Problem Set 7

Due Date ..... October 25, 2022
Name ..... **Alex Ojemann**
Student ID **109722375**
Collaborators ..... **None**

Contents

1 Instructions

- The solutions **should be typed**, using proper mathematical notation. We cannot accept hand-written solutions. (See this short intro to LaTeX plus other resources on Canvas.)
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section 2). Failure to do so will result in your assignment not being graded.

2 Honor Code (Make Sure to Virtually Sign)

- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

(I agree to the above, Alex Ojemann). □

3 Standard 18: TM Basics

Problem 1. Complete the Canvas quiz. There is nothing you need to submit with the PDF.

4 Standard 19: Turing Machines

Problem 2. Design a Turing Machine that accepts the set $\{a^n : n \text{ is a power of } 2\}$. Your description should be at the level of the descriptions Lecture 29 of Kozen (c.f., Example 29.1). In particular, do **not** give a list of transitions or a state-transition diagram.

Answer. Let tape 1 hold the input string and tape 2 hold one a.
The head on tape 1 starts at the left end symbol and the head on tape 2 start at its only a.
The head on the tape 1 moves right. The head on tape 2 moves left each step until it reaches its left end symbol, where it switches to moving right.
If the head on tape 2 reaches its last non-empty symbol before the head on tape 1 reaches its last non-empty symbol, tape 2 doubles its amount of as by adding the same number of as counted in the original string to the end of the string overwriting blanks and the process resets to the first step.
If the head on tape 1 and the head on tape 2 reach their last non-empty symbol at the same time, this machine accepts. If the head on tape 1 reaches its last non-empty symbol and the head on tape 2 doesn't do so on the same step, then this machine rejects. □

Problem 3. Let $A = L(1(1+0)^* + 0)$ be the set of binary strings with no leading zeros.

Given $x \in A$, let $\#x$ denote the number x represents in binary. Your task is to implement a binary counter using a Turing machine: Design a TM M such that, given input $x \in A$, halts with just y on the tape, where $\#y = \#x + 1$. More precisely, the final tape should be $\vdash y \sqcup \omega$. For example, on input 10011, the machine halts with only 10100 on the tape.
Hint: Sometimes the output will need to be longer than the input.

Your description should be at the level of the descriptions Lecture 29 of Kozen (c.f., Example 29.1). In particular, do **not** give a list of transitions or a state-transition diagram.

Answer. Scan the input string x from left to right. If we see a 0 first and non-empty characters after, we reject because $x \notin A$.
If we see a 0 first and empty characters after, we write a 1 in its place, then accept. If the string has at least one 0 and at least one 1, we do the following when scanning back to the left. If we see a 0 first, we write a 1, then accept. If we see a 1 first, we write a 0 and move left. As long as we encounter only 1s, we write 0s on those tape spots and move left. As soon as we encounter a 0, we write a 1 on that tape spot, then accept.
If the input string has only 1s, we can't add another 1 because #y would exceed #x + 1. We move left back to the beginning of the string without writing anything. When we're back at the beginning, we see the first one and move right. From then on, we write a 0 in place of each 1 and move right until we reach a blank. When we reach a blank we write a 0 over it and accept. □

5 Standards 2/3: Proofs

Problem 4. Consider the following CFG G in Greibach normal form:

$$S \to aB \mid bA$$
$$A \to aS \mid bAA \mid a$$
$$B \to bS \mid aBB \mid b.$$

Prove that if $x \in \{a, b\}^*$ is a non-empty string with equally many a's and b's, then G generates x.

Proof. □

Let n be the length of string x. For the given language, n can only be an even number.
Base Cases:
n = 0: $\#a(\epsilon) = \#b(\epsilon) = 0$
n = 2: Could be S − > aB − > ab or S − > bA − > ba
$\#a(ab) = \#b(ab) = 1$
$\#a(ba) = \#b(ba) = 1$
Inductive Case: Assume that G generates all strings where $\#a(x) = \#b(x)$ for some $k \geq 4$ where k is the length of the string x. We will show that G generates all strings of length k + 2 where $\#a(y) = \#b(y)$.
The only two rules which can generate the last letter of a string in G are the "a" rule of nonterminal A and the "b" rule or nonterminal "b" because all other rules end with a nonterminal that doesn't have a rule for the empty string. The following are the only possible ways to turn x into a string y of length k + 2 that is in G.
Replace the final a with the bAA rule from A. bAA − > baA − > baa. $\#a(y) = \#b(y) = \#a(x) + 1 = \#b(x) + 1$.
Replace the final a with the aS rule from A. aS − > aaB − > aab. $\#a(y) = \#b(y) = \#a(x) + 1 = \#b(x) + 1$.
Replace the final a with the aS rule from A. aS − > aBa − > aba. $\#a(y) = \#b(y) = \#a(x) + 1 = \#b(x) + 1$.
Replace the final b with the aBB rule from B. aBB − > abB − > abb. $\#a(y) = \#b(y) = \#a(x) + 1 = \#b(x) + 1$.
Replace the final b with the bS rule from B. bS − > baB − > bab. $\#a(y) = \#b(y) = \#a(x) + 1 = \#b(x) + 1$.
Replace the final b with the bS rule from B. bS − > bBa − > bba. $\#a(y) = \#b(y) = \#a(x) + 1 = \#b(x) + 1$.
Since the only possible ways to replace the last letter of a string in $\{a, b\}^*$ with a three letter string and maintain the equivalence in the number of as and bs are to replace an ending a with baa, to replace an ending a with aab, to replace an ending a with aba, to replace an ending b with abb, to replace an ending a with bab, and to replace an ending a with bba, G must generate all possible strings of length k+2 where the number of as and bs are the same.