CS 361: Lecture 1 Handout

Shikha Singh

Sets, Relations and Functions

A *set* is a collection of distinct objects, called *elements*. For example, $A = \{1, 2, 3\}$ is a set. A set A is a *subset* of a set B, written $A \subseteq B$, if every element of A is also an element of B. For two sets A and B, the *Cartesian product* is

$$A \times B = \{(a, b) \mid a \in A, b \in B\}.$$

The *empty set*, denoted $\emptyset$, is the unique set containing no elements, i.e., $\emptyset = \{\}$. The *power set* of a set A, denoted $\mathcal{P}(A)$, is the set of all subsets of A, that is,

$$\mathcal{P}(A) = \{B \mid B \subseteq A\}.$$

Relations and Functions. A *relation* R from a set A to a set B is a subset of $A \times B$. If $(a, b) \in R$, we say that a is *related* to b by R, written $a\, R\, b$.

A *function* f from a set A to a set B, written $f : A \to B$, is a relation such that for all $a \in A$, there is exactly one $b \in B$ such that $(a, b) \in f$. We typically write $(a, b) \in f$ as $f(a) = b$.

A function $f : A \to B$ is called *one-to-one* (or *injective*) if different elements of A map to different elements of B. Formally, f is infective if and only if for all $x, y \in A$ such that $f(x) = f(y)$, then $x = y$. A function $f : A \to B$ is called *onto* (or *surjective*) if every element of B has at least one pre-image in A. Formally, f is surjective if and only if for all $b \in B$, there exists $a \in A$ such that $f(a) = b$. A function is *bijective* or a bijection if it is both one-one and onto.

Finite and Countable Sets. A set S is finite if there is a bijection $f : S \to \{1, 2, \ldots, n\}$ for some $n \in \mathbb{N}$. We call n the *size* of S, denoted as $|S| = n$. S is *infinite* if no such bijection exists.

An set S is *countably infinite* if there exists a bijection $f : S \to \mathbb{N}$. We say a set S is *countable* if it is finite or infinite countably infinite. We say a set is *uncountable* if it is not countable. Example of countable sets include the natural numbers $\mathbb{N}$, the set of integers $\mathbb{Z}$, etc and an example of an uncountable set is the set of real numbers $\mathbb{R}$. We will revisit countability later in the course.

Strings and Languages

To represent the input data, we typically encode it in the form of a string from a finite alphabet. An *alphabet* Σ is defined as a **finite** set of symbols that we can use in our encoding. A *string* s is a finite (possibly empty) sequence of symbols from Σ, that is, $s = a_1 a_2 \cdots a_n$ where each $a_i \in \Sigma$.

A string over the binary alphabet $\Sigma = \{0, 1\}$ is called a *binary string*.

The *length* of a string x, denoted $|x|$ is the number of symbols contained in the string.

Consider two strings $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_m$. We say strings x and y are equal if and only if (1) $n = m$, and (2) $x_i = y_i$ for each $i = 1, \ldots, n$.

Operations on Strings. The basic operation on strings is *concatenation*. The concatenation xy of two strings x and y is the string xy, that is, x followed by y. For example, CS5361 is the concatenation of CS and 361.

Let x be a string. A string y is a *substring* of x if there exist strings u and v such that $x = uyv$. In particular, when $u = \varepsilon$, y is a *prefix* of x; and when $v = \varepsilon$ (so $x = uy$), y is called a *suffix* of x. For example, CS is a prefix of CS5400 and 5400 is a suffix of CS5400.

For a string x over alphabet Σ, the *reversal* of x, denoted by x^R, is defined by

$$x^R = \begin{cases} \varepsilon & \text{if } x = \varepsilon, \\ x_n x_{n-1} \cdots x_1 & \text{if } x = x_1 x_2 \cdots x_n, \quad \text{for } x_1, x_2, \ldots, x_n \in \Sigma. \end{cases}$$

Example. For strings x and y, $(xy)^R = y^R x^R$.

Proof. If $x = \varepsilon$, then $x^R = \varepsilon$ and hence $(xy)^R = y^R = y^R \varepsilon = y^R x^R$. If $y = \varepsilon$, then $y^R = \varepsilon$ and hence $(xy)^R = x^R = \varepsilon x^R = y^R x^R$.

Now, suppose $x = x_1 x_2 \cdots x_m$ and $y = y_1 y_2 \cdots y_n$, with $m, n \geq 1$. Then,

$$(xy)^R = (x_1 x_2 \cdots x_m y_1 y_2 \cdots y_n)^R = y_n y_{n-1} \cdots y_1 x_m x_{m-1} \cdots x_1 = y^R x^R.$$

□

Encoding Input and Output. Let Σ be an alphabet. We write Σ^* to denote the set of all strings over Σ. For example, $\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$ and $\{a\}^* = \{\varepsilon, a, aa, aaa, \ldots\}$.

Note that while Σ^* is an infinite set, each string in it has finite length.

Given a set A of objects, an *encoding* is an injective (one-to-one) function that maps A to Σ^*. This ensures that no two objects have the same encoding. In this course, we often restrict to an encoding over the binary alphabet.

Defining Computation. Once we encode the input and output data as finite strings, we can begin to define computation. Intuitively, to specify a well-defined computational problem, we need to provide an output for each input string. Thus, we will capture this specification as a function $f : \Sigma^* \to \Sigma^*$.

For example, the function reverse$(x) = x^R$ maps each string $x \in \Sigma^*$ to its reverse string. We say a computation or algorithm "solves" a function f if its output on x is the same as $f(x)$ for all input strings x.

In this course, we further simplify and only consider *decision problems*, which have a "yes" or "no" answer. In particular, a decision problem is a function $f : \Sigma^* \to \{0, 1\}$. Examples of decision problems include "does this graph have a clique of size k?" or "given a number, is it prime?".

Languages to Represent Computation. A language L over alphabet Σ is just a subset of Σ^*, that is, $L \subseteq \Sigma^*$. Intuitively, a language is just a set of words over an alphabet.

> **Example.** For $\Sigma = \{0, 1\}$, example of languages include $L_1 = \emptyset$, $L_2 = \Sigma^*$, $L_3 = \{1, 01, 001, 0001, \ldots, \}$, etc.

There is a one-to-one mapping between a decision problem over an input encoded over alphabet Σ and a language L over Σ. In particular, we say L is language for the decision problem f if $f(x) = 1$ if and only if $x \in L$.

In this course, we will study computation through the terminology of languages due to the influence of computational linguist Noam Chomsky on the field.