CSCI 510, Fall 2016, Homework # 3

SOLUTION SKETCHES

Due date: Wednesday, October 26, Midnight

1. Give an implementation-level description of a Turing machine that recognizes the language A over the alphabet $\{0, 1\}$, where

$$A = \{w | w \text{ does not contain the same number of 0s and 1s}\}$$

We use a three tape machine.

On input w, scan the input from left to right and each time a zero is encountered, mark an "X" on the second tape, and each time a one is encountered, mark an "X" on the third tape.

After w has been scanned, rewind tapes two and three, scan them from left to right until the end of one sequence of Xs is found. If both tapes run out of Xs at the same square, reject, else accept.

2. Show that a language is decidable iff some enumerator enumerates the language in the standard string order.

We describe a machine that decides the language:

(a) On input w:

(b) Start enumerating the language in standard string order.

(c) If any enumerated string is identical to w, accept.

(d) If any enumerated string is longer than w, reject.

Since there are only a finite number of strings with length $\leq w$, this algorithm is guaranteed to halt. Further, if w is not enumerated when a longer string is output, it will never be enumerated.

3. A ***queue automaton*** is like a push-down automaton except that the stack is replaced by a queue. A ***queue*** is a tape allowing symbols to be written only on the left-hand end and read only at the right-hand end. Each write operations (we'll call it a *push*) adds a symbol to the left-hand end of the queue and each read operation (we'll call it a *pull*) reads and removes a symbol at the right-hand end. As with a PDA, the input is placed on a separate read-only input tape, and the head on the input tape can move only from left to right. The input tape contains a cell with a blank symbol following the input, so that the end of the input can be detected. A queue automaton accepts its input by entering a special accept state at any time. Show that a language can be recognized by a deterministic queue automaton iff the language is Turing-recognizable.

If a language can be recognized by a deterministic queue automaton, then clearly it is Turing-recognizable, as we can simulate a queue with a portion of the Turing tape, writing at one end and erasing at the other by traversing the tape.

On the other hand, suppose a language is Turing-recognizable. We can simulate the operations of a Turing machine using a queue automaton as follows. (There are lots of details.)

We will simulate the tape of a Turing machine with the contents of the queue.

First, copy the input string to the queue. Also, put two blanks at both ends of the queue. Also mark (with the current state of the TM) the position of the read-write head on the input (it will be the first character of the input at this stage), and also mark the character in front of the scanned cell with an arrow(indicating the scanned cell is coming next). As an illustration, the following input (where △ is the read-write head):

0	1	0	1	1	1
△					

Is pushed onto the queue as follows (where q_0 is the start state of the TM):

␣	0	1	0	1	1	1	␣
	q_0						

In this figure, characters are written on the right end of the queue and read from the left (the reverse of the instructions), to make the queue copy of the input line up better with the original input.

Now we have to simulate two types of Turing machine operations, moving right and moving left:

$$\delta(a, b) = (c, d, R)$$
$$\delta(a, b) = (c, d, L)$$

We do one of these in the course of pulling and pushing the entire string. We can tell where the ends of the string are by the blank characters. We can make sure we never run out of blanks by pushing an extra blank on the front and back of the queue every time we traverse it. The blanks may grow unnecessarily, but we can always be sure we will never run out if we add one before performing each operation.

Now, as we pull and push, we maintain a two-character buffer (represented by extra states in the finite automaton), so that we can adjust the position of the head as we change the tape contents. There are two cases:

(a) If the current rule to be applied looks like this: $\delta(q_a, a) = (q_b, b, R)$, suppose the tape and buffer start out like this (the first characters will be blanks):

Buffer		Queue						
x_0	x_1	x_2	$\dots$	x_i	a	x_j	$\dots$	x_n
					q_a			

Eventually the pulling and pushing will end up like this

Buffer		Queue						
a	x_j	x_{j+1}	$\dots$	x_n	x_0	x_1	$\dots$	x_i
q_a								

at which point we can change the a to a b, change the state from q_a to q_b, and move the head to the right by pushing the changed characters like this:

Buffer		Queue								
x_{j+1}	x_{j+2}	x_{j+3}	$\dots$	x_n	x_0	x_1	$\dots$	x_i	b	x_j
										q_b

(b) If the current rule to be applied looks like this: $\delta(q_a, a) = (q_b, b, L)$, the case is similar, except that when the tape and buffer reach this state:

Buffer		Queue						
x_i	a	x_j	$\dots$	x_n	x_0	x_1	$\dots$	x_{i-1}
	q_a							

we can push the changed characters like this:

Buffer		Queue								
x_j	x_{j+1}	x_{j+2}	$\dots$	x_n	x_0	x_1	$\dots$	x_{i-1}	x_i	b
									q_b	