

18CSCI03H

Theory of Computing

Assignment 2

By Ahmed Hossam ElSabbagh

147176

Turing Machine One

The given of the equation 7x+1 for every x>=0 can be described through the following Turing Machine description and diagram.

Q = (q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11)

∑ = (1, #)

Γ = (0, 1, #, b, x)

q0 is the start state.

q11 is qaccept

Any state other than q11 is a qreject

Transitions are in diagram



This machine consists of 12 states, including the entry state q0 and the accepting state q11 colored in blue.

The input of this machine consists of an infinite tape starting with the following sequence #X#. Where X is any number of x ones, for example if x=3, X=111, therefore the input is #111#.

The machine works following this logic:

1) The index of the tape points to 0, or #, it than ignores it and goes to starting state q0 and moves the index to 1. #111#

2) For the first 1 the index is pointing at, the machine reads it at state q0, writing x and moves right, than it goes state q1. #x11#

3) The machine in q1 reads all the ones and write them back, logically ignoring them. #x11#, #x11#.

4) When finally the index reaches the #, the machine moves to q2, reading and writing # to logically ignore it. (b refers to blank) (The code logically adds blanks for every time the index reaches the end of the tape) #x11#b.

5) q2 then reads and writes all the following ones to logically ignore it and move the index right. #x11#b.

6) When finally the index reaches a blank it reads it and writes 1, then it moves right. #x11#1b, going to q3.

7) From q3 to q7, the states has only one transition, read blanks, writes 1s, and move right, go to the following states q4 q5 q6 q7. #x11#11111b.

8) When reaching state q7, the machine reads the blank at the index and moves left, and goes to q8. #x11#111111b.

9) In q8, the index moves reads and write 1s to ignore it and moves left. #x11#111111b, #x11#111111b, #x11#111111b, #x11#111111b, #x11#111111b.

10) When reading, # in q8, it continues left, ignoring # and going to q9, in q9, it goes back and ignores 1s until it reaches x. #x11#111111b, #x11#111111b, and #x11#111111b. When x is reached, the machine moves to q0 and the index moves right. #x11#111111b.

11) The machine than repeats the cycle initiated in step 2. Also following step 5, this is an example of what happens. #xx1#111111b, #xx1#111111b, #xx1#111111b, #xx1#111111b, #xx1#111111b, #xx1#111111b, #xx1#111111b.

12) This cycle is repeated until all ones behind the second # is turned into x. It should be noted that the total number of 1s should be 6\*x. In this case there are 18 ones. #xxx#111111111111111111b.

13) Going back to step 12 the machine should now be in q0, with the index pointing to the second #, therefore it should go left, turning the # into 1 and going to q10 #xxx1111111111111111111b.

14) When in q10, the machine loops on the previous x and turn them into 1s, #1111111111111111111111b. It than goes to q11 and turns the last # into 0. 01111111111111111111111b. It can be noted that the total number of 1s go as follow, 3\*6 from the first steps, which is 18, +1 that turned from the second # in step 13, a total of 19, and +3 from the x that turned into 1s in step 14, for total of 22, which is 7\*3 +1 or 21+1 or 22.

There are two variations of this machine that were created, the code of which can be found in the zip file.

Q = (q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12)

∑ = (1, #)

Γ = (0, 1, #, b, x)

q0 is the start state.

q12 is qaccept.

Any state other than q12 is a qreject.

This version puts 7 1s then turns x into 0s like this 00001111111111111111111111b (13 states).



Q = (q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10)

∑ = (1, #)

Γ = (0, 1, #, b, x)

q0 is the start state.

q10 is qaccept.

Any state other than q10 is a qreject.

While this one turns x into 1 when it comes back, for example at the beginning of the second loop, this should be the output #1x1#111111b, which is less computing time but leaves the # at the start. The final output should look like this #1111111111111111111111b (11 states).



Turing Machine Two

(This was already made before the mail)

The given of the equation x^2 for every x>=0 can be described through the following Turing Machine description and diagram.

Q = (q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11)

∑ = (1, #)

Γ = (0, 1, #, b, x, y)

q0 is the start state.

q11 is qaccept

Any state other than q11 is a qreject



This machine consists of 12 states, including the entry state q0 and the accepting state q11 colored in blue.

The input is similar to the first problem #111# for 3.

The gist of this operation is to first turn the problem from simple x to x+x using the states q0, q1, q2 and q3. Than this operation is turned into x\*x, which is equivalent to x^2 using the states q4 to q11.

The machine works following this logic:

(b refers to blank) (The code logically adds blanks for every time the index reaches the end of the tape)

1) Follow steps 1 to 5 in the previous problem explanation.

2) Same as the previous problem, the difference however is that it adds 0s after the second # and later ignores them as in step 5 instead of the ones. So the output becomes #x11#0b.

