Homework 4

Q1.

1- *Design a UTM using a RATM that has the capacity to simulate any other RATM as its input. Describe your UTM in detail , namely the contents and structure of all its registers and its external tape.*

Solution:

The Universal Turing Machine (UTM) is a machine that can simulate any other Turing Machine. A RATM is a variation of the standard Turing machine where the read/write head can move to any cell on the tape directly (random access), instead of moving step by step.

The Universal RATM should be able to store the definition of any other RATM to operate it. The tape of the Universal RATM will store the description and state of the RATM to be operated. This description will include the RATM's transition function, its current state, and the contents of its tape.

There are a couple of special register to hold some information while operating the RATM.

The Universal RATM should have a special register that is storing the address register (AR) which stores an address of any cell of the tape. It should have a data register (DR) which is used to read or write data to the cell which is pointed by AR. Instruction register (IR) is used to hold the current instruction that is Univesal RATM is executing. State register (SR) is used to hold the current state of RATM that is being simulated. Temp register (TR) is used for temporary register like register in a CPU.

Tape of the Universal RATM has special parts for different actions. They are Transition Function Section which stores the description of RATM that is being simulated. Tape Content Section which stores the contents of the RATM which is being executed.

Operation of the Universal RATM works like this:

1. AR is set to the first of the state of the RATM is loaded into the SR.
2. The AR is set to the start of the TFS, and the universal RATM starts searching for a tuple that matches the current state and symbol of the RATM.
3. If the searched state and symbol is found it is loaded to into SR, DR and AR.
4. The AR is set to the address in the TCS that matches the address in the AR, and the new symbol is written to the tape.
5. If the new state is not the halt state, the process repeats from step 1.

This Universal RATM is capable of simulating any RATM that it’s fed as a input.

Q2.

5.4.1. Say that Turing machine M **uses** k tape squares on input string w if and only if there is a configuration of M, $(q, u\underline{a}v)$, such that $(s, \triangleright\underline{\sqcup}w) \vdash_M^* (q, u\underline{a}v)$ and $|uav| \geq k$.

(a) Show that the following problem is solvable: Given a Turing machine M, an input string w, and a number k, does M use k tape squares on input w?

(b) Suppose that $f : \mathbf{N} \mapsto \mathbf{N}$ is recursive. Show that the following problem is solvable: Given a Turing machine M and an input string w, does M use $f(|w|)$ tape squares on input w?

(c) Show that the following problem is undecidable: Given a Turing machine M and an input string w. does there exist a $k \geq 0$ such that M does not use k tape squares on input w? (That is, does M use a finite amount of tape on input w?)

Solution:

(a) The given problem is solvable because there is an algorithm that solves this problem. A Turing machine M = (Q, Σ, δ, s, H) has only $|Q| \times (k-1) \times |\Sigma|^{k-1}$ distinct configurations which M uses fewer than k tape squares. This is the number of all state and tape combinations we can have with k-1 tape squares, Q states and Σ distinct alphabet symbols. Hence, if we simulate M on w for $|Q| \times (k-1) \times |\Sigma|^{k-1} + 1$ steps there are three possibilities. Either the TM M halts before finishing these many steps, or it reaches a configuration twice and starts an infinite loop or enters a configuration using k or more squares. In the first possibility, the machine halts before reaching the kth tape square and it never reaches the kth tape square. In the second possibility, the machine gets stuck in an infinite loop before the kth tape square, and it never reaches the kth tape square. In the last possibility, it reaches the kth tape square. Hence the problem is solvable.

(b) In this question, I think there is a typo in the question because f may not return a number as a result. Hence it is illogical to check $f(|w|)^{th}$ tape square. I assumed that the intention was to check the $|f(w)|^{th}$ tape square. Function f is recursive means that there is a Turing machine N which computes f. We can have a Turing Machine M' with three tapes. Firstly, it will have the input w on its first tape, and it will compute the function f(w) on the second tape. Then it will calculate the |f(w)| on the third tape. Lastly it will calculate the (M, w, |f(w)| on the first tape using the Turing machine that we described in section a. The result of the M' will be the same as the Turing machine that we described in section a.

(c) We can show that the given problem is undecidable with contradiction. Suppose that the given problem is solvable. Then there should be a Turing machine F that checks if a Turing machine M uses a finite amount of tape on input w. If it uses a finite amount of tape F accepts the input "M" and "w" if not F rejects.

Also, we can construct a Turing machine M' which solves the halting problem with F on input "M" and "w" like this:

- Operate TM F on "M" and "w". If F rejects the input "M""w" this means that M uses infinite number of tape squares on input w. In this case reject the input.

- If F accepts the input “M””w” (M uses finite number of square tapes on input w) then M’ operates M on input w until it halts or it reaches a configuration for a second time (which means a infinite loop). If M halts accept, else reject.

M’ solves the halting problem but we know that the halting problem is unsolvable. Thus M’ cannot exist. Therefore, TM F cannot exist. This proves that the given problem is undecidable.

5.4.2. Which of the following problems about Turing machines are solvable, and which are undecidable? Explain your answers carefully.

(a) To determine, given a Turing machine M, a state q, and a string w, whether M ever reaches state q when started with input w from its initial state.

