Homework 2

Yutong Huang (yxh589)

Problem 1

To prove this problem is equivalent to showing that a multicore TM can be simulated by a single core, single tape TM. We can do this by showing that a single core, single tape TM can simulate a single core, multitape TM, and then show that a single core multitape machine can simulate a multicore multitape TM.

Proof. Say MM is a multicore, multitape TM that has k heads and k tapes. Similarly, SM is a single head k tape machine, and SS is a single head single tape mahcine.

Consider an arbitrary transition function on MM:

$$\delta_{MM}((q_{1m^*}, q_{2m^*}, \dots, q_{km^*}), (a_1, a_2, \dots, a_k)) = ((q_{1n^*}, q_{2n^*}, \dots, q_{kn^*}), (b_1, b_2, \dots, b_k), (L, R, L, \dots, R))$$

where q_{im^*} represents an arbitrary state on the i-th head before the transition, and q_{jn^*} represents the resulting state on the j-th head after the transition. Similarly, a_i represents the value read at the i-th head before transition, and b_j represents the value written to the j-th head. L, R, L, S ... are movement directions for each head respectively.

This function can be simulated on SM with the following transition function:

$$\delta_{SM}(q_{m1}, (a_1, a_2, \dots, a_k)) = (q_{m2}, (b_1, a_2, \dots, a_k), (L, S, S, S, \dots))$$

$$\delta_{SM}(q_{m2}, (b_1, a_2, \dots, a_k)) = (q_{m3}, (b_1, b_2, \dots, a_k), (S, R, S, S, \dots))$$

$$\dots$$

$$\delta_{SM}(q_{mk}, (b_1, b_2, \dots, b_{k-1}, a_k)) = (q_n, (b_1, b_2, \dots, b_k), (S, S, S, S, \dots), (S, S, S, \dots, R))$$

Therefore, any arbitrary transition function on MM can be simulated on SM.

Similarly, consider an arbitrary transition function on a SM:

$$\delta_{SM}(q_m, (a_1, a_2, \dots, a_k)) = (q_n, (b_1, b_2, \dots, b_k), (L, R, \dots, L))$$

On SS, we can use a special character such as # to seperate the tape in to multiple segments, representing multiple tapes. Then we can use an asterisk(*) to mark the cell that this simulated head is currently pointing at. Next we can construct a transition function as follows:

SS: on input=$w_1, w_2, \dots, w_n$

1. the input format would be $\#w_1^* w_2 w_3 \dots w_n \#\_^* \#\_^* \# \dots \#$, which simulates k tapes.
2. SS scans the tape from the first # to the (k+1)-th # (which is essentially a left end to right end sweep) to determine the value at each simulated heads (cells marked with a *).
3. then the machine sweeps throught the tape again to update values as described in SM's transition function.
4. if one of the heads shifts right to a #, then SM writes a $\_^*$ in its place and shift the content to the right of the # (including the #) to the right by 1 cell.
5. go back to step 2 and continue the simulation.

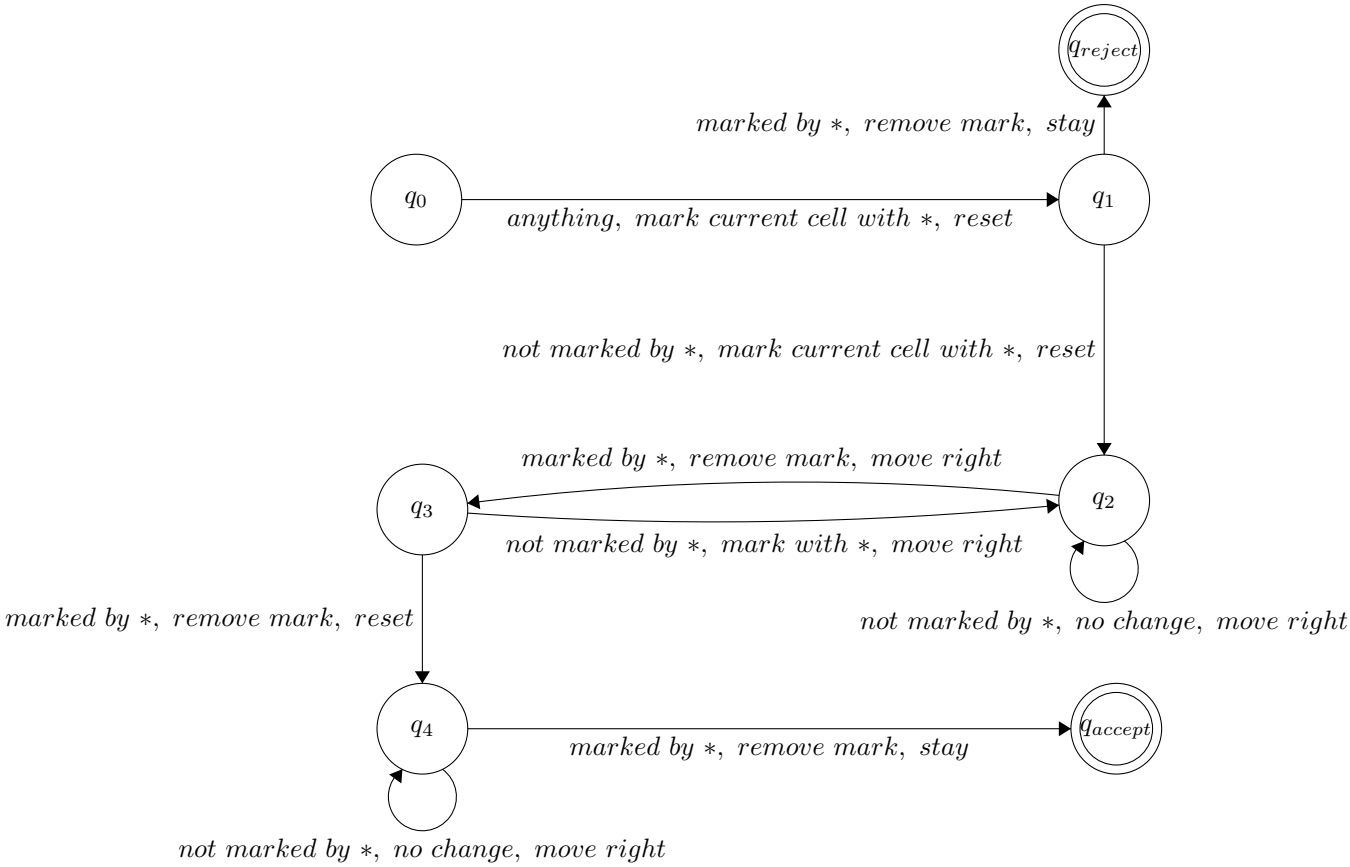
Therefore, any arbitrary transition function of SM can be simulated on SS.

