0.1.1. 3.1 This exercise concerns TM M_2, whose description and state diagram appear in Example 3.7. In each of the parts, give the sequence of configurations that M_2 enters when started on the indicated input string.

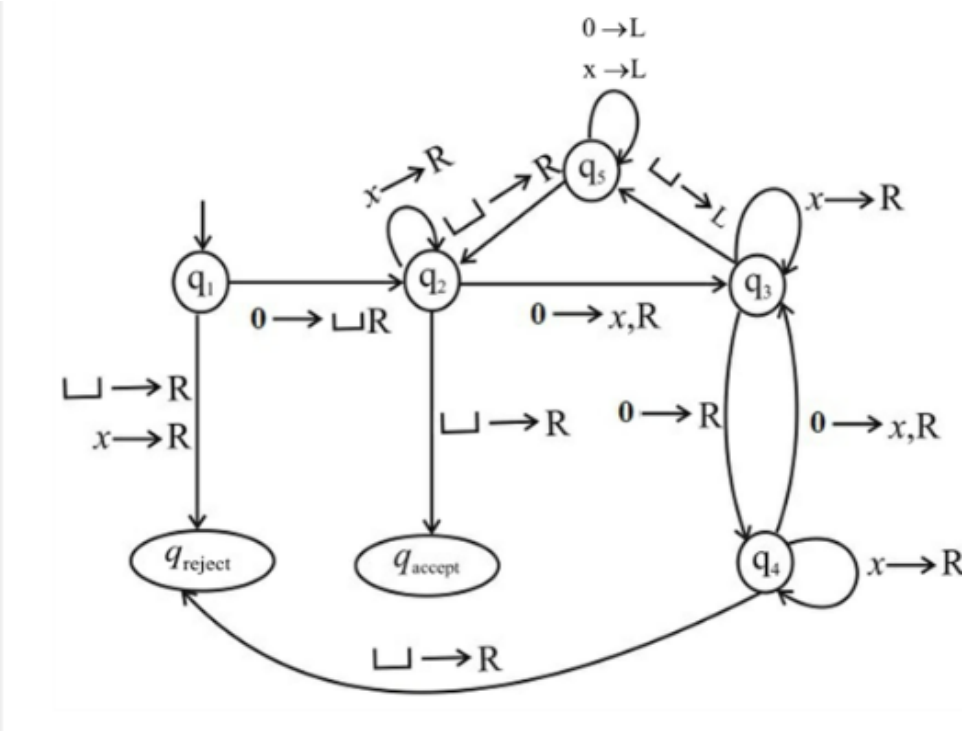

Figure 1:

0.1.1.1. a) 0

$$q_1 0$$

 note that a computation is a sequence of configurations and such sequence must be finite!

 Recognizability of turing machine requires for the machine to have a language and for that halting is not required.

 a TM is decidable iff the machine halts and it language is the set of recognizable languages

 We don't cross the starting zero because is "every other 0"