3) And then it comes back, ignoring all 1s, 0s and # until the first x. #x11#0b.

4) Repeat until the tape is like this #xxx#000b.

5) Read x, write 0 than move right to q5. #xx0#000b.

6) Read #, write # and move right. #xx0#000b, go to q6.

7) Read 0, write y and move right. #xx0#y00b, go to q7.

8) All the 0s are to be ignored until it gets to blank. #xx0#y00b.

9) Read the blank and turn it to 1, move left and go to q9, #xx0#y001.

10) Read and write 0s as well as 1s, ignoring them logically, moving left until the first y is encountered, then read y and write y to ignore it, move right and go to q6 #xx0#yy01.

11) Repeat step 7 and 8.

12) In q7, if 1 is encountered like in this tape #xx0#yy01, move right ignore 1 and go to q8 #xxx#yy01b.

13) Read the blank and make it 1, move right and go to q9 #xx0#yy011. Repeat step 10.

14) When the tape is like this #xx0#yyy111 and the machine is in q6. Read and write 1, move left, go to q10.

15) In q10 loop and turn all y into 0, and ignore #. #xx0#yyy111, ignore 0 and move left. #xx0#yyy111.

16) Ignore all the following 0s until the first x is encountered. Then repeat step 5 to 15.

The ending output should look like this 0000#000111111111b

Push Down Automata Problem.

The given of the language 𝐿 = {𝑎𝑛𝑐𝑏𝑛 | 𝑛 ≥ 1} ∪ {𝑎𝑛𝑑𝑏2𝑛 | 𝑛 ≥ 1} for every x>=0 can be described through the following PDA description and diagram.

Q = (q0, q1, q2, q3, q4)

∑ = (a, b, c, d)

Γ = (a, b, c, d, $, E)

q0 is the start state.

q2 is qaccept

Any state other than q2 is a qreject

Z0 = $ ($ in times new roman)

Transitions are in diagram.



The PDA consists of the 5 states including the start state q0 and the accept state q2 colored in blue. The epsilon is symbolized with the capital Greek letter E.

The input to the automata is a string described in 𝐿 = {𝑎𝑛𝑐𝑏𝑛 | 𝑛 ≥ 1} ∪ {𝑎𝑛𝑑𝑏2𝑛 | 𝑛 ≥ 1}, an example for it may be aacbb or aadbbbb, where any number of as is acceptable, while bs have to follow the description, where the bs are either equal to the as if c is in the middle, or are double the as if d is in the middle.

The automata works on the following steps.

1) Starting symbol $ is added to the stack, going to q0. aacbb, $.

2) q0 loops on itself, adding as to stack while the input still has as. aacbb, $a. aacbb, $aa.

3) When reading c as input, the input reader forces the automata to go q1, pushing and popping a, insuring that there was an a before it, otherwise reject. aacbb, $aa.

4) q1 then loops on itself, for every b in the input tape, popping as from the stack as it goes, until only $ remains. aacbb, $a. aacbb, $.

5) Once the end of input is reached, if the number of as is equal to the number of bs, the automata goes to q2 accept state, $ is removed, accepting the input. Elsewise, the automata will stop the input and reject the string, having the top of the stack different from the input tape.

6) Go back to step 2. aadbbbb, $. aadbbbb, $a. aadbbbb, $aa.

7) When reading d as input, the input reader forces the automata to go to q3 pushing and popping a, insuring that there was an a before it, otherwise reject. aadbbbb, $aa.

8) q3 reads b from the tape, it pops a from the stack and go to q4. aadbbbb, $a.

9) q4 reads b from the tape, it pops nothing from the stack (epsilon) and adds nothing to the stack it goes to q3. aadbbbb, $a.

10) Repeat steps 8 and 9 until no as are left in the stack, and only $ remains. aadbbbb, $. aadbbbb, $.

11) Once the end of input is reached, if the number of as is equal to the double of the number of bs, the automata goes to q2 accept state from q3, accepting the input. Elsewise, the automata will stop the input and reject the string, having the top of the stack different from the input tape.

**Test Examples:**

Here are the results for several runs of the program.

|  |  |  |
| --- | --- | --- |
| Input | Output | Target |
|  | reject | Insure it rejects empty input |
| a | reject | Insure it rejects a if b not written |
| b | reject | Insure it rejects b if neither a nor c nor d are written |
| acb, aacbb, aaacbbb, aaaacbbbb | accept | Insure that it accepts all strings that match the first requirement. |
| adbb, aadbbbb, aaadbbbbbb, aaaadbbbbbbbb | accept | Insure that it accepts all strings that match the second requirement. |
| ac, cbb, aa, aba, acbb, adb, ad, bd, db, cb, cd, aadbb | reject | Insure that it rejects all strings that match neither requirements. |
| c, d | reject | Insure that it rejects c and d if they do not have as or bs before or after it, which does not match, the requirement of n>=1 |