Homework 6

Walker Bagley

March 31, 2023

1. *Proof.* Let N be a Turing Machine with a doubly infinite tape. We proved in class that multitape TM's are equivalent to single tape TM's. The following is an implementation level description of an equivalent two tape TM M, which we know from class can be converted into a standard one tape TM.

   (a) With two singly infinite tapes 1 and 2, write the start symbol of tape 1 to be some symbol not in the alphabet of N (Σ), call it #

   (b) Follow the # with the input string and the infinitely many spaces that follow it

   (c) Write the start symbol of tape 2 to be # as well

   (d) On tape 1, read in # and move the head to the right

   (e) Simulate the rules of N with the following conditions:

      i. When the head of tape 1 is to the right of #, simulate rules of N as usual

      ii. If the head of tape 1 reaches # again, then switch to tape 2 and reverse the direction of rules on N, so any left shift is a right shift and vice versa

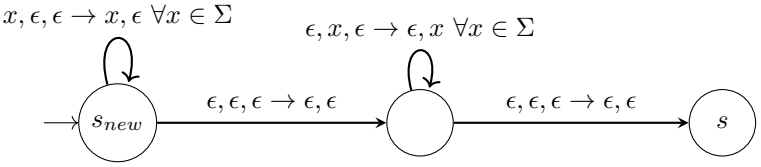
      iii. If the head of tape 2 reaches # again, then switch to tape 1 and return to regular direction of N rules

□

2. *Proof.* Consider a Turing Machine M and the following rules to contruct from M a 2PDA P:

   (a) For each state q in M, create an equivalent state q in P

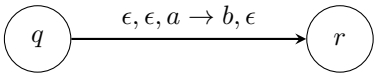
   (b) For the start state s of M, create a new start state s_{new} and transitions to s as shown below. This means we enter s with the input string in stack 2 in order as we pop from stack 2.



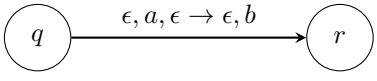
   (c) For q accept states in M, create equivalent accept states in P

   (d) For q reject states in M, create equivalent reject states in P

   (e) For each transition from a state q to r that reads symbol a and writes symbol b and moves right, replace with:



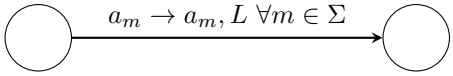
   (f) For each transition from a state q to r that reads symbol a and writes symbol b and moves left, replace with:



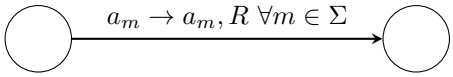
□

3. (a) To compile a $\mathscr{P}''$ program P into an equivalent TM, the following rules must be combined. These are equivalent TM transitions for each operation in $\mathscr{P}''$.

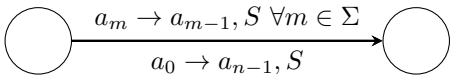
i. For a $<$, add the following transition:



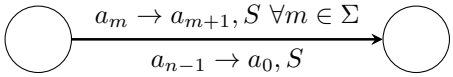
ii. For a $>$, add the following transition:



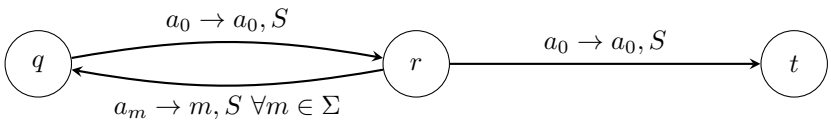
iii. For a $-$, add the following transition:



iv. For a $+$, add the following transition:



v. For a $[c]$ containing a series of commands c correlating to another TM M with start and end states q, r respectively, add the following transitions where t is the start state of the next operation following the while loop:



To compile P, operations should be concatenated, that is, a TM with q, r start and end states should then become a TM with q, t start and end states with the appropriate transition from r to t. Begin compiling by using a depth first search for while loops. Find the rightmost [in the program and match it with the leftmost]. Compile the operations inside into a TM, then apply the while loop transitions to it. Do this until the entire program has been compiled into one large TM, making sure to incorporate the other transitions as they come up.