(b) To determine, given a Turing machine M and two states p and q, whether there is any configuration with state p which yields a configuration with state q, where p is a particular state of M.

(c) To determine, given a Turing machine M and a state q, whether there is any configuration at all that yields a configuration with state q.

(d) To determine, given a Turing machine M and a symbol a, whether M ever writes the symbol a when started on the empty tape.

(e) To determine, given a Turing machine M, whether M ever writes a nonblank symbol when started on the empty tape.

(f) To determine, given a Turing machine M and a string w, whether M ever moves its head to the left when started with input w.

(g) To determine, given two Turing machines, whether one semidecides the complement of the language semidecided by the other.

(h) To determine, given two Turing machines, whether there is any string on which they both halt.

(i) To determine, given a Turing machine M, whether the language semidecided by M is finite.

Solution:

a) This problem is undecidable. If we assume that it is solvable then there would be a Turing machine M’ that is able to solve it. If the Turing machine M’ computes with input (M, q, w) whether M ever reaches the state q. This machine could be used to solve the halting problem as well. If we replace the state q with halting states, we can solve the halting problem which is not possible. Thus, this problem is undecidable.

b) This problem is undecidable. If we assume that it is solvable then there would be a Turing machine R that is able to solve it. But this machine could be used to solve the halting problem of any arbitrary TM M and input w. To achieve this, we can get the M than modify it as M’ such as M’ operates same as M. But we add another two states to M namely p and q which are not in the state table of M. We also add two rules in M. The first rule is when M starts to operate it first goes to state p then it continues with its normal execution. The second rule is if M goes to a halt state, then go to q state and halt. Now if we use TM R with M’ and states p and q the TM will accept M’ if M halts on input w. Thus, we can solve the halting problem which is not possible. This problem is undecidable.

c) This problem is solvable. We can create a TM M’ that takes a Turing machine M and its transition function δ as the input. It scans the transition function rules of M from left to right. If there is any rule such as δ (p, σ) -> (q, u) then the M’ accepts. If the machine reaches to a blank symbol without finding such a rule, then the M’ rejects. If the M’ accepts we can create a configuration such as (p, σ) because σ must belong to the Σ if it is in the transition function rules. Thus, in this case there is a configuration with state p which yields a configuration with state q, where p is a particular state of M.

d) This problem is undecidable. If we assume that it is solvable then there would be a Turing machine R that is able to solve it. This machine would take the Turing machine M and an empty tape as the input. It accepts if the M writes the symbol a on its tape. This machine could be used to solve the halting problem with the empty tape. Let T be a TM that takes a Turing machine M as input and modifies the transition function of M such as M writes an “a” (which does not belong to the alphabet of M) just before the halting states after that it continues to operate the M on empty tape. Now we can solve the halting problem of an TM M with the empty tape like this. We can give the M to the Turing machine T then we can give T_M as the input to the Turing machine R. If R accepts, then this means that M halts on the empty tape. Hence, we can solve the halting problem which is not possible. Thus, this problem is undecidable.

e) This problem is solvable. There is a Turing machine R that can solve it. The machine operates like this:
- Start to operate M on the empty tape. If M writes a nonblank symbol, accept.
- If it enters a configuration for a second time this means an infinite loop, reject.
- If it enters the same state again while it is reading blank and going right this means the machine is going to right infinitely and it won’t write any non-blank symbol. In this case reject.

Thus, the problem is solvable.

f) This problem is solvable. There is a Turing machine R that can solve it. The machine operates like this:
- Start to operate M on input w. If M moves its head left, accept.
- If M repeats a configuration for a second time this means an infinite loop, reject.
- If M moves to a position to the right of the end of w and enters two distinct configurations S and S' with the same state and same symbol being currently scanned. This means the machine is going to the right infinitely and it won’t go to left.

This means that the problem is solvable.

g) …

h) This problem is undecidable. If we assume that it is solvable then there would be a Turing machine R that is able to solve it. The TM R takes two Turing machines M_1 and M_2 as input. If this machine exists, we can use it to solve the halting problem of an arbitrary TM M. We can put M as both M_1 and M_2 as input to the TM R. If R accepts then M halts. Thus, R cannot exist and this problem is undecidable.

i) This problem is undecidable. If we assume that it is solvable then there would be a Turing machine R that is able to solve it.

5.4.3. Show that it is an undecidable problem to determine, given a Turing machine M, whether there is some string w such that M enters each of its states during its computation on input w.

Solution:

This problem is undecidable. If we assume that it is solvable then there would be a Turing machine R that is able to solve it. To show that we can reduce the undecidable problem of determining is there any input w on which a given TM M halts to this problem. To do this we need to modify M to M' like this:

M' will have a new symbol a which does not belong to alphabet of Turing machine M. and a new state q which does not belong to states of Turing machine M. Then we add rules to M. The first rule is all the halting state rules are replaced with state (a, q). Now if M attempts to halt it will go into state q and write a on its tape. We also add a rule such that if M' reads an a on its tape it will enumerate all its non-halt states and it will halt. Now we can use the R to solve the halting problem. If we want to determine is there any input w on which a given TM M halts, we can modify it to M' and feed the M' to R. If M halts on any input w, then the R will accept otherwise it will reject. R cannot exist; thus, the problem is undecidable.