Some New Exercises on Computability Theory Associated with Turing Machine

B. P. Tanana
Higher Institute for Science and Technology
ISCTEM
Maputo, Mozambique
e-mail:btanana@mail.ru

B. Cassy
University Eduardo Mondlane (UEM)
Dept. of Mathematics and Informatics (DMI)
Maputo, Mozambique
e-mail: bhangycassy@yahoo.com.br

Abstract—In this paper we present the problems previously encountered in the academic literature related to Turing machines that allow to check the students' understanding of the actual, rather than formal memorizing definitions and concepts.

Keywords*-Turing machine language recognized by a Turing machine, the encoding of the Turing machine*

*****

I. Introduction

In this paper, we present the problems not previously encountered in the academic literature related to Turing machines that allow students to check the actual understanding rather than memorization of the formal definitions and problems about Turing machines.

This article is, in some sense, a continuation of [3], [4] and [5]. Students in Mozambique, specializing in the field of engineering science, in different disciplines taught sections of discrete mathematics associated with Turing machines.

II. Definitions and Examples

We give, initially, the necessary definitions, as follow:

A **Turing machine** consists of a finite control (any non-empty finite set of states), from the infinite to the right tape divided into cells (each cell can contain one of a finite set of symbols) and the cursor that can read a character from the scanned cell, write the new symbol and move on one cell to the right or left.

More formally (p. 319 [1]), the **Turing machine** is a system of seven elements M = (Q, Σ, T, δ, q_0, B, F) where,

Q – a finite set of states;

Σ - finite set of input symbols;

T - a finite set of tape symbols (Σ proper subset of T);

δ - transition function, δ: Q × T → Q × D × {L, R}, (δ - partial function, L means moving the cursor to the left, R - right);

q_0 - initial state ($q_0 \in Q$, in this state of the Turing machine is at the beginning of the work);

B - symbol tape, indicating an empty cell (B ∈ G \ Σ);

F - a set of final or accepting states (F is a subset of Q).

Starting work, a Turing machine is in the initial state q_0, the cursor scans the first left cell, in which is always a symbol B in the (beginning of the tape), and then on the tape is located input word w over the alphabet Σ, all the remaining cells (to infinity) recorded a symbol B. Then the machine step by step starts to begin to execute commands (if possible).

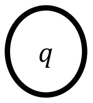
For example, the machine started to work (in our case) in the transition function there must be the command δ (q_0, B) = (q_1, B, R). For this command, the machine goes into the state q_1, the first cell on the left leaves the symbol B and the cursor moves to the cell to the right. In general, the command δ (q_i, x) = (q_j, y, L) is performed as follows: before executing the command, Turing machine is in a state of q_i in the scanned cell recorded an x.

After running this command, the machine goes into the state q_j, in scan cell replaces x by y and the cursor moves to the cell to the left, which is to be scanned.

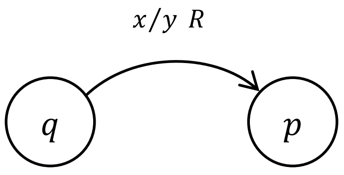
If doing so the command asked the transition function Turing machine M is stopped (no command that she could perform) and is in the final state of the Turing machine M accepts a word w, written at the beginning of work on the tape, otherwise, does not accept w. All words w, a Turing machine M that accepts form a language recognized by a Turing machine M. This language is denoted by L (M).

