Homework 5
Solution
CSCI 2670
October 6, 2005

3.2 c) New text

$q_1 1 \# \# 1$
$x q_3 \# \# 1$
$x \# q_5 \# 1$
q_{reject}

Old text

$q_1 1 \# \# 1$
$\sim q_3 \# \# 1$
$\sim \# q_5 \# 1$
q_{reject}

3.2 e) New text

$q_1 10 \# 10$
$x q_3 0 \# 10$
$x 0 q_3 \# 10$
$x 0 \# q_5 10$
$x 0 q_6 \# x 0$
$x q_7 0 \# x 0$
$q_7 x 0 \# x 0$
$x q_1 0 \# x 0$
$x x q_2 \# x 0$
$x x \# q_4 x 0$
$x x \# x q_4 0$
$x x \# q_6 x x$
$x x q_6 \# x x$
$x q_7 x \# x x$
$x x q_1 \# x x$
$x x \# q_8 x x$
$x x \# x q_8 x$
$x x \# x x q_8 \sim$
$x x \# x x q_{accept}$

Old text

$q_1 10 \# 10$
$\sim q_3 0 \# 10$
$\sim 0 q_3 \# 10$
$\sim 0 \# q_5 10$
$\sim 0 \# 1 q_5 0$
$\sim 0 \# 10 q_5 \sim$

~0#1$q_7$0
~0#1$q_7$0
~0#$q_7$10
~0q_7#10
~$q_7$0#10
q_7~0#10
~$q_9$0#10
~0q_9#10
~0#q_{11}10
~0q_{12}#x0
~q_{12}0#x0
q_{12}~0#x0
~q_{13}0#x0
~xq_8#x0
~x#q_{10}x0
~x#xq_{10}0
~x#q_{12}xx
~xq_{12}#xx
~q_{12}x#xx
q_{12}~x#xx
~q_{13}x#xx
~xq_{13}#xx
~x#q_{14}xx
~x#xq_{14}x
~x#xxq_{14}~
~x#xxq_{accept}~

3.6 The problem is that M may not halt on some string s_k. If that is the case, then E will never enumerate any string s_i with $i > k$.

3.8 b) If first symbol is a 0
Write $ and move right until the next 0
If end of string is reached **reject**
Write X and move left until $
Move right until there the next 1
If end of string is reached **reject**
Write X and move left until $
Else if first symbol is a 1
Write $ and move right until the next 0
If end of string is reached **reject**
Move right until there the next 0
If end of string is reached **reject**
Write X and move left until $
Else (first symbol must be ~) **accept**
Repeat

Move right until the next 1
    If end of string is reached
        Move left until $
        Move right until next 0
        If end of string is reached **accept**
        Else **reject**
Write X and move left until $
Move right until next 0
    If end of string is reached **reject**
Write X and move right until next 0
    If end of string is reached **reject**
Move left until $

3.12) We need to simulate a {R,RESET} Turing machine using a standard Turing machine and vice versa. First, lets assume we have a {R,RESET} machine M and we want to simulate M using a Turing machine M_1. In the descriptions below, whenever I say "mark," I mean that the current symbol at the tape head will be replaced with another symbol – each symbol has a unique replacement variable (e.g., a, b, c may be replaced with A, B, C, respectively).

Given any input, w, M_1 should do the following.

Shift all symbols right 1 space and mark tape head with some symbol not in the original tape alphabet (the "tape start marker").
Run M on w
    Whenever instructed to RESET, move left until tape start marker and move right one

Then M_1 can model M with R and RESET.

Now assume we have a standard Turing machine M_1. We want to create a {R,RESET} Turing machine M, that mimics the behavior of M_1.

Given an input w, M needs to keep track of the tape cell before the one the tape head currently points to by marking this tape cell. When the tape head is at the beginning of the tape, no cell will be marked. If we can create M_1 to do this, then M_1 will be able to move left one space (and thus mimic M_1). Below, I will first show what M needs to do at each R command to keep the cell to the left of the head marked. I will then show how to move the tape head left one cell to the left of the head is marked (and mark the cell before the new tape head).

