CS331: Homework #5

Due on March 07, 2013 at 11:59pm

Professor Zhang 9:00am

Josh Davis

Problem 1

Let $\Sigma = \{0, 1\}$. Consider the following language over Σ, $L = \{w \in \Sigma^* : w$ contains twice as many 0's as 1's $\}$

Part One Describe in pseudo-code a Turing machine, M that recognizes L.

The pseudo-code that describes M is as follows:
$M =$" On input string w:

1. Start on the left side of the word, if the word is empty, move to q_{accept}
2. Move to the left of the word
3. Move right until we find a 1, cross it out. If we reach the end of the tape, move to stage 8
4. Move back to the left of the word
5. Scan right until the first 0 is found, cross it out. If we reach the end of the tape, move to q_{reject}
6. Move back to the left of the word
7. Scan right again until the second 0 is found, cross it out and move to stage 2. If we reach the end of the tape, move to q_{reject}
8. Move to the left of the word
9. Scan over the word, if we reach anything but an x, (0, 1), move to q_{reject}, if we reach the end of the word, move to q_{accept}"

Part Two Formally define M, where $M = \langle \Sigma, \Gamma, Q, q_0, q_{accept}, q_{reject} \rangle$

M can be formally defined as follows:

1. $\Gamma = \Sigma \cup \{\sqcup, x\}$
2. $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{accept}, q_{reject}\}$, the states
3. q_0, the start state
4. Then δ as given by the table:

δ	0	1	x	$\sqcup$
q_0	$(q_1, 0, L)$	$(q_1, 1, L)$	(q_1, x, L)	$(q_{accept}, \sqcup, L)$
q_1	$(q_1, 0, L)$	$(q_1, 1, L)$	(q_1, x, L)	$(q_2, \sqcup, R)$
q_2	$(q_2, 0, R)$	(q_3, x, L)	(q_2, x, R)	$(q_7, \sqcup, L)$
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_3, x, L)	$(q_4, \sqcup, R)$
q_4	(q_5, x, L)	$(q_4, 1, R)$	(q_4, x, R)	$(q_{reject}, \sqcup, L)$
q_5	$(q_5, 0, L)$	$(q_5, 1, L)$	(q_5, x, L)	$(q_6, \sqcup, R)$
q_6	(q_1, x, L)	$(q_6, 1, R)$	(q_6, x, R)	$(q_{reject}, \sqcup, L)$
q_7	$(q_7, 0, L)$	$(q_7, 1, L)$	(q_7, x, L)	$(q_8, \sqcup, L)$
q_8	$(q_{reject}, 0, R)$	$(q_{reject}, 1, R)$	(q_8, x, R)	$(q_{accept}, \sqcup, L)$

Problem 2

Let $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, \textvisiblespace, \#, x\}$. Write a Turing machine, M, such that given any input $w \in \Sigma^*$, M halts with tape content $w\#w^R$.

Part One Describe M in pseudo-code.

The pseudo-code that describes M is as follows:
$M =$" On input string w:

1. Start on the left side the word, write # at the end and move left, go to the next stage
2. If the head is on a 0, write an x and go to stage 3, if the head is on a 1 write an x and go to stage 5, if the head is on white space, go to q_{halt}
3. Move to the right until we hit empty tape, write 0 and move left, go to next stage
4. Move to the left until we hit an x, write a 0 and move left, go to stage 2
5. Move to the right until we hit empty tape, write a 1 and move left, go to next stage
6. Move to the left until we hit an x, write a 1 and move left, go to stage 2"

Part Two Formally define M, where $M = \langle \Sigma, \Gamma, Q, q_0, q_{accept}, q_{reject} \rangle$

M can be formally defined as follows:

1. $\Gamma = \Sigma \cup \{\textvisiblespace, x, \#\}$
2. $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{accept}, q_{reject}\}$, the states
3. q_0, the start state
4. Then δ as given by the table:

δ	0	1	x	#	$\textvisiblespace$
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	-	-	$(q_1, \#, L)$
q_1	(q_2, x, R)	(q_4, x, R)	-	-	$(q_{halt}, \textvisiblespace, R)$
q_2	$(q_2, 0, R)$	$(q_2, 1, R)$	-	$(q_2, \#, R)$	$(q_3, 0, L)$
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	$(q_1, 0, L)$	$(q_3, \#, L)$	-
q_4	$(q_4, 0, R)$	$(q_4, 0, R)$	-	$(q_4, \#, R)$	$(q_5, 1, L)$
q_5	$(q_5, 0, L)$	$(q_5, 1, L)$	$(q_1, 1, L)$	$(q_5, \#, L)$	

Places with a - denote that the given configuration *should* never be reached.

