1 Formal Definitions

1.1 Deterministic Finite Automaton

A **deterministic finite automaton** is defined by the 5-tuple $\{Q, \Sigma, \delta, q_0, F\}$ where:

- Q is the finite set of **states** in the machine. e.g. $\{q_0, q_1, q_2, q_3\}$
- Σ is the finite set that represents the**alphabet** of the machine. e.g. $\{0, 1\}^*$
- $\delta : Q \times \Sigma \to Q$ is the **transition function** of the machine. Recall that $Q \times \Sigma \to Q$ means that the function accepts some pair (q_i, σ) where $q_i \in Q$ and $\sigma \in \Sigma$ and outputs some q_j where $q_j \in Q$.
- $q_0 \in Q$ is the **start state**
- $F \subseteq Q$ is the set of **accept states**

Note: If A is the set of all string that a machine M accepts, we say that A is the language of machine M, or rather $L(M) = A$. We also say that the machine M **recognizes** the language A. A language A is called a **regular language** if some finite automaton recognizes it.

1.2 The Regular Operations

Let A and B be two languages. We define the following regular operations:

- **Union**: $A \cup B = \{x : x \in A \text{ or } x \in B\}$
- **Concatenation**: $A \circ B = \{xy : x \in A \text{ and } y \in B\}$
- **Star**: $A^* = \{x_1 x_2 \cdots x_k : k \geq 0 \text{ and } x_i \in A\}$

Note: Regular languages and context free languages are closed under all of the regular operations.

1.3 Nondeterministic Finite Automaton

A **nondeterministic finite automaton** is defined by the 5-tuple $\{Q, \Sigma, \delta, q_0, F\}$ where all of the elements are the same except for δ, which is defined as:

- $\delta : Q \times \Sigma_\epsilon \to P(Q)$. This means that the function takes in a pair (q, σ) where $q \in Q$ and $\sigma \in \Sigma \cup \epsilon$ and outputs an element of $P(Q)$, which is the power set of Q. Recall that the power set of a set contains all of its possible subsets.

1.4 Regular Expressions

We say that R is a **regular expression** if R is:

- σ for some σ in alphabet Σ.
- ϵ
- $\emptyset$
- $R_1 \cup R_2$ where R_1 and R_2 are regular expressions
- $R_1 \circ R_2$ where R_1 and R_2 are regular expressions
- R^* where R is a regular expressions

1.5 Generalize Nondeterministic Finite Automaton

A **nondeterministic finite automaton** is defined by the 5-tuple $\{Q, \Sigma, \delta, q_{start}, q_{accept}\}$ where all of the elements are the same except for δ and q_{accept} which are defined as:

- $\delta : \{Q - q_{accept}\} \times \{Q - q_{start}\} \to R$. This means that the input to the transition function consists of every combination (q_i, q_j) where $q_i \in Q$ and $q_j \in Q$ but $q_i \neq q_{accept}$ and $q_j \neq q_{start}$. The output of the functions is the regular expression R which describes how to get from q_i to q_j
- q_{accept} is the *singular* accept state

Note: This is not a particularly useful entity. The one job it has is in the creation of regular expressions from DFAs. Because ever DFA has a matching GNFA, which can easily be turned into a regular expression by ripping away the states one by one.

1.6 Context Free Grammar

A context-free grammar is a 4-tuple (V, Σ, R, S), where:

- V is the finite set of **variables**
- Σ is the finite set, disjoint from V of **terminals**
- R is the finite set of rules that map from variables to strings of variables and terminals
- $S \in V$ is the start variable

Note: The **language of a grammar** is defined for grammar G as $\{w \in \Sigma^* : S \overset{*}{\Rightarrow} w\}$. The meaning of $S \overset{*}{\Rightarrow} w$ is that S **derives** w. This mean that starting from S one can create w using the rules of creation.