$$␣q_2␣$$

$$␣␣q_{\text{accept}}␣$$

0.1.1.2. b) 00

$$␣q_1 00␣$$

$$␣q_2 0␣$$

$$␣xq_3␣$$

$$␣q_5 x␣$$

$$\text{␣}q_5\text{␣}x\text{␣}$$

$$\text{␣}q_2x\text{␣}$$

$$\text{␣}xq_2\text{␣}$$

$$\text{␣}x\text{␣}q_{\text{accept}}\text{␣}$$

0.1.1.3. c) 000

$$\text{␣}q_1000\text{␣}$$

$$\text{␣}q_200\text{␣}$$

$$\text{␣}xq_30\text{␣}$$

$$\text{␣}x0q_4\text{␣}$$

$$\text{␣}x0\text{␣}q_{\text{reject}}\text{␣}$$

0.1.1.4. d) 000000

$$\text{␣}q_1000000\text{␣}$$

$$\text{␣␣}q_200000\text{␣}$$

$$\text{␣␣}xq_30000\text{␣}$$

$$\text{␣␣ x0 }q_4000\text{␣}$$

$$\text{␣␣ x0x }q_300\text{␣}$$

$$\text{␣␣ x0x0 }q_40\text{␣}$$

$$\text{␣␣ x0x0x }q_3\text{␣}$$

$$\text{␣␣ x0x0 }q_5x\text{␣}$$

$$\text{␣␣ x0x }q_5\text{ 0x ␣}$$

$$\text{␣␣ x0 }q_5\text{ x0x ␣}$$

$$\text{␣␣}xq_5\text{ 0x0x ␣}$$

$$\text{␣␣}q_5\text{ x0x0x ␣}$$

$$\text{␣␣}xq_2\text{ 0x0x ␣}$$

$$\text{␣␣ xx }q_3\text{ x0x ␣}$$

$$\text{␣␣ xxx }q_3\text{ 0x ␣}$$

$$\text{␣␣ xxx0 }q_4x\text{␣}$$

$$\text{␣␣ xxx0x }q_4\text{␣}$$

$$\text{␣␣ xxx0x ␣}q_{\text{reject}}\text{␣}$$

0.1.2. 3.2 This exercises concerns TM M_1 whose description and state diagram appear in Example 3.9. In each of the parts give the sequence of configurations that M_1 enters when started on the indicated input string.

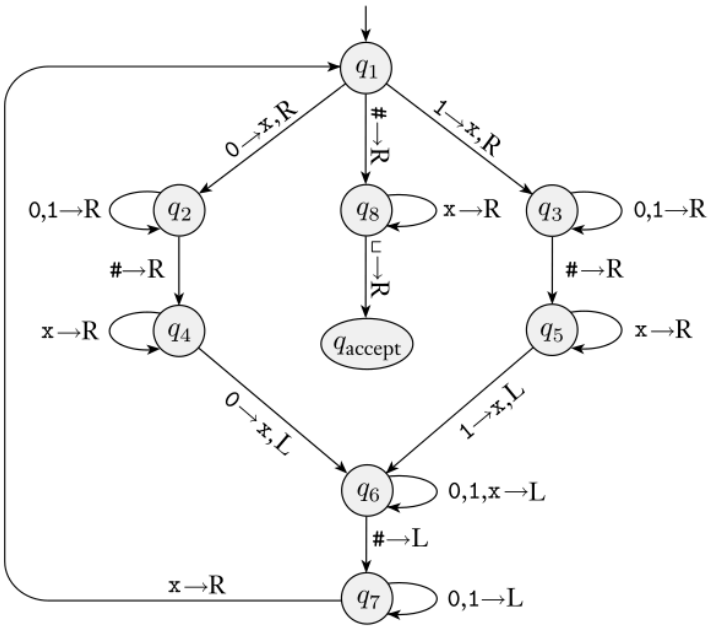

Figure 2:

0.1.2.1. a) 11

$$\llcorner q_1 11 \lrcorner$$

$$\llcorner x q_3 1 \lrcorner$$

$$\llcorner \text{x1 } q_3 \lrcorner$$

$$\llcorner \text{x1 } \llcorner q_{\text{reject}}$$

0.1.2.2. b) 1#1

$$\llcorner q_1 1\#1 \lrcorner$$

$$\llcorner x q_3 \#1 \lrcorner$$

$$\llcorner \text{x\# } q_5 1 \lrcorner$$

$$\llcorner x q_6 \text{ \#x } \lrcorner$$

$$\llcorner q_7 \text{ x\#x } \lrcorner$$

$$\llcorner x q_1 \text{ \#x } \lrcorner$$

$$\llcorner \text{x\# } q_8 x \lrcorner$$

$$\llcorner \text{x\#x } q_8 \lrcorner$$

$$\text{␣ x\#x ␣}q_{\text{accept}}\text{␣}$$

0.1.2.3. c) 1##1

$$\text{␣}q_1\text{1\#\#1␣}$$
$$\text{␣}x q_3\text{\#\#1␣}$$
$$\text{␣ x\# }q_5\text{\#1␣}$$
$$\text{␣ x\# \#}q_{\text{reject}}\text{1␣}$$

0.1.2.4. d) 10#11

$$\text{␣}q_1\text{10\#11␣}$$
$$\text{␣}x q_3\text{0\#11␣}$$
$$\text{␣ x0 }q_3\text{\#11␣}$$
$$\text{␣ x0\# }q_5\text{11␣}$$
$$\text{␣ x0\#x }q_6\text{1␣}$$
$$\text{␣ x0\# }q_6\text{ x1 ␣}$$
$$\text{␣ x0\#x }q_{\text{reject}}\text{1␣}$$

