COMS W3261 : Computability review

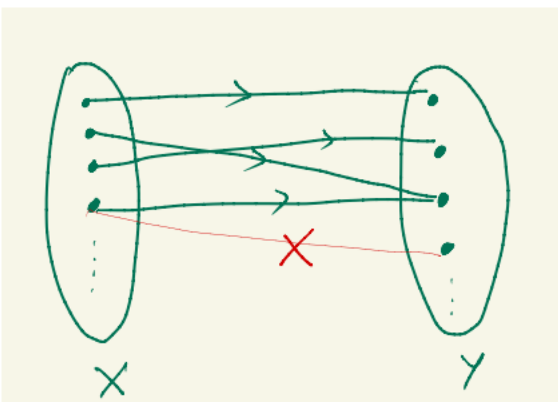
Sayak Chakrabarti

1 Relations and functions

Definition 1 (Relations)**.** A relation $R : X \to Y$ is given as a subset of $X \times Y = \{(x, y)|x \in X, y \in Y\}$.

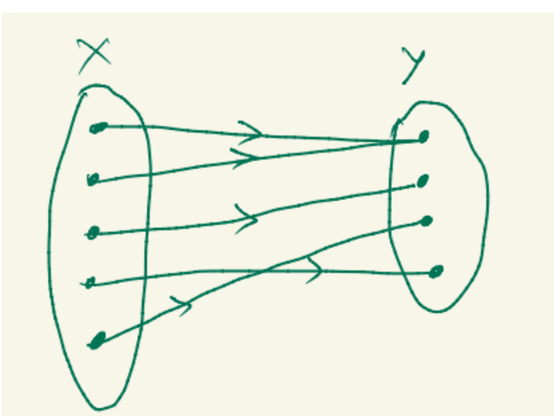
For example, we define the relation $R : \mathbb{N} \to \mathbb{N}$, $(x, y) \in R$ if $x - y$ is even. Some elements of this relation are $\{(1, 3), (2, 8), (1, 5), (3, 5) \dots\}$.

Definition 2 (Function)**.** A function $f : X \to Y$ is an assignment of an element of Y to each unique element of X. We formally write it as $f(x) = y$.



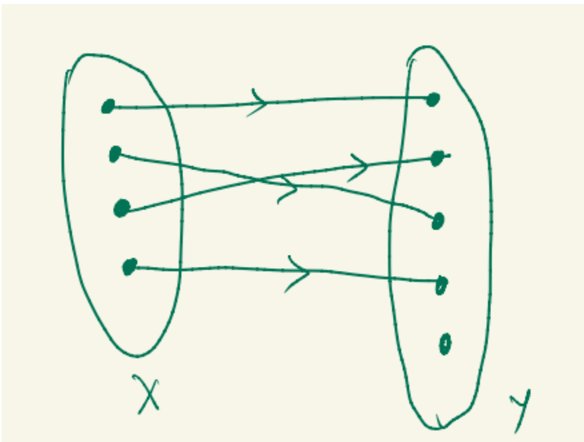
Notice that unlike relations where both $(x, y_1), (x, y_2)$ can be included, functions have a unique mapping from x to y, i.e. $f(x) = y_1$ and $f(x) = y_2$ implies $y_1 = y_2$. However, we can have $f(x_1) = f(x_2) = y$ for distinct x_1, x_2's.

Definition 3 (Surjective or onto)**.** A function $f : X \to Y$ is said to be surjective or onto if every element in Y is the mapping of some element in X.



The size of the sets, denoted by the cardinality $|\cdot|$, can be compared as $|X| \geq |Y|$.

Definition 4 (Injective or one-one)**.** A function $f : X \to Y$ is said to be injective or one-one if every element in X is mapped to a unique element in Y.



We have the cardinalities as $|X| \leq |Y|$.

We make some comments about cardinality of some sets as follows:

- $|\mathbb{N}| = |2\mathbb{N}|$, where $2\mathbb{N}$ denotes the set of even natural numbers. We prove this as constructing two functions as follows:
    1. Define $f : 2\mathbb{N} \to \mathbb{N}$ as $f(a) = a$, which is a one-one function. This implies that $|\mathbb{N}| \geq |2\mathbb{N}|$.
    2. Define the function $g : \mathbb{N} \to 2\mathbb{N}$ as $g(a) = 2a$, which is a one-one function, implying $|\mathbb{N}| \leq |2\mathbb{N}|$.
- The set of positive rational numbers $|\mathbb{Q}^+| = |\mathbb{N}|$.
    1. We know that every positive rational number can be written as $\frac{p}{q}$, where $p, q \in \mathbb{N}$. Define $f : \mathbb{Q} \to \mathbb{N}$ given as $f(p/q) = 2^p 3^q$. This is a one-one function, implying $|\mathbb{Q}^+| \leq |\mathbb{N}|$.

Exercise 1. Prove that $|\mathbb{Q}| = |\mathbb{N}|$.

Definition 5. A set S is said to be countable if there exists an injective function $f : S \to \mathbb{N}$.