Turing machines conveniently presented in the form of so-called transition diagrams (directed graphs):

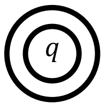
Each state q is a vertex:



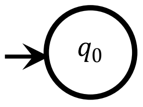
Let δ (q, x) = (p, y, R). Then there is an arc from q to p with weight x/y R:



The tops of the respective final states:



Vertex corresponds to the initial state:

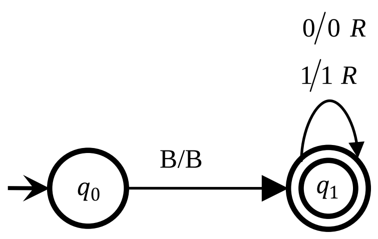


For example, a state transition diagram for a Turing machine M:

δ (q_0, B) = (q_1, B, R)

δ (q_1,0) = (q_1,0, R)
δ (q_1,1) = (q_1,1, R)

M:



Consider one of the ways of encoding of Turing machines. Without loss of generality, in matters related to the encoding of Turing machines, we can assume that, T= {0, 1, B}, the set of states Q= {q_0, q_1, ..., q_n} (p. 324 [2]), also agreed to that F={q_1}.

We need to encode commands of transition function, which have the form δ (q_i, x)=(q_j, y, D), where q_i, q_j∈Q; x, y∈{0, 1, B}; D∈{L, R}.

Encode all the characters included in the command (p. 324 [2]):

Symbol	0	1	B	q_0	q_1	…	q_n	L	R
Code	1	11	111	1	11	…	1^{n+1}	1	11

Let c(z) be a code of symbol z. **A code of the command**δ(q_i, x) = (q_j, y, D) be a binary word,

$$c(q_i)\,0c(x)\,0c(q_j)\,0c(y)\,0c(D).$$

A **code of a Turing machine M** is a binary word, denoted by R(M), which begins and ends with three zeros between which command codes are written separated by two zeros.

Consider the example:

Example 1. Find the code R(M) Turing machine M, defined by the transition function

δ (q_0, B)= (q_1, B, R);
δ (q_1, 0) = (q_1, 0, R);
δ (q_1, 1) = (q_1, 1, R).

Solution. Note that the transition function defines all the 7 elements of a Turing machine

M = (Q, Σ, T, δ, q_0, B, F);
Q = {q_0, q_1}, Σ = {0, 1},
F = {0, 1, B},
δ is given in the text of Example 1,
q_0 = q_0,
B = B,
F = {q_1} (remember our agreement).

Thus, let's find R(M).

Command	Code
δ(q_0, B)=(q_1, B, R)	101110110111011
δ(q_1, 0)=(q_1, 0, R)	110101101011
δ(q_1, 1)=(q_1, 1, R)	11011011011011

So,R(M)=

= 000101110110111011001101011010110011011011011011000

Changing the order of the command, as a result, it is possible to obtain 3! = 6 different codes R(M) for a given in Example 1 of the Turing machine M.

It is not difficult to understand that a Turing machine M from Example 1 accepts any word in the alphabet {0, 1} as always stops in the final state q_1. So, L(M), in this case, is the set of all words in the alphabet {0, 1}, including the empty word λ.

Thus Turing machines can be represented by binary words. We assume that every binary word is a code of a Turing machine, but some codes have mistaken and a Turing machine does not work (pp. 370-371 [1]).

We introduce the **canonical order** on the binary words: w_1<w_2, if |w_1|<|w_2|, the words of equal length are ordered lexicographically.

So, the initial ordered set of items are received by words λ, 0, 1, 00, 01, 10, 11, 000, ...

If for these words write 1 to the left side, then we get the binary numbers 1, 10, 11, 100, 101, 110, 111, 1000, ... or in decimal 1, 2, 3, 4, 5, 6, 7, 8 ..., words that define binary numbers under the canonical ordering (p. 369 [1]).

Sometimes instead of a binary word w_i (word having the number i under the canonical ordering) is convenient to use its number i.

To find enough i for w_i is necessary write 1 to the left side of $w_{i.}$ we introduce the **canonical order** on the binary words: w_1<w_2 if |w_1|<|w_2|, the words of equal length are ordered lexicographically.

Thus, the initial ordered set of items are received by words λ, 0, 1, 00, 01, 10, 11, 000, ...