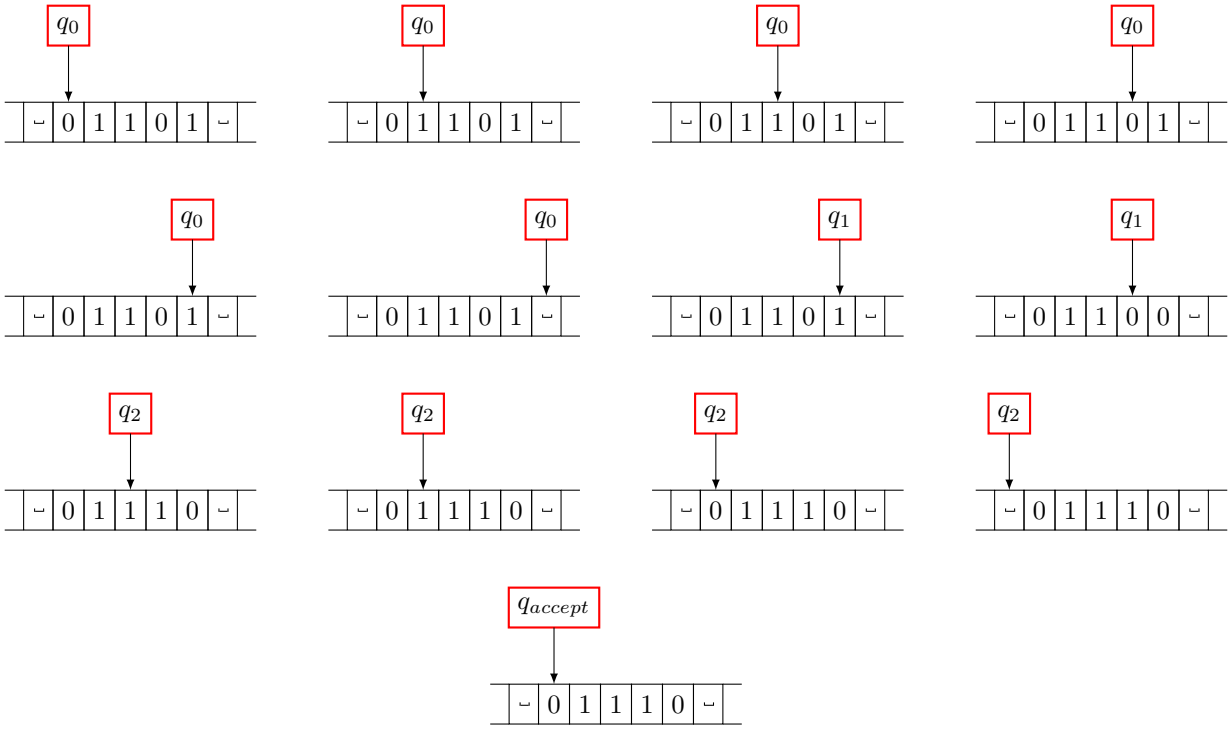
Problem 3

Let $M = \langle \Sigma, \Gamma, Q, q_0, \delta, q_{accept}, q_{reject} \rangle$ be a Turing machine where $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \sqcup\}$, $Q = \{q_0, q_1, q_2, q_3, q_{accept}, q_{reject}\}$ and δ is represented by the following table:

δ	0	1	$\sqcup$
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	$(q_1, \sqcup, L)$
q_1	$(q_2, 1, L)$	$(q_1, 0, L)$	$(q_3, 1, L)$
q_2	$(q_2, 0, L)$	$(q_1, 1, L)$	$(q_{accept}, \sqcup, R)$
q_3	-	-	$(q_{accept}, \sqcup, R)$

Part One Show the sequence of computation of M when given 01101 input.