2 Computability

2.1 Configuration graph

We define the *configuration graph* of a DFA to study the states possible after transitions. Let us assume we have a DFA given as (Q, δ, Σ, F). We construct a directed graph G consisting of vertices corresponding to each state $q_i \in Q$. There exists an edge from q_i and q_j if for some $a \in \Sigma$, we have $\delta(q_i, a) = q_j$. We label this edge as a. There can be several labels to some edge. Now notice that in the graph, there exists a path from q_i to q_ℓ if there exists a sequence of transformations from $q_i - > q_{i_1} - > q_{i_2} - > \ldots - > q_\ell$ in the original DFA. Using the labels on the edges, the sequence of strings which lead to these transformations can be found.

Exercise 2. Prove that if the starting state is q_0 and the final state is q_f, a path from q_0 to q_f exists if and only if there is some sequence of strings that are accepted by the DFA.

2.2 Dovetailing

Let us consider an example where we are interested in checking if a string from a given set of infinitely many strings is accepted/rejected by a Turing machine M. Let us assume the set of strings is $S = \{w_1, w_2, \ldots\}$.

We first try a trivial approach, as follows:

- Run M on w_1 until it halts (accepts/rejects).
- Move on to the nect string w_2.

  $\vdots$

However, the problem arises if M never halts on w_1 and we will never be able to move on to the next string. In order to circumvent this problem, we consider the *dovetailing method* as follows by creating a new TM N:

- For step $i = 1, 2, 3, \ldots$:
  - Run M on $w_1, w_2, \ldots, w_i$ for i steps each.
  - If for any w_k for $k \in [i]$[1], $M(w_k)$ halts in less than or equal to i steps, return whatever $M(w_k)$ had returned.

The TM runs as

1	-	$M(w_1)$ for 1 step.			
2	-	$M(w_1)$ for 2 steps,	$M(w_2)$ for 2 steps.		
3	-	$M(w_1)$ for 3 steps,	$M(w_2)$ for 3 steps,	$M(w_3)$ for 3 steps.	
4	-	$M(w_1)$ for 4 steps,	$M(w_2)$ for 4 steps,	$M(w_3)$ for 4 steps,	$M(w_4)$ for 4 steps.

Exercise 3. Prove the following:

1. Every string is considered to run for any finite amount of steps.
2. If M halts on some string w_j, $N(w_j)$ halts after a finitely many steps.

[1] $[i]$ refers to the set $\{1, 2, \ldots, i\}$

3 Reductions

We prove some examples of reductions among languages. We say that a language L reduces to a language P, $L \leq P$, if we can use the Turing machine of P to construct one for L.

The Turing machines have standard notations as described in class.

Example 1. $HALT_{TM} \leq A_{TM}$.

Let us assume that there is a decider N for A_{TM}, and we construct a TM for $HALT_{TM}$ using N as follows:

1. We are given input $\langle M, w \rangle$ to $HALT_{TM}$.
2. If N accepts accepts $\langle M, w \rangle$, return accept.
3. Create a new TM N' as follows:
   (a) On input x, run M on x.
   (b) If M accepts x, return reject.
   (c) If M rejects x, return accept.
4. Run N on $\langle N', w \rangle$. If N' accepts, return accept.

Exercise 4. Prove that

- $(M, w) \in HALT_{TM}$ if either M halts and accepts w, or M halts and rejects w.
- If M rejects w, then N' accepts w.

Example 2. Prove that $L = \{\langle M, D \rangle | M$ is a TM, D is a DFA, $L(M) = L(D)\}$, is undecidable.

We will reduce an undecidable language A_{TM} to L, i.e. considering a decider N for L, we will construct a decider for A_{TM}. Now, if L were decidable it will imply that we will be able to solve A_{TM}, which will lead to a contradiction as we already know that A_{TM} is undecidable.

The main idea is, given an input $\langle M, w \rangle$, create DFA D_w which accepts only the string w, and a TM M_w which accepts only the string w if and only if M accepts w, no string otherwise.

The construction for A_{TM} is as follows:

1. We are given with an input $\langle M, w \rangle$ for A_{TM}.
2. Construct DFA D_w such that $L(D_w) = \{w\}$.
3. Construct M_w as follows:

(a) On input x such that $x \neq w$, $M_w(x)$ rejects.

(b) If $x = w$, then run w on M. If $M(w)$ accepts, then accept, and if $M(w)$ rejects, then reject.

4. Run N on $\langle M_w, D_w \rangle$.

   (a) If N accepts, then accept.

   (b) If N rejects, then reject.

Notice that since $L(D_w) = \{w\}$, $L(M_w) = L(D_w)$ if and only if M accepts w. Now, by assumption, N can decide if $L(M_w) = L(D_w)$ and therefore $\langle M, w \rangle \in A_{TM}$. However, if $\langle M, w \rangle \notin A_{TM}$ then $L(M_w) = \phi$, implying that $L(M_w) \neq L(D_w)$. In this way A_{TM} can be simulated, and if N is decidable then so is A_{TM}, a contradiction.