Handout 8B

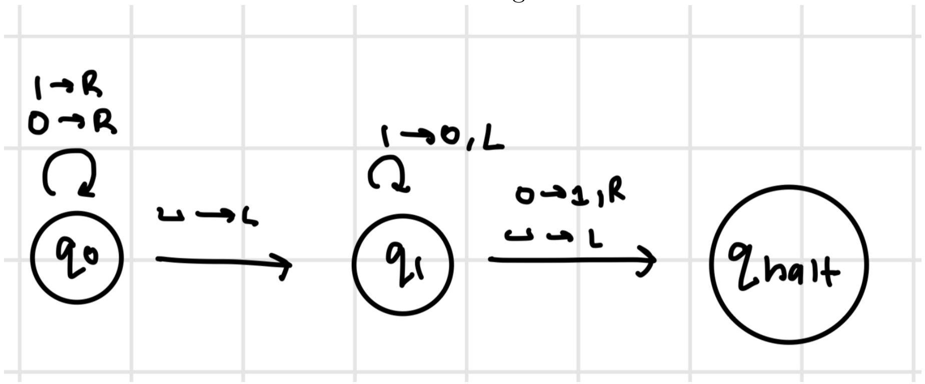
Anum Ahmad and Zachary Thayer,

Q1-3 adapted from Andrew Jin and Anastasija Tortevska

COMS 3261 Fall 2024

1. Consider the input-output TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{halt})$ where $Q = \{q_0, q_1, q_{halt}\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \sqcup\}$, and δ is given by:
$\delta(q_0, 0) = (q_0, 0, R)$, $\delta(q_0, 1) = (q_0, 1, R)$, $\delta(q_0, \sqcup) = (q_1, \sqcup, L)$
$\delta(q_1, 0) = (q_{halt}, 1, R)$, $\delta(q_1, 1) = (q_1, 0, L)$, $\delta(q_1, \sqcup) = (q_{halt}, \sqcup, L)$

Here is the TM written in the form of a diagram



(a) Provide the complete sequence of configurations of M when ran on input 100.
What is the output of M on this input?

Solution:
$q_0100 \Rightarrow 1q_000 \Rightarrow 10q_00 \Rightarrow 100q_0 \Rightarrow 10q_10 \Rightarrow 101q_{halt}$
The output of M on 100 is 101.

(b) What is the output of M on 10011? on input 11?

Solution:
The output of M on 10011 is 10100.
The output of M on 11 is 10.

(c) What function is computed by M?

Solution:
The function computed by M is the following:
On input 1^n: output 10^{n-1}
On input ε: output ε
On input $x \neq 1^n, x \neq \varepsilon$: treat x as a binary number and output $x+1$.

2. Let $\Sigma = \{\#, 0, 1\}$. Provide an *implementation level* description of a input-output TM that computes the function

$$f(\#\langle x\rangle) = \begin{cases} \#\langle x/2\rangle & \text{if } x \text{ is even} \\ \#\langle 3x+1\rangle & \text{otherwise} \end{cases}$$

where $\langle x\rangle$ stands for the binary representation of the number x.

(For example, if the TM starts with #100 on the tape it should halt with #10 on the tape; if it starts with #11, it should halt with #1010.)

You may use a TM with more than one tape – in this case the output should be written on the first tape.

Solution:

We will use a 3-tape TM. The intuition is to first check whether the input x is even (the last digit is 0) or odd (the last digit is 1). If the former, we replace the terminating 0 with a blank (dividing by 2 in binary) and halt. If the latter, we duplicate the input on a second tape and add a terminating 0 to the original (multiplying by 2 in binary). We now have $2x$ on the first tape and x on the second tape. We then add a leading zero to the first tape and two leading zeroes to the second so that we can sum the newly aligned tapes to get $3x$, to which we finally add 1. The third tape is used to keep track of carry bits in the addition step.

More specifically, construct machine M as follows: On input $\#\langle x\rangle$, where $\langle x\rangle$ is a binary string:

(a) Make sure the first character is a #, then scan the remaining binary number on the input tape from left to right until the last digit is reached. If it is a 0 (i.e., the number is even), replace the 0 with a blank and halt. (If it is a 1, go on.)

(b) Write "#00" to the second tape, then copy the contents of the first tape onto the second tape after the '#00".

(c) Shift the contents of the first tape after the # over to the right by 1. Make the new first symbol after the # on the first tape a "0".

(d) Replace the first blank on the first tape with a 0.

(e) Write a # to the third tape for each symbol on the first tape, then replace the rightmost # with a 0.

(f) Add the contents of the second tape to the first tape by repeating the following two steps until all the digits of the second tape are crossed off:

   i. Starting with the rightmost digit on the second tape that has not been crossed off, add it and the digits from the other two tapes in the corresponding place (i.e. the same number of spaces to the left of the first blank on their tape), write the result to that place on the first tape, and write any carry to the third tape:

A. If all of the three digits are 0, write 0 to that place on the first tape and replace the rightmost # on the third tape with a 0.

B. If only one of them is 1, write 1 to that place on the first tape and replace the rightmost # on the third tape with a 0.

C. If only two of them are 1, write 0 to that place on the first tape and replace the rightmost # on the third tape with a 1.

D. If all three of them are 1, write 1 to that place on the first tape and replace the rightmost # on the third tape with a 1.

ii. Cross off the rightmost digit on the second tape that has not been crossed off.

