# CIS 511 Spring 2008: Homework 4 Solutions

## Problem 1

Let $\Sigma = \{0, 1\}$. Consider the language

$$L = \{0^n 1 0^{2n} 1 0^{3n} \mid n \geq 0\}$$

Describe a standard (deterministic, single-tape) Turing Machine $M$ that accepts $L$.

We give the following standard Turing machine $M = (Q, \Sigma, \Gamma, q_0, F, \delta)$ where $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, \#, B\}$. The transitions $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$ are shown below.

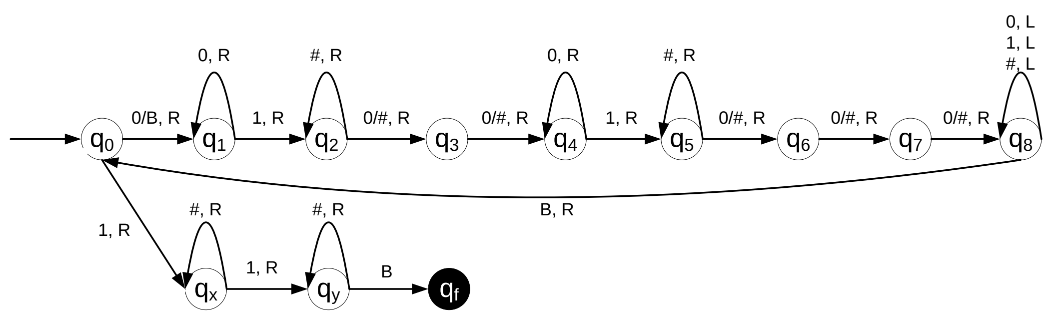


Figure 1: Standard Turing Machine M for L

Given an input $w$, $M$ checks if $w \in L$. It works as follows:

1. Overwrite one 0 (in the first sequence of 0's) by one $B$ and move right.
2. Move right skipping over 0's until the first 1 is encountered.
3. Move right skipping over #'s until 0 is encountered.
4. Overwrite two 0's (in the second sequence of 0's) by two #'s and move right.
5. Move right skipping over 0's until the second 1 is encoutered.
6. Move right skipping over #'s until 0 is enountered.
7. Overwrite three 0's (in the thrid sequence of 0's) by three #'s and move right.
8. Move left skipping over all symbols until $B$ is encountered.

We repeat the above steps until $M$ halts. Finally, we can determine if $w \in L$ by checking if all 0's on the tape are overwritten by $B$'s or #'s.

## Problem 2

Prove that both the following classes of languages are closed under concatenation: recursive languages and recursively enumerable languages.

Given two recursive languages $L_1$ and $L_2$, we will show that $L_1 \circ L_2$ is also a recursive language. Let $M_1$ and $M_2$ be the halting Turing machines for $L_1$ and $L_2$ respectively. We construct a halting Turing machine $M$ for $L_1 \circ L_2$ as follows. Given an input word $w$, consider all possible splits $w = w_1 w_2$. $M$ simulates $M_1$ on the input $w_1$ and simulates $M_2$ on the input $w_2$. If there exists a split such that both return yes ($w_1 \in L_1$ and $w_2 \in L_2$), then $M$ returns yes ($w \in L_1 \circ L_2$). Otherwise, $M$ returns no. Note that any input word $w$ has only a total of $|w| + 1$ (finite) possible spilts. We conclude that $M$ is a halting Turing machine for $L_1 \circ L_2$. Hence, $L_1 \circ L_2$ is recursive.

Given two recursively enumerable languages $L_1$ and $L_2$, we can show that $L_1 \circ L_2$ is also a recursively enuerable language by a similar construction. Let $M_1$ and $M_2$ be the Turing machines for $L_1$ and $L_2$ respectively. We construct a Turing machine $M$ for $L_1 \circ L_2$ as follows. Given an input word $w$, for each $i \geq 0$, we repeatedly check that if there exists a split $w = w_1 w_2$ such that $M_1$ accepts $w_1$ in $i$ steps and $M_2$ accepts $w_2$ in $i$ steps. If so, $M$ returns yes and halts immediately. Otherwise, $M$ continues to check other possible splits or increase $i$ by one. Note that $M$ could run forever if $w \notin L_1 \circ L_2$. However, $M$ always halts on any input $w \in L_1 \circ L_2$. Hence, $L_1 \circ L_2$ is recursively enumerable.

