0-start, $F \subset Q-$accept). Language is **regular** if a finite automaton recognizes it. Two machines are equivalent if they recognize the same language.

- Derterministic (DFA) Restrict to one transition for each unique symbol.
- Nondeterministic (NFA) Every NFA has an equivalent DFA and any DFA is a valid NFA. Therefore a language is **regular** if and only if some NFA recognizes it.
- DFA to NFA Start at start state(s). Follow and write next possible states per symbol. Create new row for resulting states. Repeat until no new states. Should be 1 more state than the NFA.
- Generalized nondeterministic finite automaton (GNFA) Only one start and reject state. Transitions are regular expressions. Used to convert DFA to a RE.
- DFA to DE Add new start state and accept state. Transition start to old start and from old accept to new accept. Identify destination states from the state that will be removed. Identify all paths destination states have that go through the state that will be removed. Write new transitions excluding the removed state. Repeat.

Regular Languages are closed under **union, intersection, complement, concatenation, star (*)**.

Power Set is the set of all subsets of a language. Size of $P(A) = 2^{|A|}$.

Regular Expression. R is a RE if it is (1) a character in the alphabet associated with R. (2) the empty string. (3) the empty language. (4) two regular languages under union. (5) two regular languages under concatenation. (6) a regular language under star. **Order of Operations** is parenthesis, star, concatenation, union. A language described by a RE is **regular**.

Pumping Lemma for RL A string of length at least pumping-length can be broken up into xyz such that (1) xy^iz is in the language for any $i \geq 0$. (2) $|y| > 0$ (3) $|xy| \leq p$.

Finite Automata Theorems For a finite automata M with n states (1) $L(M)$ is non-empty if and only if M accepts a string of length less than n (2) $L(M)$ is infinite if and only if M accepts a string of length i where $n \leq i < 2n$. It is possible to create a FA that can determine if two FA are equivalent and taking a finite amount of time if they are equivalent.

Chapter 2

Context-free Grammar ($V-$variables (states), $\Sigma-$terminals (symbols), $R-$rules (transitions), $S-$start). Parse-trees show the path the CFG takes to output the string. Any language made by a CFG is a **context-free** language. A CFG is **ambiguous** if there is more than one way to generate a string (two parse trees). A CFL is **inherently ambiguous** if all grammars for the language are ambiguous. **Leftmost** deviation means the leftmost remaining variable is the one replaced; same for rightmost.

Context-free Languages are closed under **union, concatenation, star (*)**.

Chomsky Normal Form if every rule is of the form $A \rightarrow BC$ or $A \rightarrow a$. The start variable can have a ϵ. (1) Add new start variable with rule to old start variable. (2) Eliminate all ϵ rules. (3) Eliminate all unit rules. (3) Convert remaining to proper form by moving stuff around. Any CFL can be generated by a CFG in Chomsky normal form.

Pushdown Automata (PDA) Same setup as a FA, except the inclusion of a stack and the transitions that can pop or push something on the stack. A language is **context-free** if and only if some PDA recognizes it. Every regular language is context-free.

Pumping Lemma for CFL If L is a CFL, then there is a pumping-length where if a string in L is at least pumping-length then the string can be broken up into $uvxyz$ where (1) $uv^ixy^iz \in L$ for all $i \geq 0$. (2) $|xy| > 0$ (3) $|vxy| \geq p$