(g) Add 1 to the number on the first tape by starting with the rightmost digit and checking if it is 0. If it is 0, change it to 1 and go to the final step. If it is 1, change it to 0, proceed to the next rightmost digit and repeat this step.

(h) If the first symbol after the # on the first tape is a 0, delete it and shift the remainder of the tape one to the left. Halt.

3. Let $L = \{\langle M, k\rangle | M$ is a TM, k is a positive integer, and there exists an input to M that makes M run for at least k steps$\}$
   Prove that that L is decidable.

   First attempt
   We try to construct a TM that decides L. It will do the following:

   (a) On input $\langle M, k\rangle$ where M is a TM:
   (b) For all strings $w_i, i \in \mathbb{N}$
       - run M on w_i for k steps. If M doesn't terminate on w_i within k steps, accept.
   (c) If we're finished enumerating and M terminated within k steps every time, reject

   Problem: When do we reject? We need to enumerate through an infinite number of strings! Therefore, we will never be finished enumerating , so we can never decide when to reject!. This only recognizes the language, not decides it
   Second attempt
   Each "step" of a TM's computation means it executes a single transition function. Each time a transition function is executed, the head of the TM can only move at most a single space in either direction. Even if the TM head moves right on every single transition, then in k steps, the head could only ever look at the first $k+1$ characters on the tape. Therefore, to determine whether the machine ever runs for more than k steps or whether it halts within k steps on every input, anything on the tape after the first $k+1$ symbols is not relevant. Thus, we don't need to enumerate over any strings that are longer than $k+1$ characters long.

   We can construct a TM that decides L. It will do the following:

   (a) On input $\langle M, k\rangle$ where M is a TM:
   (b) For all strings w_i where $|w_i| \leq k+1$:
       - run M on w_i for k steps. If M doesn't terminate on w_i within k steps, accept.
   (c) If we're finished enumerating and M terminated within k steps every time, reject

   Proof that this is a decider:
   $x \in L \Rightarrow M$ runs for at least k steps on some string, therefore there is a string w_i such that $|w_i| \leq k+1$ where the computation of M on w_i will not have terminated after k steps. Therefore, the program will accept on some iteration.
   $x \notin L \Rightarrow M$ terminates within k steps on every input string. Since we

are testing a finite number of strings and M terminates on all of them, we will reach step (c) and then reject.

4. Let L be a recognizable language.
Define 9/10(L) = $\{< x_1\#x_2\#...\#x_{10} > \ \mid$ at least 9 of $x_1, \ldots, x_{10}$ are in $L\}$.
Prove recognizable languages are closed under the 9/10 operation. Give both a high level and implementation level solution.

Implementational level: We solve this similarly to the example problem AtLeastOne(L).

Let M be the TM recognizing L, here's our multitape TM M', with 11 tapes, for 9/10(L):
on input $w = x_1\#x_2\#...\#x_{10}$, do the following:
1. Write each x_i on 10 separate tape.
2. Simulate 1 step of M on each of the 10 tapes. If M accepts on tape i, record this on tape 11.
3. Check if 9 tapes have accepted, if so then accept w, otherwise return to step 2

Here's why this works:
If w is in AtLeastOne(L), this means 9 out of the 10 x_i's are in L. Since M recognizes L, it will eventually accept these 9 strings, and thus M' will accept w.
If w is not in AtLeastOne(L), then less than 9 of the strings are in L. So, M won't ever accept on 9 tapes, and we run indefinitely, thus M' doesn't accept w.

High level: on input $w = x_1\#x_2\#...\#x_{10}$, run M on all ten inputs in parallel and accept if at least 9 of them accept. We see by definition if $w \in$ 9/10(L) then we accept, and if $w \notin$ 9/10(L) then we don't accept.

5. Let L be a recognizable language.
Define m/n(L) = $\{< (m, n, x_1\#x_2\#...\#x_n) > \mid$ at least m of $x_1, \dots, x_n$ are in $L\}$.
Prove recognizable languages are closed under the m/n operation. Give a high level implementation.

Note: we cannot simply simulate all n strings on m tapes like the previous problem, since we have to declare the number of tapes our Turing machine will use ahead of time, and we don't know what m and n will be right now, when designing the Turing Machine.

Consider the Turing Machine M' which operates as follows on input $w = (m, n, x_1\#x_2\#...\#x_n)$:
1. Set a counter variable to 1
2. Simulate M (the Turing Machine which recognizes L) on each x_i for "counter" many steps. That is, if the counter is 1, just do 1 step on each string.
3. If m of the x_i's accepted, accept.
4. Increment the counter and return to step 2.

Here's why this works:
If $w \in m/n(L)$ then eventually M will accept at least m of the strings when some threshold for the counter is reached, so M' accepts.
If $w \notin m/n(L)$ then M will never accept m of the strings, and M' will never accept.

6. Let CYCLE $= \{\langle G\rangle \mid G$ is a graph that contains at least one cycle$\}$ Give a nondeterministic Turing Machine which decides the language CYCLE.

   Consider the following NTM N': on input $w = \langle G\rangle$, where G is a graph, we nondeterministically choose a vertex v in G and then run the NTM N for REACHABILITY from section **2.2** on $\langle G, v, v\rangle$, and output what N outputs.

   Observe that if $w \in$ CYCLE, then some vertex is, by definition, reachable from itself, so there exists a choice of vertex that makes N accept, and thus so does N'.
   If $w \notin$ CYCLE, then there is no vertex which is reachable from itself, so every choice makes N reject, and thus every choice makes N' reject.