0.1.2.5. e) 10#10

$$\text{␣}q_1\text{10\#10␣}$$
$$\text{␣}x q_3\text{0\#10␣}$$
$$\text{␣ x0 }q_3\text{\#10␣}$$
$$\text{␣ x0\# }q_5\text{10␣}$$
$$\text{␣ x0 }q_6\text{ \#x0␣}$$
$$\text{␣}x q_7\text{ 0\#x0␣}$$
$$\text{␣}q_7\text{ x0\#x0␣}$$
$$\text{␣}x q_1\text{ 0\#x0␣}$$
$$\text{␣ xx }q_2\text{ \#x0␣}$$
$$\text{␣ xx\# }q_4 x\text{0␣}$$
$$\text{␣ xx\#x }q_4\text{0␣}$$
$$\text{␣ xx\# }q_6\text{ xx ␣}$$

$$\text{␣ xx } q_6 \text{ \#xx ␣}$$

$$\text{␣}x q_7 \text{ x\#xx ␣}$$

$$\text{␣ xx } q_1 \text{ \#xx ␣}$$

$$\text{␣ xx\# } q_8 \text{ xx ␣}$$

$$\text{␣ xx\#x } q_8 x \text{␣}$$

$$\text{␣ xx\#xx } q_8 \text{␣}$$

$$\text{␣ xx\#xx ␣} q_{\text{accept}} \text{␣}$$

0.1.3. 3.3 Modify the proof of Theorem 3.16 to obtain corollary 3.19 showing that a language is decidable iff some nondeterministic Turing machine decides it. (You may assume the following theorem about trees. If every node in a tree has finetly many children and every branch of the tree has finitely many nodes, the tree itself has finitely many nodes.)

0.1.3.1. language decidible ⇒ some nondeterministic Turing Machine decides it

Suppose L is a decidible language we can deduce that a deterministic turing machine, M, must decide it, but any deterministic TM is a subset of non deterministic TM, thus any NDTM decides such language.

0.1.3.2. some nondeterministic Turming Machine decides a language ⇒ language decidible

If a NDTM decides a language, that implies all braches of computation must halt

we'll convert such NDTM to DTM

we'll use 3 tapes for this approach

- first tape has the input
- second tape has the current computation
- third tape has current node address

a node address is a subset of the power set for the alphabet indexed, that is if the alphabet has 4 elements, such as a,b,c and d, each element would correspond to 0 to 3 as mapping.

the machine works as follows

- first put the input in the first tape
- copy the input from tape 1 to tape 2
- set the third tape to ε, the root node address
- run the computation in tape 2, until a reject or accept state happens
  - to run the current TM configuration in tape 2 make sure the last non empty position in tape 3 is used as a decision of next symbol to read, that is as input to transition function
    - if the computation is rejected modify tape3 to visit another possible branch in a BFS fashion

- if the computation is accepted, accept and end the computation

because the set of addresses for each node in the computation history must be finite eventually all the possibilities will be seen, thus the computation must halt

we succesfully created a DTM that has the same language as NDTM in the beginning, and the following is true as well

- if all branches of computation are exausthed the string is rejected
- otherwise is accepted

thus the language must be decidible ■

0.1.4. 3.4 Give a formal definition of an enumerator. Consider it to be a type of two-tape Turing machine that uses its second tape as the printer. Include a definition of the enumerated language.

$$E :=< \Gamma, \Sigma, q_{\text{start}}, Q_{\text{print}}, Q_{\text{accept}}, Q_{\text{reject}}, Q, \delta :: Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\} \times \Sigma_{\varepsilon} >$$

 We use S as a special motion where the machine does nothing, but is equivalent to $L \to R$ or $R \to L$

The machine work as follows

- for an alphabet $\mathbf{\Sigma}$ the machine will generate a list of every possible string available according to size
- for each string generated according to size run the machine and check if $\boldsymbol{Q}_{\textbf{print}}$ state is reached, the string are separated by markers, such as '#'
  - if the answer is yes move the cursor to closest leftmost marker to the $\boldsymbol{Q}_{\textbf{print}}$ state and copy the string betwwen the marker and such state to the next tape
    - mark each end of string in the second tape with a symbol and each string between such symbol is the printed value.
  - Move the work tape cursor to the next marker and move the printer cursor to the end of the tape and create a marker there as well

 repetion may occur, but that's allowed

0.2. 3.5) Examine the formal definition of a Turing machine to answer the following questions, and explain your reasoning.

