**Theory of Languages and Automata - Fall 1403**

**Exercise Number 9 + Answers**

**Instructor of this collection:**

**Mostafa Kermaninia**

**Email: mostafakermaninia@gmail.com**

**Submission Date: December 15 (25th of Azar)**

---

**Q1) Mostafa works at a company that builds standard Turing machines! To satisfy his boss, he builds two new and advanced Turing machines using the entire company's budget and believes that the power of these new machines is greater than that of the standard Turing machine. Prove that the Turing machines he built do not differ from the standard Turing machine in terms of power and that he is mistaken.**

**a) A Turing machine that has K+1 tapes (where K is the size of the tape alphabet) such that initially the input is written on the (K+1)-th tape, writing on this tape is not allowed, and on the i-th tape ($1 \le i \le K$) only the i-th symbol of the tape alphabet can be written or erased. (10 points)**

- **Constructing a Standard Turing Machine from the Mentioned Machine:**

We move all the tape heads simultaneously. Initially, the contents of each cell of the standard Turing machine are set according to the symbol on the corresponding tape of the new machine and in the same position as the standard machine's tape. Eventually, specific cells on each of the K tapes contain content. For example, if on the standard Turing machine's tape we move to the right and the content of that cell does not change, then in the K+1-tape machine, all tape heads are moved one position to the right. However, if the content of the cell changes, on that tape, among the K tapes that have content in the cell that the head points to, the cell is cleared, and based on the new content that we want to write, we move to the corresponding tape and fill that cell. Then, all tape heads are moved one position to the right. Similarly, for moving to the left, all tape heads are moved one position to the left.

- **Constructing the Mentioned Machine from the Standard Turing Machine:**

Place the contents of all tapes onto a single standard tape, separating them with unique characters. Additionally, store the positions that each tape's head points to on the same tape, for example, by marking the content of that cell with a dot on a specific symbol. (This is exactly like constructing a multi-tape machine from a standard machine. Since the equivalence of the standard Turing machine and the multi-tape Turing machine has been proven, this approach—using K+1 tapes where each tape only writes one type of symbol—is correct.)

**b) A Turing machine that, instead of a one-dimensional tape, has a two-dimensional grid where both rows and columns are infinitely long in one direction.** (10 points)

- **Constructing the Mentioned Machine from the Standard Turing Machine:**

**Method1:**

In the standard Turing machine, before accessing the content of a cell, write its location. Specifically,

first write the row number, then after a # symbol, write the column number, and finally, after another #, write the content of the cell. Additionally, the current head location can be stored on a second tape (since we know that multi-tape machines are equivalent to standard machines). Each time we want to move in a direction, we first check if that location is written on our tape. If it is, we read the content there and apply the required change. If not, the content is blank, and if we intend to write something there, we add that location and its content to our tape. Finally, we also update the head location.

**Method2:**

Use a pairing function to map each cell of the two-dimensional grid to a single cell on a one-dimensional tape. For example, the Cantor pairing function is one of the well-known functions for mapping pairs of natural numbers to unique natural numbers. This function is particularly suitable when both coordinates $(i, j)$ are non-negative. The Cantor function is defined as:

$f(i,j) = j + (i + j + 1)(i + j)/2$ This function maps each pair of natural numbers $(i, j)$ to a unique natural number. Thus, we build a Turing machine that takes two numbers $k_1, k_2 \in N$ and, according to the Cantor function, outputs a natural number. Now, a cell in the two-dimensional grid that the head points to is denoted as $(k_1, k_2)$. We input this into f and obtain a number. We then move to the cell on the one-dimensional tape that corresponds to the output of f. We place the content of the $(k_1, k_2)$ cell of the two-dimensional tape in that cell. For all cells in the two-dimensional tape that have content, we repeat this process. For the transitions, the following occurs:

- If on the two-dimensional tape we move left, on the one-dimensional tape we move to the cell $f(k_1 - 1, k_2)$.
- If on the two-dimensional tape we move right, on the one-dimensional tape we move to the cell $f(k_1 + 1, k_2)$.
- If on the two-dimensional tape we move down, on the one-dimensional tape we move to the cell $f(k_1, k_2 - 1)$.
- If on the two-dimensional tape we move up, on the one-dimensional tape we move to the cell $f(k_1, k_2 + 1)$.

