Retake of Quiz 4: NFAs and TMs

Collaboration: This quiz is open note but individual. You may use any resources that were provided to you by the course, or that you had recorded as part of your own notes prior to when you first viewed this quiz. You may not discuss this quiz with your cohort-mates, class-mates, tutor, friends, family, or anyone else except the course staff (who will answer clarification questions only).

Problem 1: Non-Deterministic Finite State Automata

Give a non-deterministic finite state automaton which decides each of the languages described below, using no more than the number of states indicated. Give both a drawing and a description.

(a) $\{\varepsilon\}$, using no more than one state.

(b) $\{x \in \{0,1\}^* \mid x$ contains the substring 001010 or 110 (or both)$\}$, using no more than ten states.

Problem 2: Turing Machine Running Time

Below I have provided a Turing machine which computes the following function:

$$DoubleLength(w,x) = \begin{cases} 1 & \text{if the length of } w \text{ is twice the length of } x. \\ 0 & \text{otherwise} \end{cases}$$

(Note that you were asked to describe a Turing machine to compute $DoubleLength$ in the original Quiz 4. Please answer this question based on the Turing machine provided below and not the Turing machine that you described then.)

This function returns one for all strings that have the format: a binary string, then the # symbol, then a string that is exactly half as long as the first one was. The following strings therefore belong to the language:

– $1011\#11$

– $0000\#11$

– $11\#1$

– $\#$

And then these strings do not belong to the language:

– $1011\#110$

– $01\#01$

– $1\#$

– ε

– $0\#\#0$

– 011

The behavior of the Turing Machine is as follows (note that this description is of slightly higher specificity than our expectations for Turing Machine "psuedo-code"):

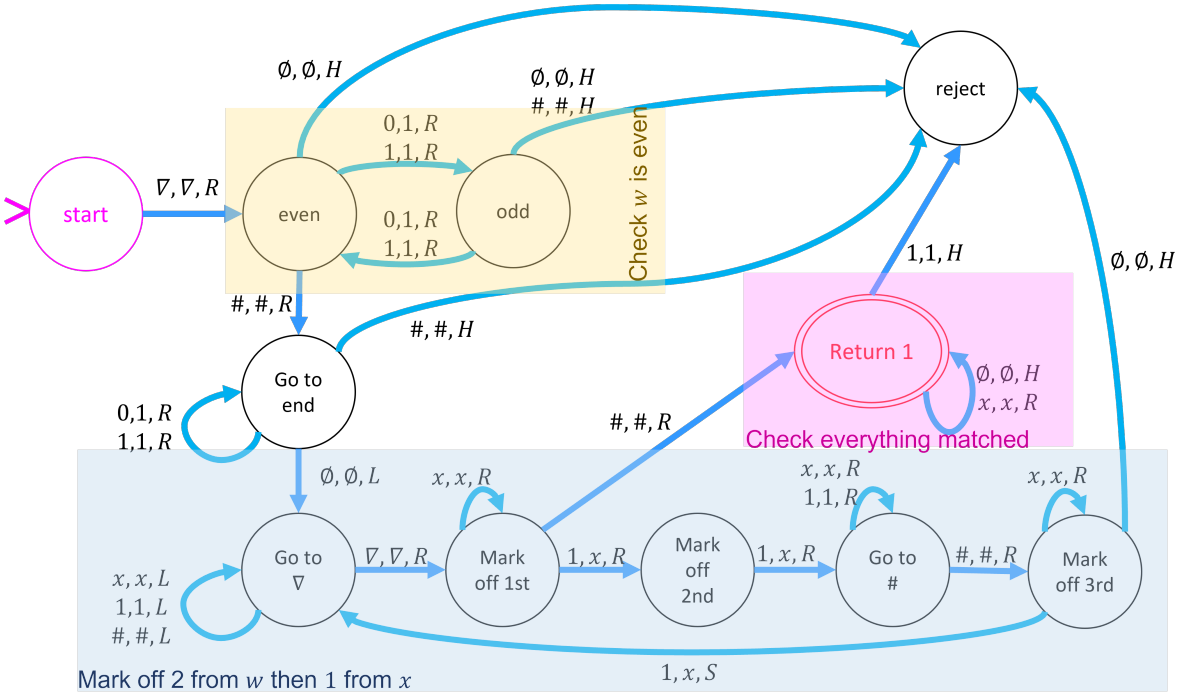
1. Check that the portion of the input before the # has even length by (see yellow states of the machine):