0.2.1.1. a. Can a Turing machine ever write the blank symbol ␣ on its tape?

note, $\mathbf{\Sigma}$ doesn't contain the ␣ symbol as such the symbol must be in the $\mathbf{\Gamma}$ alphabet, every symbol in this alphabet is an allowed symbol for being written in the tape, so the answer is yes a turing machine can write the ␣ sybol.

0.2.1.2. b. Can the tape alphabet $\mathbf{\Gamma}$ be the same as the input alphabet $\mathbf{\Sigma}$?

never because ␣ always belong to $\mathbf{\Gamma}$ as such $\mathbf{\Sigma}$ can never the same set.

0.2.1.3. c. Can a Turing machine's head ever be in the same location in two succesive steps?

Yes, just a matter of the first transition be a L-transition, the cursor moves one tape element left, and the other must be a R-transition also a R-transition and then a L-transition are possible sequence of configurations as well.

0.2.1.4. d. Can a Turing machine contain just a single state?

no, even if q_{accept} were the same as q_{start} there's the imposition that $q_{\text{reject}} \neq q_{\text{accept}}$ and also note that Q is a set of states, that is each of it elements must be a state, can't be the $\emptyset$ because is a set, not a state as such a turing machine must always have at least two states.

0.2.2. 3.6 In Theorem 3.2.1, we showed that a language is Turing-recognizable iff some enumerator enumerates it. Why didn't we use the following simpler algorithm for the forward direction of the proof? As before, $s_1, s_2, \ldots$ is a list of all string in Σ^*.

$E =$ "Ignore the input.

1. Repeat the following for $i = 1, 2, 3$
2. Run M on s_i.
3. If it accepts, print out s_i"

the trick here is noticing that the enumerator need to deal with not halting and also printing every valid string. If we print sequentially every possible string in Σ^* the machine might never halt for such a string and as such we can't validate that E enumerates R. But by employing the trick of testing in parallel for each input if a tape never halt that won't be a problem because the other's will continue to work.

0.2.3. 3.7 Explain why the following is not a description of a legitimate Turing machine.

$M_{\text{bad}} =$ "On input <p>, a polynomial over variables $x_1, \ldots, x_k$:

1. Try all possible settings of $x_1, .., x_k$ to integer values
2. Evaluate p on all of these settings.
3. If any of these settings evaluates to 0, accept; otherwise reject. "

note that the input alphabet is infinite, but in the turing machine description we assume that each set must be finite, but beacuse the input alphabet is non terminating the memory also would be which would also force time of computation to be infinite and as such it would never halt, so is not an algorithm and because TM's and algorithms are the same thing such device can't possibly be a turing machine.

0.2.4. 3.8 Give implementation-level description of Turing machines that decide the following languages over the alphabet {0,1}.

0.2.4.1. a. {w | w "contains an equal numbers of " 0s "and" 1s}

M = "On input <w>, $w \in \{0,1\}^*$:

1. Copy all the 1's to a tape and all the 0's to another tape
2. Run in parallel each tape going always to the right
3. If at any moment a cursor hits a $\sqcup$ halt that tape, if the other didn't hit such symbol reject otherwise accept"

0.2.4.2. b. {w | w “contains twices as many 0s and 1s”}

M = "On input <w>, $w \in \{0,1\}^*$:

1. Copy all the 1's to a tape, T_a, and all the 0's to another tape T_b
2. Consider the following possible transitions

2.1. If T_b hits a 0, execute a R-transition, T_a doesn't move the cursor, a S-transition happens to T_a instead

2.2. If T_b hits another 0, T_a now executes a R-transition

2.3 If T_b hits $\sqcup$ check if T_a would also hit the same symbol with a R-transition, if yes then accept otherwise reject

