Theory of Computation Assignment 3

Due Date: November 7, 2023

Write clear and concise solutions. It is fine to discuss with others, but your solutions *must* be in your own words that you have fully understood. All problems carry equal marks. Upload solutions on moodle. Note: only handwritten solutions are acceptable.

1. If L is a context-free language show that there is a pushdown automaton M that accepts L by final state such that M has just two states and doesn't make ϵ-moves.

2. If L is a context-free language show that $L = L(M)$ for a pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ such that if (p, γ) is in $\delta(q, a, X)$ then $|\gamma| \leq 2$. In other words, the top stack symbol is replaced by a string of length at most 2.

3. Show that $\{0^n 1^n \mid n \geq 0\} \cup \{0^n 1^{2n} \mid n \geq 0\}$ cannot be accepted by a deterministic pushdown automaton.

4. Construct a pushdown automaton that accepts $w \in \{a, b\}^*$ such that w has twice as many a's as b's.

5. Prove that the set of primes encoded in binary is not a context-free language.

6. Let $G = (V, T, P, S)$ be a context-free grammar. Give an algorithm that takes as input two string $\alpha, \beta \in (V \cup T)^*$ and checks if $\alpha \underset{G}{\Rightarrow} \beta$. What is the running time in terms of $|\alpha| + |\beta|$? Assuming G is in CNF, can the CYK algorithm be used to make it efficient?

7. Let $G = (V, T, P, S)$ be a CFG in CNF. Associate a cost function $c : P \to \mathbb{R}^+$ to each production. Modify the CYK algorithm that takes as input a string $w \in T^*$ and computes a minimum cost parse tree for w if $w \in L(G)$. Analyze your algorithm's running time.

8. Describe a Turing machine M that converts 0^n, given as input, to the binary encoding $\text{bin}(n)$ for any $n \in \mathbb{N}$.

9. Prove that a language L is recursive iff both L and its complement $\overline{L}$ are recursively enumerable.

10. A *queue automaton* has access to a queue as storage instead of a stack. Formally, we can describe it as a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Here, Σ is the input alphabet, Γ (containing Σ) is the queue alphabet, and $Z_0 \in \Gamma$.

    - Like a PDA, a queue automaton has a one-way input tape from which it reads the input letter by letter. It can also make ϵ-moves (without advancing the input head to read the next letter). The queue initially contains just Z_0.
    - Given (q, a, X), where $q \in Q$ is the current state, $a \in \Sigma \cup \{\epsilon\}$ and X is the head of the queue, the automaton enters a new state q' removes X from the queue and can insert one or more symbols at the end of the queue. So the transition function can be described as $\delta : Q \times \Sigma \times \Gamma \to Q \times \Gamma^*$, where $\delta(q, a, X) = (p, \gamma)$ means that the machine in state q, for $a \in \Sigma \cup \{\epsilon\}$ and head of the queue X enters state p, removes X from the head of the queue, and inserts γ as the tail of the queue. The automaton accepts by final state.

    Prove that queue automata are computationally equivalent to Turing machines by showing that you can simulate a one-tape deterministic Turing machine on a queue automaton.