Theoretical Foundations of Computer Science

2021 Assignment

This assignment will not be submitted, but the questions in the assignment tests will be based directly on the questions below. Note that while some questions may be identical, the general idea is that the concepts covered in the questions are the same. In other words, you need to understand the answers to these questions, not just memorize them.

If you think that a question has multiple interpretations, you may want to prepare for all possible interpretations. If there is an obvious modification to a question, you may want to consider what the answer would be if that modification were made. If you need to make any assumptions in order to answer a question, be sure to state these if asked about that question and be ready to justify why they were needed. If you're unsure about a question, just do your best.

The assignment tests themselves will be vivas (individual verbal assessments) carried out via a platform such as Skype or Teams. More details will be available on Blackboard.

Section α – Computability Classification

For each of the following problems, classify it as being either Regular (Tier 1), Context-Free (Tier 2), or higher, and prove its membership of that category.

To prove membership, do the following:

- To show that something is Regular you should provide a DFA, NFA or RE that solves the problem.
- To show that something is Context-Free you should provide a proof that it is not Regular and should also provide a PDA or CFG that solves the problem.
- If you believe that something is higher than Tier 2, prove this in an appropriate way.

If you cannot prove or disprove membership formally, give a convincing argument. If you can't provide a full machine, provide enough to demonstrate that such a machine is possible. If you need to make assumptions, state these clearly. If you are not sure of a term, make sure that you look up a definition from a reliable source.

PROBLEM 1 – Simple Strings I

The string $01^x0^y1^z0^w$ where $x, y, z, w > 0$.

PROBLEM 2 – Simple Strings II

The string $01^x0^y1^x0^y$ where $x, y > 0$.

PROBLEM 3 – Simple Strings III

The string $01^x0^x1^y0^y$ where $x, y > 0$.

PROBLEM 4 – Combined Substrings

The language of binary strings that contain the sub-string 01^n0 and the sub-string 10^m1 where

$0<n<m$.

PROBLEM 5 – Combined Substrings

The language of binary strings that contain either the sub-string 01^n0 or the sub-string 10^n1 where $0<n$.

PROBLEM 6 – Happy Cats

For L = {cat, dog, fish} find all strings with happy cats (which means that the string contains at least as many fish as cats and no dogs).

PROBLEM 7 – Happy Dogs

For L = {cat, dog, fish} find all strings with a happy dog (a dog is happy when it immediately follows a cat).

PROBLEM 8 – Counting Coins

A machine designed for counting coins is to have an additional circuit added. The existing machine will send a bit string representing each coin to the new circuit (11 is for $2, 10 for $1, 01 for 50c and 00 for 20c – the machine does not accept 10c or 5c coins). The new circuit is to check whether the machine receives more than twice as many $1 and $2 coins (combined) as $50c coins in order to determine whether the size of the container for 50c coins needs to be decreased.

PROBLEM 9 – Skin in the Game

A player is studying the relative usage of his favourite Fortnite skins. He is particularly keen on Meowscles, Mandalorian and Deadpool. He has set up a bot to monitor Twitch Fortnite streams which reports whether any of those skins were used in that stream.

The bot will return three numbers, each corresponding to the number of players in that match that were spotted using the corresponding skin. So, if the bot returns 301 it means that three different players were spotted using Meowscles, non were spotted with Mandalorian and one was spotted wearing Deadpool.

While the bot may end up not seeing every player in each stream and may indeed scan a multiple match several times if it is streamed by different content creators, the player feels that this is still a representative sample. He plans to let the bot run over one week of streams and then see which one is the *least* popular. Your software should return the least popular skin, although you may use code to do so (e.g., 1 for Deadpool, etc).

PROBLEM 10 – Getting Bigger

A chip manufacturer is experiencing faults in their chipsets. Your task is to build a device to check that the increment is successful. The input will be the original input to the increment part of the circuit and the output from that circuit. For example, if the original input was A and the output was B (which is hopefully equal to an incremented A) then you will be given both A and B as inputs to your device. You may accept the digits of the inputs in any order but can only read each digit once. Your device will return accept if the output B is correct, given input A.

[Examiner's note - While it would be entertaining to ask you to do this for the full 64-bit increment it is sufficient to look at the 8-bit increment. In practice, several 8-bit increment circuits are used in parallel or in sequence, so this isn't unrealistic. Even looking at the 4-bit version will be sufficient, since the principle is the same.]

PROBLEM 11 – ASDF

Your input is a paragraph of English text. Accept if the number of times the phrase “I like trains” occurs is equal to the number of times the word “aargh” occurs (without quotes).

PROBLEM 12 – British Civilization

A You Tube content creator called the Spiffing Brit has asked your help with checking one of his exploits is working. He is testing yet another possible exploit in the game Sid Meier’s Civilization XIII.

Spiff has set up some bots to play Civ XIII and take a very specific series of actions with a Scout unit. The idea is that doing this then causes a nearby barbarian encampment to spawn an additional unit. This can be exploited using a proper setup that gives discounts on hiring barbarian units.

The output of the bots is somewhat complex, but if reduced to binary, the completion of the series of actions by the scout unit is shown by a 101 in the string and the spawn of a barbarian unit is shown by 001. The theory is that every time the scout finishes their actions, a barbarian unit will spawn quickly enough so that there are at most 4 binary digits between the two. If this happens every time, you are to accept the string.

Spiff will let the bots run many times, each time for an hour, which will then be input to your program as a binary string. Your software is to accept if his theory is correct for the given string.