If for these words write 1 to the left side, then we get the binary numbers

1, 10, 11, 100, 101, 110, 111, 1000, ...

or in decimal,1, 2, 3, 4, 5, 6, 7, 8 ..., words that define binary numbers under the canonical ordering (p. 369 [1]).

In other times, instead of a binary word w_i (word having the number i under the canonical ordering) it is convenient to use its number i.

So, to find enough i for w_i is necessary to write 1 at the left side of w_i.

III. EXERCISES/PROBLEMS

A. *Problem 1*

Construct a Turing Machine M, which converts the binary word 1w for word w (w any binary word).

Solution – Problem 1 (see below).

B. *Problem 2*

Construct a Turing Machine M, which converts the binary word w for word 1w (w any binary word).

Solution – Problem2 (see below).

C. *Problem 3*

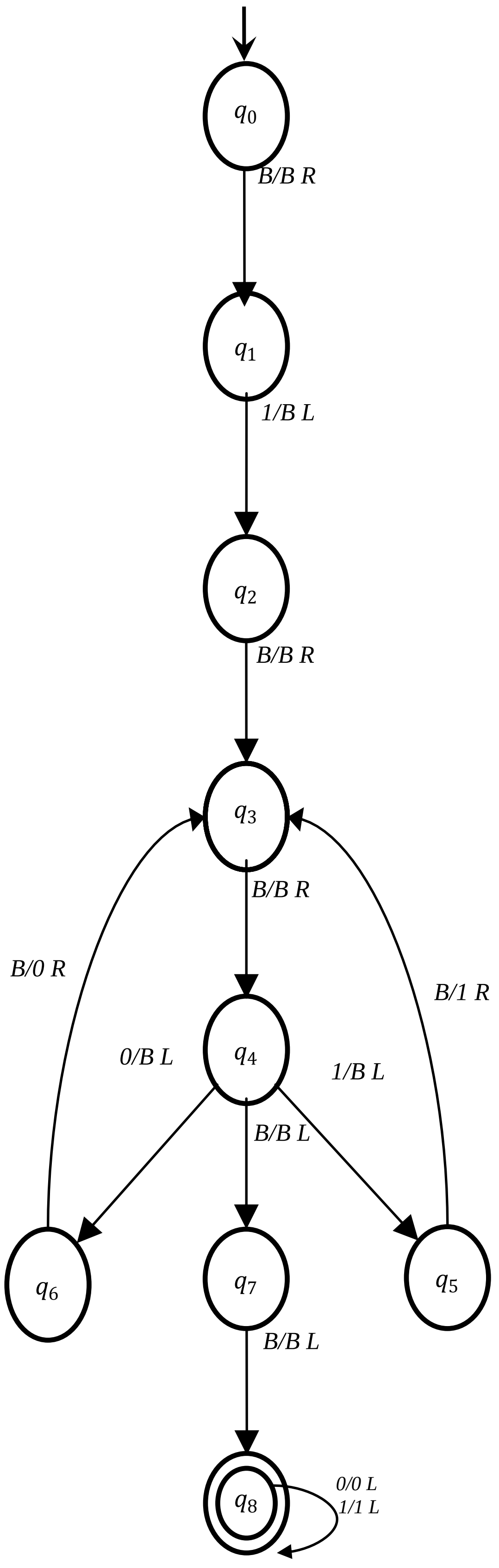
Find the smallest number of binary words with respect to the canonical order, which determines the actual code of a Turing machine.

Solution – Problem 3 (see below).