2.4 cursor of T_b moves back to 2.1"

0.2.4.3. c. {w | w does not contain twice as many 0s as 1s}

M = "On input <w>, $w \in \{0,1\}^*$:

1. Copy all the 1's to a tape, T_a, and all the 0's to another tape T_b
2. Consider the following possible transitions

2.1. If T_b hits a 0, execute a R-transition, T_a doesn't move the cursor, a S-transition happens to T_a instead

2.2. If T_b hits another 0, T_a now executes a R-transition

2.3 If T_b hits $\sqcup$ check if T_a would also hit the same symbol with a R-transition, if yes then reject otherwise accept

2.4 cursor of T_b change state to 2.1"

0.2.5. 3.10 Say that a write-once Turing machine is a single-tape TM that can alter each tape square at most once (including the input portion of the tape). Show that this variant Turing machine model is equivalent to the ordinary Turing machine model.(hint: As a first step, consider the case whereby the Turming machine may alter each tape square at most twice. Use lots of tape.)

To prove this we need to show that a write-once TM recognizes the same language as an ordinary TM.

the constraint imposed in the question implies the following

$$\delta_{\text{wo}}(q_i, b_u) = (q_j, b_v, T) \text{ and } \delta_{\text{wo}}(q_i, b_u) \neq (q_j, b_w, T) \forall w \neq v$$

we need a way that would allow rewriting a square multiple times if needed, note that is the position that is constraining us, but we could copy the symbol and paste it

now let’s consider a transition function for the ordinary TM

$$\delta(q_i, b_u) = (q_j, b_v, T)$$

where T is a R,L,S- transition

 S-transition means don't move the cursor

let's consider a single tape position after a sequence of configurations

$$\text{␣ ab0 } q_j \text{ def ␣}$$
$$\text{␣ ab } q_r \text{ 1def ␣}$$
$$\text{␣ ab2 } q_j \text{ def ␣}$$

note that for 0,1,2 the position is the same

then to get equivalent behavior in the writing only TM all we have to do for any modified symbol is as follows

–1 when resuming/starting make sure to move every string one position to the right and place a mark, '#' in the starting position.

–2 for every modified symbol that is not a mark, put a dot in the top of the position

–3 if an extra modification is required , that is the symbol with a dot will be overwritten, do as follows, pause the main loop

–3.1 move the cursor back to the closest mark, copy everything to the right of the mark until a ␣ is found

–3.2 now go back to 1

–4 if cursors hits q_{accept} or q_{reject} stop the computation

0.2.6. 3.11 A Turing machine with doubly infinite tape is similar to an ordinary Turing machine, but its tape is infinte to the left as well as to the rigmt. The tape is intially filled with blanks except for the portion that contains the input. Computation is defined as usual except tha the head never encounters an end to the tape as sit moves leftward. Show that this type of Turing machine recognizes the class of Turing-recognizable languages.

We need to show that the infinte TM is equivalent to a recognizer TM to prove that we need to revise what it means for a TM to be a recognizer it means that the machine might not halt for every input.

we now show that such infinite TM may not halt for every input and as such is a recognizer TM as well

we know that a enumerator TM can be used to classify a language as Turing recognizable as such we could create an enumerator for the infinite TM.

and if that is possible it must mean such infinite TM is a recognizer according to the theorem that every enumerator TM is equivalent to a recognizer TM.

first let's identify the constraint for this TM

$$\delta_{\text{q0}}(q_i, b_u) = \left(q_j, b_v, L\right) \text{ where } q_0 \text{ is the first tape position}$$

in other words a L-transition in the beginning of the tape must allow a new tape position that is not the beginning

and in a normal TM such transition should result in the cursor at the same position and possibility of symbol being overwritten as well

to convert an infinite TM to a recognizer do as follows

- move the input one step to the right and mark the initial position with ‘#’ and the final position with the same marker but with a dot in the top
- whenever a L-transition happens in the position Immediately to the right of the marker do as follows
- copy everything from the right of the marker until the marker with the dot is hit by the cursor
- paste the copied contents to the left of the marker, add a dot to the marker without a dot and create a marker,without a dot, in the position Immediately to the left of the first copied position
  - make sure to offset the cursor enough from the marker so to not overwrite it!