Any arbitrary transition function of SM can be simulated on SS $\wedge$ any arbitrary transition function on MM can be simulated on SM $\implies$ SS can simulate all transition functions on SM that simulate MM $\implies$ SS can simulate any arbitrary transition function on MM.

□

Problem 2

To show that this machine can recognize all languages recognizable by a normal TM, we just need to show that we can emulate a left move on this broken machine. Below is a diagram that describes a procedure that move one cell to the left.



This procedure will find the cell to the left of current cell if current cell is not the leftmost cell, otherwise the procedure would reject and stay in place.

Proof. Assume an arbitrary language L is recognized by a normal TM.
$\implies \exists M$ that recognizes L.

Consider an algorithm M' which is exactly identical to M except that every left move is replaced by the above procedure. Then M' also recognizes L.

Therefore, any language recognizable to a normal TM is also recognizable to this broken machine.

□

Problem 3

(a)

Proof. Assume L_1 and L_2 are decidable $\implies \exists M_1, M_2$ that decides L_1 and L_2 respectively.

Consider an arbitrary language $l = ab$, and the following algorithm:

Algorithm 1: M

Input: ab
Run M_1 on a
if M_1 *rejects* **then**
  return reject
Run M_2 on a
if M_2 *rejects* **then**
  return reject
return accept;

Case 1: $l \in L \implies a \in L_1 \wedge b \in L_2$:
Both M_1 and M_2 accepts;
$\implies$ M accepts;

Case 2: $l \notin L$:
Case 2.1: $a \notin L_1 \wedge b \in L_2$
M_1 rejects $\implies$ M rejects;
Case 2.2: $a \in L_1 \wedge b \notin L_2$
M_2 rejects $\implies$ M rejects;
Case 2.3: $a \notin L_1 \wedge b \notin L_2$
Both M_1 and M_2 reject $\implies$ M rejects;

Therefore, M accepts if $l \in L$, rejects if $l \notin L \iff$ M decides L

$\implies$ L is decidable. □

(b)

Proof. Assume L_1 and L_2 are recognizable $\implies \exists M_1, M_2$ that recognize L_1 and L_2 respectively.

Consider the abouve algorithm again, and an arbitrary input $l = ab$

Case 1: $l \in L \implies a \in L_1 \wedge b \in L_2$:
Both M_1 and M_2 accepts;
$\implies$ M accepts;

Case 2: $l \notin L$:
Case 2.1: $a \notin L_1 \wedge b \in L_2$
M_1 rejects or infinite loop $\implies$ M rejects or infinite loop;
Case 2.2: $a \in L_1 \wedge b \notin L_2$
M_2 rejects or infinite loop $\implies$ M rejects or infinite loop;
Case 2.3: $a \notin L_1 \wedge b \notin L_2$
Both M_1 and M_2 reject or infinite loop $\implies$ M rejects or infinite loop;

Therefore, M accepts if $l \in L$, M rejects or infinite loop if $l \notin L$
$\iff$ M recognizes L $\iff$ L is recognizable. □