CSC 363 — Solutions to Test #1, Winter 2010

1) Consider the Nondeterministic Finite Automaton (NFA) with alphabet $\Sigma = \{ 0, 1 \}$, state space $Q = \{ q_0, q_1, q_2, q_3, q_4 \}$, start state q_0, set of accepting states $F = \{ q_2, q_4 \}$, and transition function δ defined as follows:

$\delta(q_0, 0) = \{q_0\}$	$\delta(q_0, 1) = \{q_1, q_3\}$	$\delta(q_0, \epsilon) = \emptyset$
$\delta(q_1, 0) = \emptyset$	$\delta(q_1, 1) = \{q_2\}$	$\delta(q_1, \epsilon) = \emptyset$
$\delta(q_2, 0) = \{q_2\}$	$\delta(q_2, 1) = \emptyset$	$\delta(q_2, \epsilon) = \emptyset$
$\delta(q_3, 0) = \{q_3\}$	$\delta(q_3, 1) = \{q_4\}$	$\delta(q_3, \epsilon) = \emptyset$
$\delta(q_4, 0) = \emptyset$	$\delta(q_4, 1) = \emptyset$	$\delta(q_4, \epsilon) = \emptyset$

1a) [20 marks] For each of the following strings, say whether or not this NFA accepts the string, and if it does accept, give the sequence of states gone though for each of the accepting branches of the computation.

- 11
  Accepts. Two accepting branches: q_0, q_1, q_2 *and* q_0, q_3, q_4.
- 1010
  Does not accept.
- 0011
  Accepts. Two accepting branches: q_0, q_0, q_0, q_1, q_2 *and* q_0, q_0, q_0, q_3, q_4.
- 11000
  Accepts. One accepting branch: $q_0, q_1, q_2, q_2, q_2, q_2$.

1b) [10 marks] Write a regular expression that describes the language recognized by this NFA.
Two possible answers (there are others too): $(0^*110^*) \cup (0^*10^*1)$ *and* $0^*1(10^* \cup 0^*1)$.

2) Define a *2-tape One-Way-Read-Only-Input Turing Machine* (a 2-OWROI Turing Machine) as follows. It has two tapes, with the input string being stored on tape 1 (followed by a blank), and tape 2 initialized to all blanks (infinitely far to the right). Each tape has a head that is initially at the leftmost square. There is a finite-state control unit as for regular Turing Machines. Transitions are determined by the current state and the two symbols on the squares seen by the heads on the two tapes. In each transition, the square under the head on tape 2 is replaced by a new symbol (which may be the same as the old). The head for tape 1 either stays on the same square (S), or moves to the right (R), except a move right does nothing when the head is at the blank symbol after the input. Note that the head for tape 1 *cannot* move left. The head for tape 2 either moves right (R) or left (L), with a move left when the tape is on the leftmost square doing nothing. The transition function is therefore of the following form:

$$\delta : \quad Q \times \Gamma^2 \rightarrow Q \times \Gamma \times \{ R, S \} \times \{ L, R \}$$

Note that the symbols on tape 1 are never changed from the initial string. A 2-OWROI Turing Machine has accept and reject states like a regular Turing Machine, and accepts an input string if and only if it ever enters the accept state, as for a regular Turing Machine.

2a) [20 Marks] Prove that every language that is recognizable by a regular Turing Machine is recognizable by a 2-OWROI Turing. You should be explicit about the logic of the proof. You should give implementation-level details of any Turing Machine construction you use

(but needn't specify every transition in complete detail if not necessary for understanding), and explicitly state what state space and tape alphabet are used.

If a language L is recognizable by a regular TM, then some regular TM, say M, recognizes it. We can show that L is recognizable by a 2-OWROI Turing Machine by showing that a 2-OWROI TM that recognizes L can be produced by some procedure that modifies M to produce a 2-OWROI Turing Machine M'. There are many ways that one could produce such an M' from M. Here is one.

