8.1 Introduction

Many pattern recognition problems and their solutions can be expressed in terms of regular expressions and finite automata. We will begin with some definitions:

- An **alphabet** is a set of symbols.
- A **string** over the alphabet $A$ is a finite sequence of symbols from $A$. The **empty string**, $\epsilon$, is the string with no symbols.
- Strings $x$ and $y$ are **concatenated** to form the string $x \circ y = xy$. This concatenated string contains the sequence of symbols in $x$ followed by the sequence of symbols in $y$.
- If $xyz$ is a string, then $x$ is a **prefix**, $y$ is a **substring**, and $z$ is a **suffix** of $xyz$. In addition, $x$ and $y$ are also substrings of $xyz$ - using the terms “prefix” and “suffix” is just a more specific way to refer to them.
- The length of string $s$ is denoted as $|s|$. It is the total number of symbols that make up $x$. For example, the string $aaabb$ has length 5, so $|aaabb| = 5$. By definition $|\epsilon| = 0$, as you would expect.
- A **language** over $A$ is a specific set of strings over $A$.

This last definition - ie. languages over an alphabet - is very important in formalizing pattern recognition problems and used widely in the field of theoretical computer science. Below are some definitions related to languages.

- Let $L$ be a language. Define:

  \[
  L^{(0)} = \{ \epsilon \} \\
  L^i = L \circ L^{i-1}, \text{for } i > 1
  \]

  That second definition means that $L^i$ is the set of strings that result from taking every element in the whole language $L$ and concatenating it with each element of $L^{i-1}$ individually. The reason for this recursive definition will become clear when we define the Kleene Closure operation below.

- The **Kleene Closure** (also know as **Kleene Star**) of a language $L$ is denoted $L^*$. It is defined as:

  \[
  L^* = \bigcup_{i=0}^{\infty} L^i
  \]

  In words, the Kleene Closure of a set of strings is a new set containing all possible combinations of concatenations of strings from the original language. These concatenations can contain any number of strings from the original language in any order any number of times. For example, the Kleene Closure of an alphabet $A$, denoted $A^*$ is the set of all strings that can be composed using the symbols in alphabet. $A^*$ contains $\epsilon$, has $A$ as a subset, and by definition is infinite size (contains an infinite number of elements).
• The Positive Closure of a language \( L \) is denoted \( L^+ \). It is defined as:
\[
L^+ = \bigcup_{i=1}^{\infty} L^i
\]

Regular expressions and the languages they denote (known as regular sets) are useful concepts in computer science. In particular, they are effective descriptors of patterns.

## 8.2 Defining Regular Expressions

Let \( A \) be an alphabet. The regular expressions over \( A \) and the languages they denote are defined recursively as follows.

1. \( \emptyset \) is a regular expression, denoting the empty set
2. \( \epsilon \) is a regular expression and denotes the set containing the empty string (ie. \( \{ \epsilon \} \))
3. For each \( a \in A \), \( a \) is a regular expression and denotes \( \{a\} \)
4. If \( p \) and \( q \) are regular expressions denoting sets \( P \) and \( Q \) respectively then \( (p + q), pq \), and \( p^* \) are regular expressions that denote \( P \cup Q, P \circ Q, \) and \( P^* \) respectively.

In writing regular expression in practice, we can omit many parentheses if we assume \( (\ast) \), the Kleene Closure operator, has higher precedence than concatenation and union. Additionally, concatenation has higher precedence than the union operation \((+ / \cup)\). Additionally you may see the shorthand \( p^+ \) which is equivalent to \( pp^* \). Below are some examples of equivalent regular expressions:

By Order of Operations: \( ((0(1^*)) + 0) = 01^* + 0 \)

By Equivalent Notation: \( (0 + 1)^* = \{0, 1\}^* \)

And a third example, using words to describe the language:
\[ 1(0 + 1)^*1 + 1 = \{ \text{all strings beginning and ending with } 1 \} \]

A language is said to be regular if and only if it can be denoted by a regular expression. Two regular expressions are said to be equivalent if they denote the same set. Equivalence between regular expressions is denoted with the equals sign (\( = \)). For example \( (0 + 1)^* = (0^*1^*)^* \) tells us that the two regular expressions involved describe the same set of elements.

## 8.3 Deterministic Finite Automata

Now we will present the concept of a Deterministic Finite Automaton (DFA). A DFA is a machine automaton with a central device always in one of a finite set of states. It has an input tape which is scanned by a tape head from left to right. The automaton makes “moves” determined by the current state of control and the input symbol under the input head. Each move consists of entering a new state and shifting the input head one square (ie. one symbol) to the right. More formally:

Def: A Deterministic Finite Automaton is an \( M = (S, A, \delta, s_0, F) \) where...

1. \( S \) is a finite set of states of control
2. $A$ is the alphabet from which input symbols are chosen

3. $\delta$ is the **state transition function**
   \[ \delta : S \times (A \cup \{\epsilon\}) \rightarrow S \]

4. $s_0$ is the **initial state** of the finite control

5. $F \subseteq S$ is the set of **final or accepting** states

An **instantaneous description (ID)** of a DFA $M$ is a pair $(s, w)$ where $s \in S$ represents the current state and $w \in A^*$ represents the unused portion of the input string (the symbol under the tape head followed by the rest of the string to the right). An **initial ID** of $M$ is an $(s_0, w)$ for some $s \in A^*$. An **accepting ID** of $M$ is of the form $(s, w)$ where $s \in F$.

We represent moves of a DFA by a binary relation $\vdash$ on IDs. If $\delta(s, a) = s'$ then we write $(s, aw) \vdash (s', w)$ for all $w \in A^*$ where $a \in A \cup \{\epsilon\}$. If $a = \epsilon$ then the state transition may be made independently of the input symbol scanned. If $a \neq \epsilon$, the $a$ must appear on the next input square and the input head is shifted on square to the right. We use $\vdash^*$ to denote the reflexive and the transitive closure of $\vdash$. We say that $w$ is **accepted** by $M$ if $(s_0, w) \vdash^* (s, \epsilon)$ for some $s \in F$. To conclude and wrap up the concepts of DFAs and languages, we use $L(M)$ to denote the language accepted by $M$. That is:

$$L(M) = \{\text{set of strings accepted by } M\}$$

In the next set of notes we will expand on the idea of DFAs to construct non-deterministic finite machines which will have even more power to recognize strings and patterns.