- **Constructing the Standard Turing Machine from the Mentioned Machine:**

It is sufficient to use only the first row of the two-dimensional grid to simulate the behavior of the standard Turing machine.

**Q2) In this section of the lesson, the equivalence of the standard Turing machine with other types of Turing machines was discussed. However, machines that are equivalent in power to the standard Turing machine differ from it in some aspects, two examples of which we will examine:**

**a) We want to add two binary nn-bit numbers AA and BB together and write their sum on the tape of a Turing machine. A bad method to do this on a standard Turing machine is to first input the numbers as A#BA\#B on the tape, then subtract one unit from BB and add it to AA each time until finally, when BB reaches zero, A+BA + B forms at the beginning of the tape. However, this solution is exponential in order and is not a good approach. Prove that this problem can be solved in the standard Turing machine in the best case with O(n2)O(n^2), while in a multi-tape Turing machine, it can be solved with O(n)O(n). (Explain the number of tapes used in the multi-tape machine and the type of head movements.) (8 points)**

- Solution with $O(n^2)$ on a standard Turing machine:

Initially, the numbers are placed on the tape with a separator such as #. Now, another # is added, and the third part of the tape is considered for writing the sum of the numbers (on the tape, we have something like #A#B#). Now, exactly like the usual method of adding two binary numbers, we start from the LSB of number B, add it to the LSB of number A, and write the result in the position designated for the corresponding bit in the sum. Details such as considering the carry and whether the sum becomes n-bit or n+1 bit naturally exist, but eventually, for the addition of each bit of the two numbers, the head moves O(n). Therefore, adding all n bits will have an order of $O(n^2)$. (The $O(n^2)$ solution is the best possible case because, in any case, for adding two numbers, we must individually examine the corresponding bits. For each movement from the i-th bit of number A to the i-th bit of number B, given that the numbers are n-bit, the head must perform at least O(n) movements, and this must be repeated at least O(n) times.)

- Solution with O(n) on a three-tape Turing machine:

Initially, number A is written on the first tape, number B is written on the second tape, and the third tape is used for writing the sum. Then, all the heads are simultaneously moved to the right. The first head reads the bit of interest from number A, the second head reads the corresponding bit from B, and the third head writes the sum (and any possible carry) in the corresponding position on the third tape. This addition operates just like manually adding two binary numbers, and it is only necessary to traverse the numbers once (of length n), performing the addition bit by bit and writing the result on the third tape. Thus, this operation will ultimately be of order O(n).

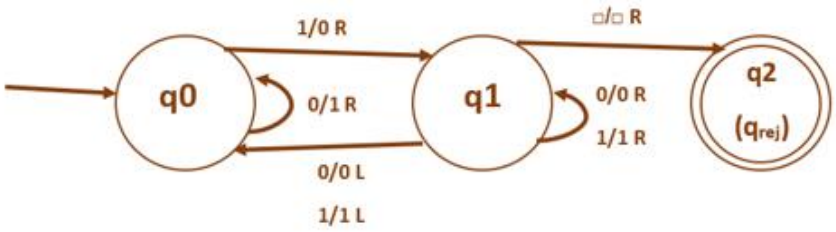
**b) We have a nondeterministic Turing machine M defined as follows:**

$$M = \{\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, \square\}, \delta, q_0, \{q_2\}, \{\}\}$$

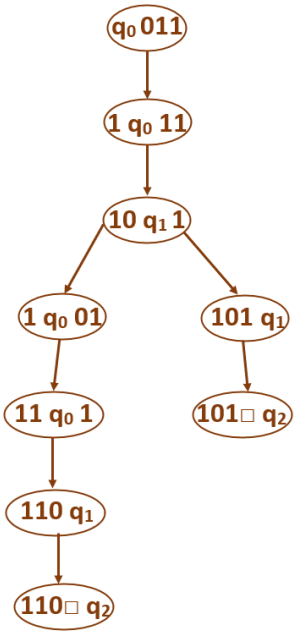
| $\delta$ | 0 | 1 | $B$ |
|---|---|---|---|
| $q_0$ | $\{(q_0, 1, R)\}$ | $\{(q_1, 0, R)\}$ | $\emptyset$ |
| $q_1$ | $\{(q_1, 0, R),\ (q_0, 0, L)\}$ | $\{(q_1, 1, R),\ (q_0, 1, L)\}$ | $\{(q_2, B, R)\}$ |
| $q_2$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |

