Challenge Problems 3

Here's one final set of challenge problems on topics from across the quarter. As usual, we haven't put together solutions and instead want you to play around with these problems if you're up for a bit of a challenge. Hop on the Struggle Bus, have fun, and see what you can find!

Propositional Completeness

On Problem Set Two, you proved that all propositional formulas can be rewritten to just use the $\to$ and $\bot$ connectives. Is the same true of $\leftrightarrow$ and $\bot$? If so, prove it. If not, disprove it.

The Barberville Sequence Revisited

The ***multiplicative Barberville sequence*** is an inductively-defined sequence of values. The first multiplicative Barberville number, denoted M_0, is zero. Then, for all $n \in \mathbb{N}$, we define M_{n+1} as follows: B_{n+1} is defined as the smallest natural number that can't be written as $B_i \cdot B_j$, where $0 \leq i < j \leq n$. In other words, B_{n+1} is the smallest natural number that can't be written as the product of two previous Barberville numbers.

Which numbers are multiplicative Barberville numbers? Can you prove it?

Splitting Regular Languages

Let L be an infinite regular language (that is, a regular language containing infinitely many strings). Prove that there are infinite regular languages L_1 and L_2 where $L_1 \cup L_2 = L$. As a hint, think about what happens if you have a DFA D for a language L and run a string $w \in L$ through D when $|w|$ is greater than the number of states in D.

Context-Free Languages

Here's one of my favorite "challenging" context-free grammar design problems. Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and consider this language:

$$L = \{\ wx \in \Sigma^* \mid w \neq x, \text{ but } |w| = |x|\ \}$$

Design a CFG for L.

Turing Machines

For any $k \in \mathbb{N}$, let's consider the following language $A_{TM}{}^{(k)}$:

$$A_{TM}{}^{(k)} = \{ \langle M, w \rangle \mid M \text{ is a TM with at most } k \text{ states and } M \text{ accepts } w \}$$

It's known – though the proof is way beyond the scope of this class – that $A_{TM}{}^{(2)}$ is decidable, even though A_{TM} is not.

Prove that there is some natural number k where $A_{TM}{}^{(k')}$ is undecidable for all $k' \geq k$.

R and RE Languages

A ***very sparse language*** is a language L over an alphabet Σ that contains exactly one string of each length. That is, if L is a very sparse language, it contains the empty string (the only string of length 0), exactly one string of length one, exactly one string of length two, etc.

Prove that if L is a very sparse language and $L \in \mathbf{RE}$, then $L \in \mathbf{R}$.

Time Complexity

Let M be a Turing machine. Let's define one "step" of a TM to be the act of reading a character, writing back a character, moving the tape head, and changing state.

Suppose that M is a TM with the following property: M never makes more than k steps when run on any input, where k is some fixed natural number. Prove that $\mathscr{L}(M)$ is regular.

NP-Hardness

There's an efficient algorithm for converting a DFA for a language L into a minimum-state DFA for the language L. It works by using a variation on the Myhill-Nerode theorem.

It turns out that NFA minimization is a really hard problem. In fact, if you can design an efficient algorithm that takes as input an NFA and outputs an NFA with the same language and the minimum possible number of states, then you've proven $\mathbf{P} = \mathbf{NP}$.

Prove this. As a hint, show that for any 3-CNF formula φ, you can construct a polynomially-sized NFA that accepts all *unsatisfying* assignments to φ. Once you've done that, see if you can come up with a polynomial-time algorithm for solving 3SAT.