**COMP 314: Automata Theory**

**Homework 6**

Student Outcome: 1

Topics: Turing Machines

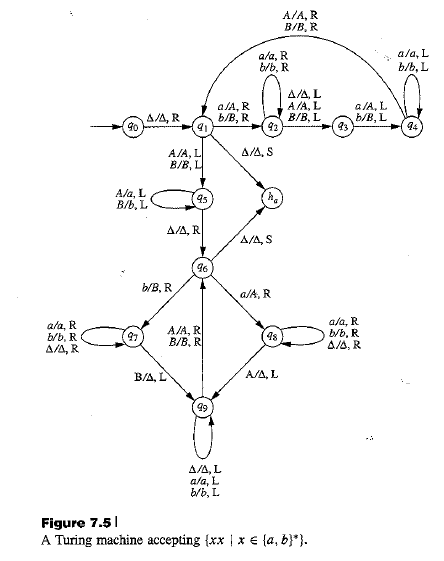
Due: April 17, 2023 11:55 PM

77 pts

Upload a Word document or OneNote with your solution. Note: Not all of the problems below may be graded; I may choose only a subset to grade.

6c

1) Trace the following TM, showing its configuration at each step as it processes the string *aaba*. For a “configuration,” you need to indicate state, contents of tape, and the current tape cell; the easiest way to do so is to use the syntax we talked about in class (e.g., ∆ab3bba). (8 pts)



1. 0∆*aaba*∆
2. ∆1*aaba*∆
3. ∆*A2aba*∆
4. ∆*Aa2ba*∆
5. ∆*Aab2a*∆
6. ∆*Aaba2*∆
7. ∆*Aab3a*∆
8. ∆Aa4bA∆
9. ∆A4abA∆

10. ∆4AabA∆

11. ∆A1abA∆

12. ∆AA2bA∆

13. ∆AAb2A∆

14. ∆AA3bA∆

15. ∆A4ABA∆

16. ∆AA1BA∆

17. ∆A5ABA∆

18. ∆5AaBA∆

19. 5∆aaBA∆

20. ∆6aaBA∆

21. ∆A8aBA∆

22. ∆Aa8BA∆

22. ∆AaHrBA∆

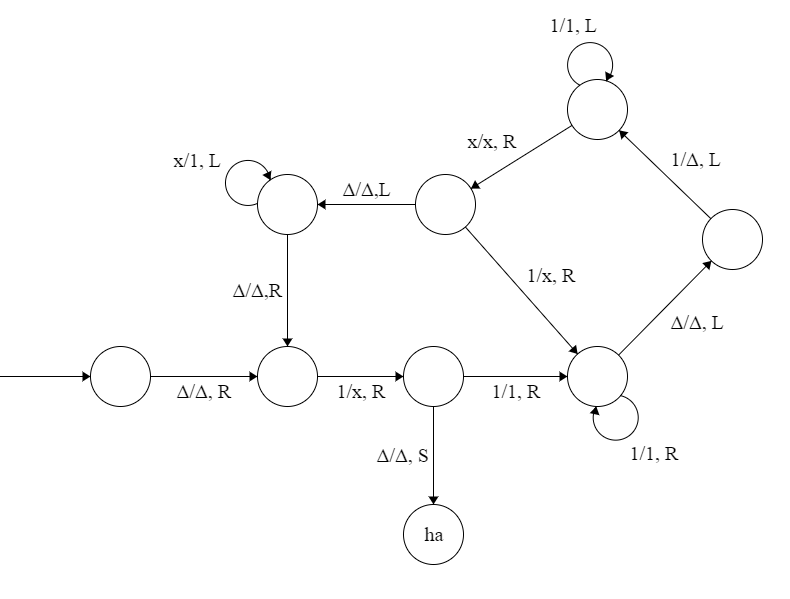
2) Construct a TM that accepts the language of all strings with an equal number of a’s and b’s, using the following method. Search the string left-to-right for an *a*. If you find one, replace it with an *X* and return to the far left of the string; then, search through again, looking for a *b*. If you find one, replace it with an *X*, return to the far left of the string, and start over from the beginning. Continue in this way until one of the two searches is unsuccessful; you must decide how the machine should handle these cases. (Hint: There are two ways that an unsuccessful search could indicate failure, and one way it could indicate success. Make sure you can tell the three apart!) (10 pts)

Diagram, schematic

Description automatically generated



3) Describe the language accepted by the following TM. (Hint: you may want to try a longer string, like 11111 or 111111, to make sure you’re seeing the pattern.) (6 pts)



**It accepts the powers of 2 in unary.**

4) We’ve never bothered to create null transitions for a TM; in fact, it doesn’t make sense to do so. Why not? What features of a TM make them unnecessary or inappropriate? (5 pts)

**When designing and FA we would sometimes use a null transition because we were in a state where we want to move to another state without eating a character. It was important to not eat a character until we were ready to do so because in an FA you are only allowed to go through the string left to right and you can only do so once. But a TM can go left, right, or even stay in the same place which means we can iterate through the string any number of times we want in either direction, so null transitions aren’t in any way useful to a TM.**

5) Create a TM called *Insert*. At the beginning of its operation, the tape contents are ∆*y*0*z*, where 0 denotes the initial state, and each of *y* and *z* is a string in {a, b}\*. Notice explicitly here: unusually for a TM, we are beginning at the start of string z, not at the beginning of the tape. Your objective is to provide a machine that will end in state ∆ha*yxz*, where *x* is literally the character “x”; that is, you should insert a single *x* character between the two existing strings. (8 pts)

Diagram

Description automatically generated



6) Create TMs to calculate the following results. Typically, a calculation TM requires you to copy your answer back to the beginning of the tape, but that is not necessary here; you may assume that the string will be copied back to the beginning after you finish, as long as you end up pointing at a blank immediately before the answer. (30 pts)

1. f(x) = x+2

Diagram

Description automatically generated

1. f(x) = 2x

Diagram, schematic

Description automatically generated

**Note:** The +/+ transitions from states 0 and 6 are useless for the default implementation of the doubler but are important for later TMs.

1. f(x) = x2

**Note:** this TM has 2 other TMs built into it. ­­“dub” represents the doubling machine outlined in problem 6b and “sqr” represents this machine (i.e the machine is recursive). This machine solves for x2 using the formula

Here is a general layout in case its difficult to read.

1. State 1 checks if we need to recurse or if we have hit a base case (n = 0).
2. States 10-14 are for after we hit our base case. When we hit the base case, in which case the algorithm will get the sum of all numbers on the tape and return it.
3. States dub-4 create 2n-1.
4. States 4-9 append +n-1 to the end of the tape and then recurses.

Diagram, schematic

Description automatically generated

7) Draw a TM that takes as its input a binary representation of a nonnegative integer and produces as its output a unary representation of that integer. (So, for instance, it might turn 101 into 11111). Hint: You can solve this problem by doing an awful mess involving lots of doubling, but there’s an easier way. Think about different approaches you could take to turn a binary number into a unary one; which one seems easiest? Come see me if you’re stuck. (10 pts)

**Note:** state “dub” (short for double) represents the entirety of the machine outlined in problem 6b

Diagram, schematic

Description automatically generated