Implementing R
RESET
Move right to marked cell
If no marked cell
    RESET

Mark tape head and move right
Else (some cell is marked
Unmark the current cell and move right
Mark the cell and move right

Implementing L
RESET
Move right to check for marked cell
If there is no marked cell, do nothing and return
If there is a marked cell do the following
RESET
Mark the first cell
Move right
If the tape head points to a marked cell, unmark it and RESET and return (this occurs if you were at the second tape cell and want to move to the first tape cell … in this case no cells should be marked)
Otherwise repeat the following loop
RESET
Move right to marked cell
Unmark the cell and move right one cell
Mark the cell
Move right one cell
If the current cell is not marked then loop
(You will only get to this point if you have two marked cells in a row … i.e., if you have marked the cell to the left of the originally marked cell)
Unmark current cell and RESET
Move right to marked cell
Move right one cell and return

Thus, you can implement L and R using R and RESET and vice versa. Therefore {L,R} Turing machines accept the same class of languages as {R,RESET} Turing machines.

3.13 To show a Turing machine with directions {R,S} is not equivalent to a standard Turing machine, we must show there is some language that is accepted by one of these models, but not the other. Since we cannot move backwards on the tape with the {R,S} machine, we have intuitively lost our memory. Therefore, it seems likely that the standard Turing machine is stronger than the {R,S} Turing machine.

Consider the language $A = \{0^n1^n \mid n > 0\}$. We have shown in class that this language is Turing recognizable. It is easy to see that there is no way to write a {R,S} Turing machine that recognizes A – we have to be able move backwards to remember how many 0's we've read to ensure we read the same number of 1's.

In fact, {R,S} Turing machines can only accept regular languages. We show this by construction. It is clear that any NFA can be simulated by a {R,S} Turning

machine. The construction below illustrates than an {R,S} Turing machine can be simulated by an NFA.

Given any {R,S} Turing machine M, we can construct an NFA N such that L(M)=L(N). Let $M = (Q,\Sigma,\Gamma,\delta,q_0,q_{accept},q_{reject})$. Then $N=(Q',\Sigma,\delta',q_{start},F)$, where $Q' = \{q_{start}\}\cup\{(q_{iR},a) \mid q_i \in Q, a \in\Sigma\}\cup\{(q_{iS},a) \mid q_i \in Q, a \in\Gamma\}$. The states of N (other than the start state) keep track of what the state of M would be and what symbol the tape head points to. The q_{iR} states are the states that were arrived at by a R move in M and the q_{iS} states were arrived at by an S move. The NFA will read an input symbol a only if it is in some state (q_{iR},a). The accept states of N are all the pairs (q_{accept},a) for some $a \in\Gamma$. Finally, the transition function mimics M's transition function as follows:

- Let $\delta'(q_{start},\varepsilon) = \{(q_{0R}, a) \mid a \in \Sigma \}$. The start state non-deterministically jumps to q_{0R} with any possible input alphabet symbol at the tape head.
- For each non-halting state q_i and each $a \in \Sigma$, if $\delta(q_i,a) = (q_j,b,R)$, then $\delta'((q_{iR},a),a) = \{(q_{jR},c) \mid c\in \Sigma\}$. You can read the a and go to state q_{jR}. After moving right, any input symbol may be at the tape head.
- For each non-halting state q_i and each $a \in \Sigma$, if $\delta(q_i,a) = (q_j,b,S)$, then $\delta'((q_{iR},a),a) = \{(q_{jS},b)\}$. You can read the a and go to state q_{jS}. The tape head will now be b.
- For each non-halting state q_i and each $a \in \Gamma$, if $\delta(q_i,a) = (q_j,b,R)$, then $\delta'((q_{iS},a),\varepsilon) = \{(q_{jR},c) \mid c\in \Sigma\}$. You do not read the a since you have already read the tape head at this cell – simply go to state q_{jR}. After moving right, any input symbol may be at the tape head.
- For each non-halting state q_i and each $a \in \Gamma$, if $\delta(q_i,a) = (q_j,b,S)$, then $\delta'((q_{iS},a),\varepsilon) = \{(q_{jS},b)\}$. You do not read the a since you have already read the tape head at this cell – simply go to state q_{jS}. The tape head will now be b.

It is clear that N mimics the behavior of M.