Read from left to right, top to bottom, the sequence is as follows:



Part Two What is the functionality of M? Justify your answer.

Given a word that represents a number in binary, it will increment it by one based on the rules of binary addition.

Justification

In binary addition a 1 is added to the farthest right place value of the number. If this value is 1, it will flip it to 0 and carry to the next place value to the left.

This flipping will occur until a 0 is reached or the read/write head reaches the end of the number. It will then flip the 0 to a 1, or the space to a 1.

The result is that the binary representation of the number will be incremented by one.

Problem 4

Let L be the language of $\Sigma = \{0, 1\}$. Prove that L is Turing-decidable if and only if L can be enumerated by an enumerator Turing machine in strictly increasing order.

Proof. This proof will consist of two parts. First proving that any L that is Turing-decidable can be enumerated by an enumerator Turing machine in strictly increasing order. The second part is that an enumerator Turing machine in strictly increasing order gives a language, L, that is Turing-decidable.

Part One Proving that if a language is Turing-decidable, it can be enumerated by an enumerator Turing machine in strictly increasing order.

Let the Turing-decidable language be L decided by M and the enumerator Turing machine be E.

We can use M to help us in the construction of E.

Since L is the language decided by M, let the lexographic ordering of the strings be denoted as $s_1, s_2, s_3, \ldots$.
We can then define E as follows:
$E =$" On input string s:

1. For input, $i = 1, 2, 3, \ldots$
2. Let M run on the string, s_i
3. If M accepts the string, s_i, print it.
4. If M rejets the string, continue through the loop"

This concludes the first part of the proof.

Part Two Proving that if an enumerator Turing machine can enumerate a language in strictly increasing order, then it is Turing-decidable.

Let the Turing-decidable language be L decided by M and the enumerator Turing machine be E.

There are two cases to consider, when L is finite and when it is infinite.

When it is finite, it has to be decidable. Every finite language can be decided.

When it is inifinte, we can use E to help us in the construction of M.

The pseudo-code that describes M is as follows:
$M =$" On input string w:

1. Receive input from E, s
2. If s is the same as w, accept it
3. If s is lexicographically larger than w, reject it
4. If s is lexicographically smaller than w, go back to stage 1"

This works because w is guaranteed to be in L because it is infinite.

This concludes the second part of the proof. Thus both sides have been proven and the proof is complete. □

Problem 5

Prove that any Turing machine can be simulated by a 2-PDA.

Proof. To prove that any Turing machine can be simulated by a 2-PDA, we will use a proof by construction. We will construct a Turing machine, M, where

$$M = \langle Q_M, \Sigma_M, \Gamma_M, q_{M0}, q_{accept}, q_{reject} \rangle$$

such that it is constructed from a 2-PDA, M, where

$$P = \langle Q_P, \Sigma_P, \Gamma_P, \delta_P, q_{P0}, F_P \rangle$$

M will be defined as follows:

1. $Q_M = Q_P$, the state machine is preserved
2. $\Sigma_M = \Sigma_P$, the alphabets are identical
3. $\Gamma_M = \Gamma_P \cap \{\sqcup\}$, the stack alphabet
4. $q_{M0} = q_{P0}$, the start states are the same
5. Next we have to implement the transition function, δ_M, A Turing machine can do two movements on the tape, Left and Right. To replicate this with the 2-PDA, whenever the M is supposed to move to the left, we can just pop the value off the left stack and push it onto the right stack. Trivially we can do the same for moving right, pop off the right and push to the left.

   To handle writing new values to the tape, instead of pushing the value that was just popped, we can push any value such that value is in Γ, the stack alphabet.

The reason this works is because the first stack can be viewed as everything to the left of the read/write head, while everything on the second stack can be viewed as everything to the right of the read/write head. Since stacks are infinite in PDA's, two of them can represent the tape in a Turing machine.

It follows that a Turing machine can be simulated using a 2-PDA and thus the proof is complete.

□