We'll let M' have the same tape alphabet as M. (Solutions where M' has a bigger tape alphabet are also possible.) The state space of M' will consist of all the states of M plus a fixed number of extra states (whose number does not depend on what M is) that are needed to implement the following operations of M':

1) *Remember the first symbol on tape 1 in the finite control. To do this, the set of extra states in M' will be of the form $\{i\} \cup (\Gamma \times \{s_1, s_2, \ldots, s_k\})$, where Γ is the tape alphabet of M, and i is the start state of M'. Every extra state of M' (ie, that's not a state of M) other than its start state will be a pair in which the first element of the pair is remembering the first symbol on the tape.*
2) *Move right one square on both tape 1 and tape 2, and then copy the symbols from tape 1 to tape 2, until the blank symbol on tape 1 is reached.*
3) *Move left on tape 2 until the blank that is still in the first square is reached.*
4) *Write the first symbol from tape 1 that was remembered in stage 1 to the first square of tape 2, move left on tape 2 (which will leave the head at the first (leftmost) square), and transition to the start state of M.*

The transition function for the states of M' corresponding to those of M ignores tape 1, and does with tape 2 exactly what M does with its single tape; state transitions (including to the accept and reject states) also correspond to those for M. Since stages 1 to 4 above also set up tape 2 of M' to have the same content and same head position as M does initially, M' will accept, reject, or loop in exactly the same way as M does. M' therefore recognizes the same language as M.

2b) [15 Marks] Prove that every language that is recognizable by a 2-OWROI Turing Machine is recognizable by a regular Turing Machine. For this proof, you may *not* use the fact (discussed in the text and in class) that k-tape Turing Machines have the same power as regular Turing Machines, though your proof might use ideas similar to those used in the proof of that. You should be explicit about the logic of the proof. You may describe the operation of any Turing Machine you construct at a high level (similar to that used in the proofs in Sipser's book), but you should explicitly state what tape alphabet is used.

Note: This question involves more details than other questions on the test, so you may want to leave it to the end, or write down just your basic idea (for part marks) and fill in the details if you have time.

If a language L is recognizable by a 2-OWROI TM, then some 2-OWROI TM, say M, recognizes it. We can show that L is recognizable by a regular Turing Machine by showing that a regular TM that recognizes L can be produced by some procedure that modifies the 2-OWROI TM M to produce a regular Turing Machine M'.

There are many ways that one could produce such an M' from M. Here I will describe a way in which the single tape of M' holds the contents of tape 1 of M, followed by a blank, followed by the contents of tape 2 of M. The tape alphabet for M' will be as follows:

$$\Gamma' \;=\; \Gamma \times \{0,1,2\} \times \{0,1\}$$

where Γ is the tape alphabet for M. So a tape alphabet symbol in Γ' is a triple of the form (s,h,b). Here, s is the symbol that would be on tape 1 or 2 of M. The head positions

are marked using h, with $h = 1$ marking the position of the head for tape 1 of M, $h = 2$ marking the position of the head for tape 2 of M, and $h = 0$ for squares where neither head is positioned. The first square on tape 2 is marked with $b = 1$, with $b = 0$ for other squares. The set of states for M' will be

$$Q' \;=\; \{s_0, s_1, \ldots\, s_m\} \cup (Q \times \{t_0, t_1, \ldots, t_n\})$$

Here, $s_0, s_1, \ldots, s_m$ are states needed to implement the initial operations described below, with s_0 being the start state of Q'. After the initial operations, control is transferred to (q_0, t_0), where q_0 is the start state of Q. Each transition of M, from state q_i, is mimicked using states $(q_i, t_0), (q_i, t_1), \ldots, (q_i, t_n)$ of Q', with the substates with $t_0, t_1, \ldots, t_n$ being used to implement the operations described below.

The initial operations of M' are as follows:

1) Mark the start of the tape with $h = 1$.

2) Move right to the blank at the end of the input string, then right one more square. Mark that square with $h = 2$ and $b = 1$.

3) Go to the state (q_0, t_0), which begins the process of mimicking the operation of M.

A transition of M from state q_i is mimicked as follows, using states $(q_i, t_0), (q_i, t_1), \ldots, (q_i, t_n)$ of M':

a) If q_i is the accept state of M, accept. If q_i is the reject state of M, reject.

b) The tape head should already be positioned at the square marked with $h = 2$. Remember the contents of this square in the finite control (ie, using $t_0, \ldots, t_n$).

c) Move left until the square marked with $h = 1$ is found.

d) Based on the symbol corresponding to that on tape 2 of M remembered at stage (b), and the symbol corresponding to that on tape 1 of M that is stored where the head of M' is currently positioned, and on q_i, mimic the action that M would do. Specifically,

- *If M would move right on tape 1, and the current square does not contain the blank symbol, rewrite the current square to unmark it (ie, set $h = 0$), move right, and mark that square with $h = 1$.*
- *Move right until the square marked with $h = 2$ is found. