HW4

Graded

Student

Vidit Dharmendra Pokharna

Total Points

38 / 40 pts

Question 1

1 **8** / 8 pts

✔ **– 0 pts** Correct

– 2 pts Did not generalize the solution in terms of k and k+1

– 3 pts Did not show how to simulate moving 1 step left, solution shows only moving 1 step right

– 6 pts Did not show what combination of moves in broken TM can simulate regular TM

– 6 pts Major error

– 8 pts Missing

Question 2

2 **8** / 8 pts

✔ **– 0 pts** Correct

– 1 pt Minor Error

– 4 pts Major Error

– 8 pts No answer

Question 3

3 **8** / 8 pts

✔ **– 0 pts** Correct

– 1 pt Clarity in writing needed

– 2 pts Minor Error

– 4 pts Major Error

– 8 pts Missing/Incorrect

Question 4

4 **8** / 8 pts

✔ **– 0 pts** Correct

– 1 pt Minor Error

– 2 pts Minor errors

– 4 pts Major Error

– 8 pts no submission

Question 5

5 **6** / 8 pts

– 0 pts Correct

✔ **– 2 pts** points left unjustified (see comments)

– 8 pts no submission

+ 1 pt great answer (bonus point)

https://www.youtube.com/watch?v=SSEjPuR11_k

1 but why

2 but why

Question assigned to the following page: 1

CS 4510 Automata and Complexity 2/23/2024

Homework 4: Counting and Computation

Vidit D. Pokharna *Due:3/06/2024*

You should submit a typeset or *neatly* written pdf on Gradescope. The grading TA should not have to struggle to read what you've written; if your handwriting is hard to decipher, you will be asked to typeset your future assignments. Four bonus points if you use LaTeX, and our template. You may collaborate with other students, but any written work should be your own.

1. (8 points) Oops. We took a Turing machine and dropped it and the gears are broken or slipping or something. Instead of moving one unit left, or one unit right. It can now move only k units left or $k+1$ units right. Prove that for all $k > 1$, this broken Turing machine is still Turing-complete. Show your work. (Hint: As a warmup, think about a 3R, 2L machine).

   To prove that a modified Turing machine, which we'll call TM', that can only move k units left or $k+1$ units right is still Turing-complete, we need to demonstrate that it can simulate a standard Turing machine.

   Considering how to simulate the single-step movements of a standard TM using the capabilities of TM', we can use the case of a 3R, 2L machine as an example, where $k = 2$, but the same logic can be applied for any $k > 1$.

   A standard TM moves one unit to the right (1R). TM', on the other hand, can achieve this by moving $k+1$ units to the right and then k units to the left:

   - Move $k+1$ units to the right: In the specific case of 3R (when $k = 2$), the machine moves three units to the right
   - Then, move k units to the left: Following the 3R, the machine would move two units to the left (2L)

   The net effect is a movement of one unit to the right, successfully simulating a standard TM's right move.

   Simulating a left move is slightly more complex due to the nature of the TM' constraints. The idea is to use a series of k left and $k+1$ right movements to net a single unit of movement to the left:

   - Move k units to the left
   - Move $k+1$ units to the right
   - Repeat these two steps $k-1$ times, resulting in $k-1$ units to the right
   - Move k units to the left, resulting in 1 unit to the left

   In the specific case of 2L, 3R machine:

Question assigned to the following page: 1

- Move two units left
- Move three units right
- Move two units left

The net effect is a movement of one unit to the left, successfully simulating a standard TM's left move.

By setting up our TM' with a series of states and transitions that implement these movement patterns, we can ensure that any single move (either left or right) of the standard TM can be replicated by TM'.

Question assigned to the following page: 2

2. (8 points) Prove that $\mathscr{L}(NFA)$ is countable. Note that this is not the set of NFAs, but the set of languages which have NFA's to decide them. Each language is of course, countably infinite. The question is asking about the number of languages.

Since Q and Σ are finite, there are only finitely many possible mappings for δ for each pair of Q and Σ. Since there are also only finitely many options for q_0 and F, each component of the NFA can be represented by a finite string of characters from a fixed, finite alphabet (including all the symbols of Σ and additional symbols to represent states, transitions, and the structure of the 5-tuple).

Since the set of all finite strings over a finite alphabet is countable (by the same reasoning that the set of all natural numbers, words, or binary strings is countable), the set of all NFAs, being representable by such strings, is also countable.

Given that the set of all NFAs is countable, we can construct a list (or sequence) that includes every possible NFA. Since each NFA corresponds to a unique language (though note, different NFAs can recognize the same language), the list of all NFAs maps to a list of languages in $\mathscr{L}(NFA)$.

While this mapping from NFAs to languages is surjective (each NFA recognizes a language), it is not necessarily injective (different NFAs can recognize the same language). This means there could be many NFAs corresponding to the same language.

Regardless, since we start with a countable set (the NFAs), mapping to the languages they recognize cannot result in more than a countable set of languages, because each language in $\mathscr{L}(NFA)$ is represented at least once in the mapping from our countable list of NFAs. In mathematical terms, the cardinality of $\mathscr{L}(NFA)$ cannot exceed the cardinality of the set of all NFAs.

Since both the set of all NFAs is countable and the set of languages they recognize is at most as large, $\mathscr{L}(NFA)$ is countable. This concludes the proof.