2 Pumping Lemma

2.1 Pumping Lemma for Regular Languages

If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

- For each $i \geq 0$, we have $xy^iz \in A$
- $|y| > 0$ thus the y segment is non-empty
- $|xy| \leq p$

Note: We use this lemma to prove a language non-regular. The proofs are by contradiction, so we assume a language regular, then show that there exists a string over pumping length that cannot be pumped

2.2 Pumping Lemma for Context Free Languages

If A is a context-free language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into five pieces $s = uvxyz$ satisfying the conditions:

- For each $i \geq 0$, we have $uv^ixy^iz \in A$
- $|vy| > 0$ thus the v and y segments can't both be empty
- $|vxy| \leq p$

3 Theorems

- Regular languages are closed under the regular operations:
  - Regular languages are closed under union:

    Use the Cartesian product method to 'track' two DFA's simultaneously and all the accept states still accept. This could be done even easier by creating an NFA that has epsilon transitions to the starts of the two DFAs
  - Regular languages are closed under concatenation:

    Put epsilon transitions from the accept states of the first machine to the start state of the second
  - Regular languages are closed under the star operations:

    Put epsilon transitions back from the accept states to the start state so the machine has to option to repeat. Also need to create new start state to except empty strings
- Regular languages are closed under complement:

  The set of accept states becomes all of the states that previously were not accept states.
- Regular languages are closed under intersection:

  Use the Cartesian product method to 'track' two DFA's simultaneously and only accept if it ends in accept states for both machines. Or we could have used the above proofs because $A \cap B = \overline{(\bar{A} \cup \bar{B})}$
- Regular languages are closed under difference:

  Use the Cartesian product machine and only accept if the state is in an accept for one, but not the other
- Every NFA has a equivalent DFA:

  Let the states of the DFA be $P(Q)$ from the NFA. Use these states instead of the 'lots of fingers' method
- A language is regular if and only if some regular expression describes it:

  we split the proof into two implications:
  - If a language is described by a regular expression, then it is regular: create NFAs for $\sigma \in \Sigma, \epsilon, \emptyset$, and then use concatenation, union, and star.
  - If a language is regular, then is is described by a regular expression: Use the GNFA method of forming a regex from a DFA

- Context free languages are closed under the regular operations:
  - Context free languages are closed under union:

    take the two languages and make a new language with rule $S \to S_1|S_2$

  - Context free languages are closed under concatenation:

    take the two languages and make a new language with rule $S \to S_1 S_2$

  - Context free languages are closed under star:

    take the language and make a new language with rule $S \to SS|\epsilon$

- Regular languages are closed under reversal:

  Flip all the transition functions, make start state the accept state, put a new start state with epsilon transitions to all the old accept states.

- Regular languages are closed under homomorphism and inverse homomorphism:

  Apply $h(\sigma)$ to ever element of the languages regular expression. The resulting regular expression describes $h(L)$. Inverse proof if very in depth.

- The intersection of a context free language and a regular language is context free.

  proof is out of bounds, but it is still true.

- If L is a CFL and R is a regular language, then $L - R$ is a CFL

  $L - R = L \cap \bar{R}$ and since $\bar{R}$ is regular we are done from the above proof.

4 Example Languages

4.1 Regular

- $\{w : w$ contains a 0, then an even number of 1s, then a final $0\}$
- $\{w : w$ is the empty string or ends with a $0\}$
- $\{w : w$ has an equal number of occurences of 01 and 10 as substrings$\}$

4.2 Non-Regular but Context Free

- $\{0^n 1^n : n \geq 0\}$
- $\{w : w$ has an equal number of 1s and 0s$\}$
- $\{0^i\, 0^j : i > j\}$
- $\{0^a 1^b\, 2^c\, 3^d : a + b = c + d\}$

4.3 Non-Regular and Non-Context Free

- $\{ww : w \in \{0,1\}^*\}$
- $\{a^n b^n c^n : n \geq 0\}$
- $\{a^i\, b^j\, c^k : 0 \leq i \leq j \leq k\}$
- $\{a^{2^n} : n \geq 0\}$
- $\{t_1 \# t_2 \# \cdot \# t_k : k \geq 2$ and $t_i \in \{a,b\}^*$ and $t_i = t_j$ for some $i \neq j\}$
- $\{a^n : n$ is prime$\}$
- $\{0^i 1^j : i \leq j^2\}$

5 Grab Bag

- let A be any set. Then $A \circ \emptyset = \emptyset \circ A = \emptyset$
- $\emptyset^* = \{\epsilon\}$
- $R \cup \emptyset = R$
- $R \circ \emptyset \neq R$
- $R \circ \epsilon = R$
- $R \cup \epsilon \neq R$

6 Turing Machines

6.1 Universal Turing Machine

$U =$ On input $\langle M, w \rangle$, where M is a TM and w is a string:

1. Simulate M on input w.
2. If M ever enters its accept state, *accept*; if M ever enters its reject state, *reject*.