**If the input for this Turing machine is 011, draw the tree of all configurations this machine reaches until it halts. (7 points)**

First, to make the task easier, draw the diagram of this Turing machine based on the transition table:




Now, based on the initial tape input and the fact that, in the standard Turing machine, the head is on the first cell of the tape, the tree of all configurations can easily be derived:



**Q3) Mostafa, after being fired from the company he worked for, still believed he was a master in building Turing machines. So, using all the money he had saved, he bought a black spray paint and created a Turing machine called the "Monochromatic Turing Machine." This machine is identical to the standard Turing machine in every aspect, except for two limitations:**

1. **Since there is only one black spray paint, each cell on the tape can either remain blank or be painted black. (Thus, the tape alphabet is $\Gamma = \{\square, \blacksquare\}$).)**
2. **There is no tool for erasing the spray paint, so we can only replace a blank cell (□) with a black character (■), but not vice versa.**

**Prove that the computational power of the monochromatic Turing machine is equivalent to that of the standard Turing machine. (15 points)**

**Building a Standard Turing Machine Using a Monochromatic Machine:**

To begin, we need to simulate the alphabet of the standard Turing machine on the monochromatic machine. Without loss of generality, assume the alphabet is $\Sigma = \{a_1, a_2, \ldots, a_n\}$. Each of the n symbols in the alphabet can be encoded as an n-bit string as follows:

$a_1 =$ □□… □ ■

$a_2 =$ □□… ■ □

…

$a_n =$ ■□… □ □

Given that the length of each letter in this encoding is n, the contents of the standard Turing machine's tape can be represented on the monochromatic machine's tape without requiring additional separator characters or risking the mixing of symbols.

Next, we add two extra positions to this encoding:

- The first position, if marked black, indicates that the character is invalid.
- The second position, if marked black, indicates that the head of the standard Turing machine is currently positioned on that character.

Thus, the equivalent length of each symbol in the alphabet becomes n+2.

To simulate the movements of the standard Turing machine (e.g., a right movement), we proceed as follows:

- Since ■ cannot be replaced with □, we copy all the tape's contents into the empty space of the tape.
- After copying each symbol, we invalidate the original symbol by marking its "invalid" position black.
- During the copying process, if the instruction is a right movement, we mark the "head" position of the character to the right of the character previously under the head.

- If necessary, we overwrite the original character with the n-bit encoding of the new character being written.

We continue this process with the new tape contents, where no character is invalid. For a left movement, we follow the exact same steps to update the character under the head and move the head.

**Building a Monochromatic Machine Using a Standard Turing Machine:**

To simulate the monochromatic machine with a standard Turing machine, we simply designate one character from the alphabet as ■ and never overwrite ■ with a blank character (□).

This approach demonstrates that the monochromatic Turing machine and the standard Turing machine are computationally equivalent, as each can simulate the other.

**Q4) We have a Turing machine with a jumping head. That is, whenever it wants to move to the right on the tape, it suddenly jumps A units to the right, and whenever it wants to move to the left, it suddenly jumps B units to the left. (A and B are natural numbers.)**

**Part (a): If AA and BB are coprime, is the power of this machine equal to the standard Turing machine? Explain. (10 points)**

**Hint: The equation $Ax - By = i$ has solutions for those i that are divisible by gcd(A,B).**

**Yes, they are equal.**

- **Simulating the jumping machine with the standard Turing machine:** To move to the right, simply move one unit to the right on the tape, change the content of the tape as needed, and then move A−1A-1 additional units to the right, ignoring the content of the cells. Similarly, to move to the left, perform the same process for B−1B-1 additional units.
- **Simulating the standard Turing machine with the jumping machine:** Based on the hint and the coprimality of A and B, there exist integers x1 and y1 such that $Ax_1 - By_1$. Similarly, for the equation $By - Ax$, there exist x2 and y2 such that $Ax_2 - By_2 = -1$.