IV. Solutions of Problems 1-3

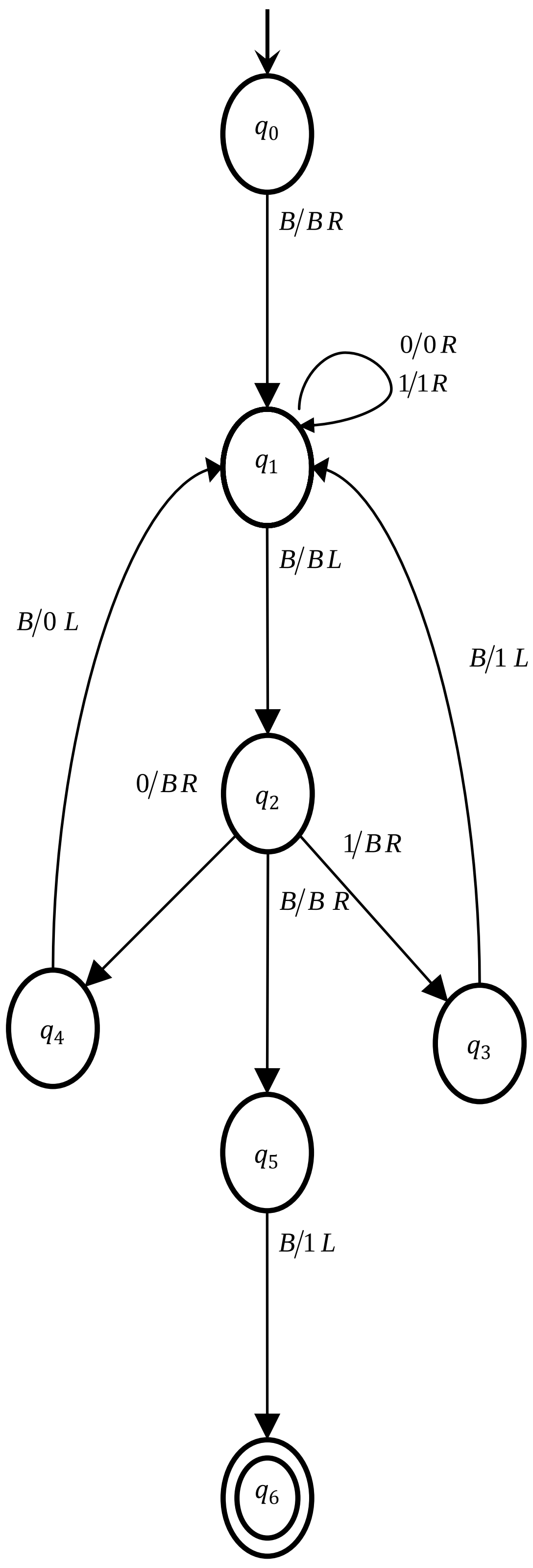
Solution – Problem 1:

M:



Solution – Problem 2:

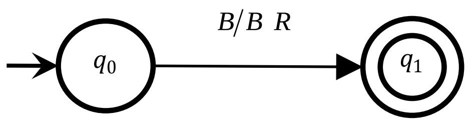
M:



Solution – Problem 3:

It is clear that for the restrictions accepted by us, "the smallest" Turing machine M has a single command:$\delta(q_0, B) = (q_1, B, R)$.

M:



So, R(M) = 0001011101101110110000.Then, the number of the word R(M) in the binary system is 10001011101101110110000, and the decimal notation of this numberis equal to 1.175.000.

REFERENCES

[1] J. E. Hopcroft, R. Motwani, J. D. Ullman. Introduction to Automata Theory, Languages, and Computation. 2nd ed, Addison Wesley, 2001.

[2] T. A. Sudkamp. Languages and Machines. 2nd ed, Addison Wesley, 1997.

[3] B.P.Tanana, B.Cassy, “On the Coding of a Turing Machines and Non – Recursive Languages”, International Journal on Recent and Innovation Trends and Communications, Vol. 3, Issue 2, February 2015, pp. 613-615.

[4] B. Tanana and B. Cassy, “About Pumping Lemma for Regular Languages”, International Journal of Innovative Science, Engineering & Technology, Vol. 3, Issue 3, March 2016.

[5] B.P. Tanana. On the Coding of a Turing machines and Non-Recursive Languages. MATVEST, N 17, April, 2015.