## Problem 3

A *two-dimensional* Turing machine has the usual finite-state control, but a tape that is a two-dimensional grid of cells, infinite in all directions. The input is placed on one row of the grid, with the head at the left end of the input and the control at the start state. Acceptance is by entering final state. Prove that the languages accepted by two-dimensional Turing machines are the same as those accepted by ordinary TMs. lts and A language accepted by a standard Turing machine $M_1$ can be accepted by a 2D TM $M_2$. $M_2$ can directly simulate all the steps of $M$ (while using only one row of the grid).

We will show that a language accepted by a 2D TM $M_2$ can be accepted by a multi-tape Turing machine $M_1$. Thus it will follow that it can be accepted by a standard single-tape TM. Let us number the cells of $M_2$ by pairs of integers $(x, y)$ where the initial position of $M_2$'s head is numbered $(0, 0)$. This splits the grid into four quadrants. $M_1$ stores the pair of integers on the first "position" tape. $M_1$ will have four one-way infinite "grid" tapes, each representing a quadrant of the 2D grid (e.g. the first tape represents the quadrant where $x \geq 0, y \geq 0$). To simulate an access to an element of the grid, $M_1$ uses the signs of $x, y$ to determine the tape on which the symbol is stored. The absolute values of $x, y$ are used to determine the position of the element on the grid tape, using the standard diagonal numbering function

$$f(a, b) = \frac{(a+b)(a+b+1)}{2} + b.$$

The value of $f(|x|, |y|)$ is calculated on an additional "scratch" tape. The simulation of a single transition of $M_2$ is then done as follows:

1. Use the signs of $x$ and $y$ to determine the grid tape which should be used.

2. Compute $a = f(|x|, |y|)$ using the scratch tape.

3. Move the appropriate grid tape head to the $a$-th cell. (Note that grid tapes are one-way infinite, and we can use a special marker symbol to mark the leftmost cell.)

4. Based on the current state of $M_2$ and the symbol read, determine the next state, overwrite the current cell on the grid tape, and simulate the move of the head of $M_2$ by increasing or decreasing $x$ or $y$ on the position tape.

## Problem 4

Suppose we modify the definition of standard deterministic single-tape Turing machine so that it can either move right or stay put, but has no option to move left. Thus, its transition function has the form

$$\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{R, S\}.$$

Here, $S$ stands for stay put (do not move the head). Show that this variant is *not* equivalent to the standard version. What class of languages do these machines accept?

Turing machines with stay put instead of left can only recognize regular languages. The intuition is that $M$ cannot move left and cannot read anything it has written on the tape as soon as it moves right, and therefore it has essentially only one-way access to its input, much like a DFA.

For every DFA $D$, there clearly exists a Turing machine with stay put instead of left that accepts the same language, since a DFA is simply a Turing machine with a read-only tape and a tape head which only moves to the right.

Consider a Turing machine $M = (Q, \Sigma, \Gamma, B, \delta, q_0, F)$ where $\delta$ is a partial function from $Q \times \Gamma$ to $Q \times \Gamma \times \{R, S\}$. We will construct a finite-state automaton $M'$ with $\epsilon$-transitions as follows. The set of states of $M'$ is $Q \cup Q \times \Gamma$: a state $q$ means that $M$ is in state $q$ and the machine has just moved to the current cell, and a state $(q, X)$ means that $M$ is in state $q$ and current cell holds the symbol $X$. Note that $M'$ has only finitely many states. In states of the former kind, $M'$ consumes an input symbol, and in states of latter kind, it use $\epsilon$-transitions to simulate $M$ (if $M$ stays put while repeatedly overwriting current cell). The initial state is $q_0$. The set of final states is $F \cup F \times \Gamma$: $M'$ accepts whenever $M$ is in a final state. Transitions are as follows.

- For $q \in Q$ and $a \in \Sigma$, if $\delta(q, a) = (q', X, R)$ (that is, $M$ moves right while processing input symbol $a$), then $M'$ contains a transition $(q, a, q')$, and if $\delta(q, a) = (q', X, S)$ (that is, $M$ stays put while processing input symbol $a$), then $M'$ contains a transition $(q, a, (q', X))$.

- For $q \in Q$ and $X \in \Gamma$, if $\delta(q, X) = (q', Y, R)$ (that is, $M$ moves right while processing tape symbol $X$), then $M'$ contains a transition $((q, X), \epsilon, q')$, and if $\delta(q, X) = (q', Y, S)$ (that is, $M$ stays put while processing tape symbol $X$), then $M'$ contains a transition $((q, X), \epsilon, (q', Y))$.

It is easy to verify that $M$ accepts an input word $w$ precisely when $M'$ accepts $w$.