1 Overview

This week we will discuss Turing machines and the Church-Turing thesis.

1.1 Turing Machines

Formally, a Turing machines is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where

- $Q$ is the set of states.
- $\Sigma$ is the input alphabet.
- $\Gamma$ is the tape alphabet. We need $\Sigma \subseteq \Gamma$ and we have a special blank symbol $\sqcup \in \Gamma - \Sigma$.
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function.
- $q_0, q_{\text{accept}}, q_{\text{reject}} \in Q$ are the start state, accept state, and reject state respectively.

A Turing machine configuration is a string $uqv \in \Gamma^*Q\Gamma^*$ that encodes (i) the state $q$ of $M$, (ii) the tape contents $uv$, and (iii) the location of the head within the tape. Note that we ignore trailing blanks so $uqv\sqcup$ and $uqv$ are considered equivalent configurations.

Configurations are updated as follows.

- $uq\sigma v \rightarrow u\sigma'q'v$ if $\delta(q, \sigma) = (q', \sigma', R)$.
- $u\tau q \sigma v \rightarrow u\tau \sigma v'$ if $\delta(q, \sigma) = (q', \sigma', L)$.
- $q \sigma v \rightarrow q' \sigma v'$ if $\delta(q, \sigma) = (q', \sigma', L)$.

What about a configuration of the form $uq$?

A computation starts in the configuration $q_0x$, where $x$ is the input. It accepts if it enters a configuration of the form $uq_{\text{accept}}v$. It rejects if it enters a configuration of the form $uq_{\text{reject}}v$. It is possible that $M$ never accepts and never rejects; if this happens, we say that $M$ does not halt.

We say that a language $L \subseteq \Sigma^*$ is recognized by a Turing machine $M$ if $M$ accepts every $x \in L$. If we also have that $M$ rejects every $x \in \Sigma^* - L$, then we say that $L$ is decided by $M$. 
1.2 Church-Turing thesis

The Church-Turing thesis is the assertion that Turing machines capture our intuitive notion of computability. This is not a mathematical statement! We cannot prove it, but we have good reasons to believe it nonetheless. There are many equivalent models of computation. (e.g. General grammars (PS6), multitape TMs, recursive functions, \(\lambda\)-calculus.)

2 Exercises

Exercise 2.1. Construct a TM that decides the language \(L = \{a^n b^m c^{nm} : n, m \geq 0\}\).

Exercise 2.2. Convert the following grammar to CNF.

\[
S \rightarrow SS | [S] | A \\
A \rightarrow aAa | \varepsilon
\]

Use the algorithm from class to parse \([[aa][[]]]\).

Exercise 2.3. Imagine a special Turing Machine with a 2-dimensional tape. So instead of the usual linear tape this TM has an infinite upper-right quadrant where the head starts at position \((0, 0)\). Upon reading each symbol this 2D Tape TM can choose to move left, right, up, or down (but of course cannot move off the edge). The input string \(w\) will start along the bottom row of the 2D tape, from positions \((0, 0)\) to \((n, 0)\). Show that this 2D Tape TM is no more powerful than a standard TM by simulating a 2D Tape TM with a normal one.

Exercise 2.4. Show that we can assume that a TM always halts with an empty tape. That is, show how to convert a TM \(M\) into a TM \(M'\) with \(L(M) = L(M')\) where the configuration of \(M'\) when it accepts or rejects is either \(q_{\text{accept}}\) or \(q_{\text{reject}}\) and not \(uq_{\text{accept}}v\) or \(uq_{\text{reject}}v\) with \(uv \notin \{\|$\}^*\).