TOC PROBLEM SET-12

SIDDHANT CHAUDHARY
BMC201953

Problem 1. A Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where Q is the set of states, Γ is the tape alphabet, $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function, q_0, q_{accept}, q_{reject} are the initial, accept and reject states respectively. Note that accept and reject states are distinct.

Give the Turing Machine for the following language by describing the TM with the 7-tuple defined above or the corresponding state diagram (as in Sipser, Chapter 3):

$$L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$$

Solution. The idea we use is very simple. Let

$$Q = \{s, q_1, q_2, q_3, q_4, r, t\}$$

where s is the starting state, r is the reject state and t is the accepting state. Also, let

$$\Gamma = \{\vdash, \sqcup, a, b, c, a_m, b_m, c_m\}$$

where the m in the subscript means *marked*. Let me explain the working of the TM with an example. Suppose the input is $\vdash aaabbbccc$. The machine will find the *first* unmarked a, mark it and keep moving right, then find the *first* unmarked b, mark it and keep moving right, then find the *first* unmarked c, mark it, and then start moving *left*, until we find the first marked a (i.e a_m), and at that point we move right and repeat the process. After all the letters are marked, the machine just checks if there are any other remaining letters. If there are none, then the word is accepted, otherwise it is rejected. The state diagram of the machine is as follows (we assume that in the below diagram, if an edge is not present, then it leads to the reject state r. We also assume that the states t and r have self loops for every possible edge).

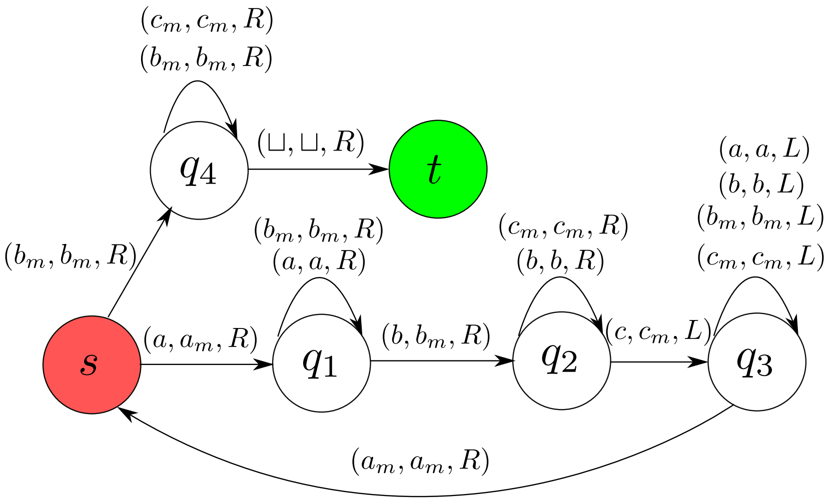
I have not included the state r in the above diagram because of the remark made just above the diagram in pink. ■

Problem 2. A Multitape Turing Machine is like an ordinary Turing Machine with multiple tapes and read/write head for each tape. Initially, the input appears on tape 1. The transition function is defined as $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$. Prove that the ordinary single tape Turing Machines are as powerful as Multitape Turing Machines.

Solution. In this problem, writing the transitions down explicitly will be a mess. So, I will just give an informal description about how the TM will work. All of the work will be done by operations on a single tape.

Suppose we are given a Multitape Turing Machine M, where the transition function is of the form $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$. We will construct a Turing

Date: November 2020.



Machine M' on a *single tape* which accepts the same language. Suppose the left-endmarker of M is $\vdash$. So, we will let $\vdash_1$ be the left-endmarker of M'. Now the idea is as follows; suppose our input word in Σ^* is w. So, the initial configuration of the k-tapes in the Multitape TM M will be $s \vdash w$, $s \vdash$, ... , $s \vdash$, i.e only the first tape will have the input $\vdash w$, and the rest of the tapes will only consist of the left-endmarker $\vdash$. So, our TM M' will first write the initial configuration of *each* of the k-tapes on its tape, and each configuration will be separated by a special symbol, say #. So, the TM M' will begin by writing the following on its tape:

$$\vdash_1 s \vdash w \sqcup \# s \vdash \sqcup \# s \vdash \sqcup \# \ldots \sqcup \# s \vdash \sqcup \sqcup \sqcup \ldots$$

where there are $k-1$ #'s, as M contains k-tapes (notice the blank symbol $\sqcup$ before every #. This will be important as we see ahead). Note that this notation is helpful, because it indicates the position of the head on each of the tapes. This suggests that the tape alphabet of M' is $Q \cup \Gamma \cup \{\#\}$.

Now, we describe the transitions in M'. The idea is just to simulate $M's$ transitions on each of its tapes on the single tape. So, suppose the tape of M reads the following.

$$\vdash_1 w_1 q c_1 w_1' \sqcup \# w_2 q c_2 w_2' \sqcup \# \ldots \sqcup \# w_k q c_k w_k' \sqcup \sqcup \sqcup \ldots$$

where each $w_i, w_i' \in \Gamma^*$ and each $c_i \in \Gamma$. As an example, suppose the transition in the original machine M is $(q, \{c_1, c_2, ..., c_k\}) \to (q', \{c_1', c_2', ..., c_k'\}, \{R, R, ..., R\})$. Then our machine M' will execute these transitions *one-by-one* for each of the k-tapes, i.e after the first step the tape of M' will be

$$\vdash_1 w_1 c_1' q' w_1' \sqcup \# w_2 q c_2 w_2' \sqcup \# \ldots \sqcup \# w_k q c_k w_k' \sqcup \sqcup \sqcup \ldots$$

and similarly the subsequent transitions will be performed on each tape (by each tape we mean each #-separated block), and hence after k-steps the tape will read

$$\vdash_1 w_1 c_1' q' w_1' \sqcup \# w_2 c_2' q' w_2' \sqcup \# \ldots \sqcup \# w_k c_k' q' w_k' \sqcup \sqcup \sqcup \ldots$$

Note that all of these k-steps will be done *consecutively* in M'. Finally, we need to resolve the issue of the tape space running out for each of the k-tapes. But this is easy. Before performing any transition as above, our TM M will check whether there is *atleast one* blank sybmol $\sqcup$ *immediately before* every # (this is where the importance of the blank symbols comes in). If there is no blank symbol before some #, then the machine M' will copy *all of the remaining* tape and move it right by *one step*, and will add a blank symbol at that position. (All of this is hard to explain by explicit notation, and that's why we have to resort to explaining it in words).

Hence, we have created a single tape Turing Machine M' that is equivalent to the machine M. The set of states of M' will be the ones which help in doing the tape operations as mentioned above, but the overall idea is to just simulate the k-tapes on a single tape. ■

Problem 3. Show that the class of recursive languages is closed under the operation of

(1) union
(2) concatenation
(3) star
(4) complementation
(5) intersection

Solution. To be completed ■

Problem 4. Show that the class of recursively enumerable languages is closed under the operation of

(1) union
(2) concatenation
(3) star
(4) intersection
(5) homomorphism

Solution. To be completed ■