Therefore:

  - To move one unit to the right, the jumping machine performs x1 jumps to the right (of size A) and y1 jumps to the left (of size B).
  - To move one unit to the left, the jumping machine performs x2 jumps to the right and y2 jumps to the left.

Thus, the jumping machine can simulate the standard Turing machine and vice versa, proving their computational equivalence.

**Part (b): If there is no specific condition on AA and BB, is the power of this machine equal to the standard Turing machine? Explain. (10 points)**

**Yes, they are equal.**

- **Simulating the jumping machine with the standard Turing machine:** This is done in exactly the same way as in part (a), by making A−1 or B−1 additional steps in the desired direction.
- **Simulating the standard Turing machine with the jumping machine:** In this case, the jumping machine cannot simply move exactly one unit to the left or right. Instead, for gcd(A,B)=k, the jumping machine can move nk units to the left or right, where n is a natural number.

To simulate the standard Turing machine:

  - Treat the initial position of the head as the "zero-th" cell.

- Consider the nk-th cell of the jumping machine's tape as corresponding to the n-th cell of the standard Turing machine's tape.

With this re-indexing of the tape, any operation performed on the standard Turing machine can be replicated by the jumping machine, ensuring equivalence.

**Q5) Mostafa, after his success in building the "monochromatic" Turing machine, decided to start his own Turing machine company. For the first product, he plans to implement a computer using a multi-tape Turing machine. On the first tape, representing the computer's memory, numbers are written. On the second tape, commands separated by # are written, which the computer reads and executes sequentially. Help Mostafa execute the following commands with his Turing machine (you can use additional tapes or any type of Turing machine that has been proven equivalent in power to the standard Turing machine). (15 points)**

**Commands:**

1. **#inc#: Increases the number at the current head position by 1. (If the cell under the head is empty, consider it as 0 and convert it to 1.)**
2. **#right#: Moves the memory head one cell to the right.**
3. **#store n i#: Writes the number n in the i-th cell of the memory tape.**
4. **#add i j k#: Writes the sum of the numbers in the i-th and j-th cells of the memory tape into the k-th cell. (If a cell is empty, consider its value as 0.)**
5. **#jump a#: Jumps to the a-th command written on the instruction tape and continues executing from there.**

**Implementation:**

**Tapes Setup:**

1. **Tape 1:** Memory.
2. **Tape 2:** Instructions.
3. **Tape 3:** Address of the memory head.
4. **Tape 4:** Address of the instruction head (acts as the program counter, PC).
5. **Tape 5:** Scratchpad for temporary computations.

**Instruction Reading:**

- The head of Tape 2 moves sequentially to read instructions.
- Non-deterministically check which type of command it is (or deterministically scan all commands in order).
- After identifying the command, execute it and increment the value on Tape 4 by 1 to move to the next instruction.
- For #jump#, update the value on Tape 4 directly to the address of the target command.
- Whenever the memory head moves (left, right, or to a new position), update the value on Tape 3 to reflect the new address.

**Executing Commands:**

1. **#inc#:**
    - Increment the number under the memory head on Tape 1.
    - If the cell is empty, write 1.
    - Use the "stay" capability to leave the memory head in place.
2. **#right#:**
    - Move the memory head on Tape 1 one step to the right.
    - Update Tape 3 to reflect the new address of the memory head.
3. **#store n i#:**
    - Move the memory head on Tape 1 to the start of the tape.
    - Navigate right, updating Tape 3, until reaching the i-th cell.
    - Write the number n at the destination cell.
4. **#jump a#:**
    - Similar to #store, but applied to Tape 2.
    - Move the instruction head to the start of Tape 2.
    - Navigate right, using Tape 4 to track addresses, until reaching the a-th command.
    - Execute the command from there.
5. **#add i j k#:**
    - Navigate to the i-th cell and copy its value to the scratchpad (Tape 5).
    - Move to the j-th cell and append its value to Tape 5, separating the two numbers with #.
    - Perform the addition:
        - Decrement the first number on Tape 5 bit by bit, adding each bit to the second number.
        - Use binary addition: Flip all 1s to 0s until reaching the first 0, which is flipped to 1.
        - Similarly, for subtraction: Flip all 0s to 1s until reaching the first 1, which is flipped to 0.
    - Navigate to the k-th cell and write the result.

