Home Work 3

Rusu Wu
11694764

1. Assume that L= $\{ < M >: L(M)$ contains words abba $\}$ is recursive and recognized by an algorithm M_L. Then we construct an algorithm M' to recognize L_u.
M' : input $< M, w >$
Build a TM M'' ; input x to M''
  Check whether $x == abba$;
    YES, simulate M on w;
      if M accept w, then M'' accept x
    NO, M'' says no to x;
Run M_L on $< M'' >$
  M_L reject $< M'' >$, say NO ;
    Then M' rejects $< M, w >$
  M_L accepts $< M'' >$,
    Then $< M' >$ accepts $< M, w >$
Clearly, L_u is recognized by M', which is a controdiction.
‘

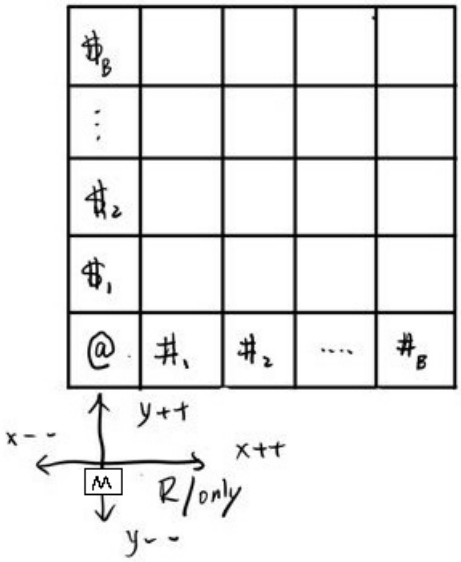
2. In order to prove that the emptiness problem for jump-LBAs is undecidable, we can prove that jump-LBAs is equivalent to LBAs.
We can construct a configuration for the jump-LBAs as follow;
When the cell change, its head will jump back to the beginning of the tape and come back to the cell.
Then the head moves to the right. If the next change happens; we mark the symbol, then we have two marks now.
The head jump back to the beginning of the tape, and when it reaches the first mark, erases the mark and back to the beginning of the tape. Repeat the configuration when the next change happens.
Such that we can simulate the jump-LBAs with normal write.
Assume that the emptiness for jump-LBAs is decidable by an algorism M_EJLBA, then we can show L_u can be recognized by algorism M', which is a controdiction.
M’ is constructed as follow;
Input TM M and input word w; Build a jump-LBA M'':
  input x
  check x is the following farmat
    $a_0\#a_1\#a_2\#....a_n\#$
    for some n, and each a_i is a configuration of TM M.
  Check a_0 is the initial configuration of M over input w.
  Check a_n is an accepting configuration.
  Check each a_i can reach $a_i + 1$ by a one step transistion in M.($a_i + 1$ is the resulting configuration from a_i after running one instruction in TM)
Once all Check out. M'' says yes on x. Run M_EJLBA on $< M'' >$
  M_EJLBA rejects $< M'' >$; then M' accepts $< M, w >$
  M_EJLBA accepts $< M'' >$; then M' rejects $< M, w >$
Clearly, M' recognize Lu, which is a controdiction.

3. L is recursive language, so we have a TM M to recognized it.We construct a TM M' to accept L' as follows.
M' : Input x;
    Then enumerate y string from short to long as y= 0;1;00;01;10....;
Run M on yxy; if M says YES on yxy,such that M' says YES on x;

Clearly, L' is regular because it can be recognized by M'.

4.Since L is recursive,so there is an algorithm M recognizing L. We construct a TM M' to accept L^r as follows.
M' : input x
Reverse $x = x^r$
Run M on x;
If M says YES on x, M' says YES on x;
If M says NO on x, M' says NO on x;
Clearly, L^r is recognized by M', such that L^r is recursive.

5. There is a two counter machine $M'(x,y)$, and we can use the 2-Turing machine M to simulate $M'(x,y)$.The simulation is as follow.
When $M'(x,y)$ runs, M uses its head position to simulate the read of x and y.
When M' excutes x++, M moves its head position move one cell from left to right;
When M' excutes x–, M moves its head position move one cell from right to left;
When M' excutes y++, M moves its head position move one cell from down to up;
Since we take a picture of the input M initially, we can assume that the symbol of its left boundary is $\$_B$; the symble of its down boundary is $\#_B$; the initial cell symble is @
When M' excutes x==0?, M checks whether the symbol of the cell is $\$_B$ or @;
When M' excutes y==0, M checks whether the symbol of the cell is $\#_B$ or @;



When $M'(x,y)$ halts, M says YES on the given input M;

However,the head position can not move beyond the boundary of M, otherwise M will crash. So, there is a number B, such that;
$M'(x,y)$ halts only if x and y can never exceed the value B in the process before halting;
$M'(x,y)$ halts only if there is an input w such that M halts on the input;
$M'(x,y)$ halts only if $L(M)$ is not empty.
Therefore, we know that two-counter machine $M'(x,y)$'s halting problem is undecidable,such that the emptiness problem of read-only 2-Turing machine M is undecidable.

6. A $P_i o$ path means that there are infinite prefixes of the path in which the vactors satisfies the given property. In order to prove that a path is $P_i o$, we need to prove its reachability. In other words, we need to find a state that the path would reach it infinitely. We can bulid a PDA to simulate the run. To explain, we use the particular symbols's appearance on the path as the input, and when we read a symbol that we never read before, we push it into the stack. When we read a symbol that we read before, we push a symbol out. Then check whether the highest

and lowest position are always the same; in this way, we can check whether there is a infinite loop. A given graph has an $P_i o$ infinite path when it has a loop that runs infinitely.