Question assigned to the following page: 3

3. (8 points) Recall the definition of a DIA from the first homework. Prove that the set of all DIAs is uncountable. Lets clarify the definition of a DIA. Q is countably infinite. $F \subseteq Q$ is finite or countably infinite. Σ is finite. δ is defined appropriately for countably infinite Q and finite Σ.

To prove that the set of all DIAs is uncountable, we can focus on the transition function δ, since the other components of the DIA have fixed sizes or cardinalities (except for Q, which is always countably infinite).

Since Q is countably infinite and Σ is finite, the set of all possible inputs to δ is also countably infinite. However, for each input pair $(q, \sigma) \in Q \times \Sigma$, there are countably infinitely many possible states in Q that could be the output of the transition function $\delta(q, \sigma)$.

The set of all possible transition functions δ is then the set of all mappings from the countably infinite set $Q \times \Sigma$ to the countably infinite set Q. The cardinality of this set of functions is $|Q|^{|Q \times \Sigma|}$, which, since both Q and $Q \times \Sigma$ are countably infinite, is equivalent to $|\mathbb{N}|^{|\mathbb{N}|}$, known to be uncountable as shown by Cantor's theorem.

Since the transition function δ can vary in an uncountably infinite number of ways (even if every other component of the DIA were fixed), the set of all possible DIAs is likewise uncountable.

The set of all possible DIAs is uncountable because the set of all possible transition functions δ is uncountable. This remains true regardless of the finite nature of Σ and the finite or countably infinite nature of F.

Question assigned to the following page: 4

4. (8 points) Prove that the set of total functions from $\mathbb{N} \to \mathbb{N}$ is uncountable by diagonalization.

To prove that the set of total functions from $\mathbb{N}$ to $\mathbb{N}$ is uncountable, we can use Cantor's diagonalization argument.

A total function from $\mathbb{N}$ to $\mathbb{N}$ is a function that assigns to every natural number n a unique natural number $f(n)$. Assume for the sake of contradiction that the set of all such functions is countable. If this were true, then it would be possible to list all possible total functions:

$$\begin{aligned} f_1 &: 0 \mapsto a_{10}, 1 \mapsto a_{11}, 2 \mapsto a_{12}, \ldots \\ f_2 &: 0 \mapsto a_{20}, 1 \mapsto a_{21}, 2 \mapsto a_{22}, \ldots \\ f_3 &: 0 \mapsto a_{30}, 1 \mapsto a_{31}, 2 \mapsto a_{32}, \ldots \\ &\vdots \end{aligned}$$

Now, construct a new function $g : \mathbb{N} \to \mathbb{N}$ defined by $g(n) = f_n(n) + 1$. This function g is clearly different from every function f_n in the list, because it differs from each f_n in at least the n-th term; specifically, for each n, $g(n) \neq f_n(n)$ by construction.

However, this leads to a contradiction because g is also a total function from $\mathbb{N}$ to $\mathbb{N}$, and by our assumption, it should have been listed in the enumeration of all such functions. Since it was not, our original assumption that the set of all total functions from $\mathbb{N}$ to $\mathbb{N}$ is countable must be false.

Therefore, the set of total functions from $\mathbb{N}$ to $\mathbb{N}$ is uncountable.

Question assigned to the following page: 5

5. (8 points) This question tests your ability to construct and defend a rigorous formal argument. Recall one of the greatest applications of the Church-Turing Thesis. You may be considerate of an algorithm (to some extent) and then, by the Church-Turing Thesis, a device must exist (to some extent). Your assignment is to determine to what extents does the Church-Turing Thesis apply. See the attached chart on canvas. There are two axii. One of logical purism, neutrality, and rebellion. The other of existential purism, neutrality, and rebellion. I have provided an example for each of the nine categories, organized into a three by three table. Write a rigorous and persuasive argument as to which of the nine categories you can best apply the Church-Turing Thesis. Note you are not arguing correctness of the provided example, but of its category. Your argument should convince me your selection of the nine is correct, and the other eight are incorrect.

   The detailed analysis of the provided chart with the nine categories reveals diverse applications of the Church-Turing Thesis across various logical and existential dimensions. However, the core of the Church-Turing Thesis aligns most significantly with the category of Logical Neutrality and Existential Neutrality.

   This suggests that while the Church-Turing Thesis provides a foundation for understanding computation, it acknowledges the complexities and nuances of real-world applications and existential implications. This neutrality allows for a balanced view, recognizing the limits of formal computation while embracing the practicality and unpredictability of computational applications in natural and social systems.

   The categories of purism, either logical or existential, may limit the scope of the Church-Turing Thesis by confining it to theoretical or idealized contexts, which could ignore practical considerations and real-world applications. On the other hand, the categories of rebellion, whether logical or existential, might stretch the thesis beyond its meaningful applicability, challenging its foundational principles without providing a coherent framework for computation.

   Therefore, the combination of Logical Neutrality and Existential Neutrality best represents the Church-Turing Thesis's scope and applicability. It accommodates the thesis's foundational strength in defining what is computationally possible while allowing for the flexibility needed to address real-world computational challenges and philosophical questions.

   This balanced approach does not overextend the reach of the Church-Turing Thesis into areas where it might not apply while still acknowledging the broad spectrum of computational theory. Hence, among the nine categories, Logical Neutrality and Existential Neutrality best encapsulate the application of the Church-Turing Thesis.