**Q6) We have a nondeterministic Turing machine with a tape that is infinite in both directions. Initially, the tape is completely empty, except for one cell that contains the character $. The machine's head is initially positioned on one of the empty cells.**

**Part (a): Explain the steps to locate the $ character. (5 points)**

We move nondeterministically in both the right and left directions, searching for $. Whenever $ is found in any path, the machine transitions to the accept state immediately.

**Part (b): If this machine were a deterministic standard Turing machine, would it still be able to find $? (Note that if anything is written on the tape during the search, it must ultimately be erased, and the head must end on $.) If yes, explain the method. (10 points)**

**Yes, in this case, if we were to only move in one direction from the start, we might choose the wrong direction and never reach $. Therefore, we search oscillatingly and use two markers, L and R, to indicate how far we have searched in each direction.**

**Method:**

1. First, move one cell to the left and place the marker L.
2. Then, move two cells to the right and place the marker R.
3. As long as $ is not found:
    - Each time you encounter the marker L, move left and shift the marker L one cell further to the left.
    - Then, move right until encountering the marker R and shift the marker R one cell further to the right.
4. Once $ is found:
    - Move left until reaching L, erasing it.
    - Then, move right until reaching R, erasing it as well.
5. Afterward, since $ is guaranteed to be to the left of the marker R, continue moving until reaching it again.

**Q7) One of the byproducts of Mostafa's company is the same standard Turing machine, but the challenges of the standard Turing machine haven't left him alone here either. Customers have reported that after one month, the left-moving operation of the Turing machine's head malfunctions, and whenever we try to move the head to the left, it goes straight to the start of the tape.**

**Part (a):**

**Mostafa claims that this problem does not reduce the power of the Turing machine, and the machines produced by his company are still as powerful as the standard Turing machine. Prove that his claim is correct. (7 points)**

- **Simulating the faulty machine using a standard Turing machine:** It suffices to first add a marker to the tape and shift the entire input one cell to the right. After that, for every operation where the head moves to the start of the tape, the machine can move left until it reaches the marker at the start of the tape.
- **Simulating a standard Turing machine using the faulty machine:** Since all operations except leftward movement are still available, the problem reduces to simulating the leftward movement correctly. To do so:

● **1:** Mark the current cell with a dot (.) and move to the start of the tape, marking the first cell with a hat (^).

Now, in a loop, gradually move the hat one cell to the right until it is directly to the left of the cell marked with the dot. At that point, the goal of finding the left-hand neighbor of the initial cell is achieved. To achieve this:

● **2:** If the current cell doesn't have a hat, move right until encountering a cell with a hat, then move one more cell to the right. (For instance, during the first loop iteration, the first cell has the hat, and the machine simply moves one cell to the right to bypass it.)

● **3:**
**Case 1:** If the current cell has no marker, it means the left-hand neighbor of the original cell (marked with a dot) hasn't been reached yet. Move the hat one cell to the right by:

  - Marking the current cell with a hat.
  - Returning to the start of the tape.
  - Moving right until reaching the previous hat marker, removing it, returning to the start of the tape, and returning to step 2.

**Case 2:** If the current cell has a dot marker, it means the left-hand neighbor of the dot-marked cell has been successfully identified with a hat. At this point:

  - Remove the dot marker.
  - Return to the start of the tape, move right until reaching the hat marker, which represents the left-hand neighbor of the initial head position.

**Part (b):**

**After a few months, the malfunction of the company's machines reaches its peak, to the extent that the head of these machines can only move to the right and cannot move left at all. In response to customer complaints, Mostafa claims that this problem also does not reduce the power of the Turing machine. Do you think Mostafa's claim is correct, or should he declare bankruptcy? If you disagree with his claim, briefly describe what languages the company's machines can recognize at this stage. (3 points)**

**Mostafa's claim is incorrect, and he should declare bankruptcy.**

The company's machines, at this stage, are unable to move backward, meaning that anything written on the tape can no longer be accessed or read. As a result, these machines behave like a **DFA** (deterministic finite automaton) that reads the input from the beginning, transitioning between states based on the input and the current state. Thus, these machines can only recognize **regular languages** and are less powerful than the standard Turing machine.