   (a) After the start of tape character, move right
   (b) for each 0 or 1 that you see, alternate between a state labelled "even" and one labelled "odd" (also it overwrites every 0 with a 1 because it makes things a bit simpler later if we can assume there aren't any 0s, this does not change the length of the string)
   (c) if you see the $\#$ character after seeing an odd number of bits then halt and return 0.
   (d) Go to the end of the input making sure there are no more $\#$ characters (and overwriting 0s with 1s as you go).

2. Repeatedly mark off two bits from the w portion of the string and one bit from the x portion by (see blue states):
   (a) Go to the beginning of the tape by moving left until the read head is at $\triangle$
   (b) Mark off (by overwriting with x) two bits from before the $\#$ symbol.
   (c) Go to the $\#$ symbol by moving right on all unmarked bits.
   (d) Mark off the first unmarked bit that's after the $\#$ symbol. (Return 0 if there is no such bit)
   (e) repeat until there are no more bits to be marked off before $\#$.

3. Check whether all bits after the $\#$ character have been crossed off by (see pink states):
   (a) After the $\#$ character, move right on every marked-off character seen.
   (b) if the first non-marked character is a bit, then return 0. If the first unmarked character is $\emptyset$, return 1.

The following image depicts one potential implementation of the machine described above:



Consider that this machine is given each of the inputs describe below, all of which are $3n+1$ characters long. Give a Θ bound on the total number of transitions this machine would take before halting as a function of n. Assume $n \geq 1$. You should informally justify how you came to that conclusion (e.g. "after each 0 we overwrite we need to take one more transition"), but a formal proof is not necessary.

a. A string of 3 1s, followed by #, followed by $3n-2$ 1s. (e.g. for $n=2$ this would be $111\#1111$)

b. The # character followed by a string of $3n$ 1s (e.g. for $n=2$ this would be $\#111111$)

c. A string of $2n$ 1s followed by the # character, then followed by n 1s. (e.g. for $n=2$ this would be $1111\#11$)

d. A string of $3n+1$ #s (e.g. for $n=2$ this would be $\#\#\#\#\#\#\#$)

Problem 3: Describe the TM

Each subproblem below gives a function that can be computed by a Turing machine. Describe, using high level Turing machine "pseudo-code", how a Turing machine could compute the given function. For example, your pseudo-code might have statements like "scan left until you find a 1, then cross it off", or "halt and return 1 if there you get to the start of the tape without seeing a 1". Refer to Problem 1 to see an example.

For all problems, the input alphabet of the Turing machine will be $\{0,1,\#\}$ where the input string given will be $w\#x$ where w and x are the inputs to the functions below (so, for example, to compute $SameLength(11,011)$ the Turing machine's input would be $11\#011$). You may use whatever tape alphabet you wish.

(a) for $w \in \{0,1\}^*, x \in \{0,1\}^*$,

$$BitwiseNeg(w,x) = \begin{cases} 1 & \text{The bitwise negation of } w \text{ is } x. \\ 0 & \text{otherwise} \end{cases}$$

Where the bitwise negation of a string s is the string produced by negating each individual bit of s. For example, the bitwise negation of 01011 is 10100.

(b) for $w \in \{0,1\}^*, x \in \{0,1\}^*$,

$$TwoShorter(w,x) = \begin{cases} 1 & \text{if the string } w \text{ is two characters shorter than the string } x. \\ 0 & \text{otherwise} \end{cases}$$

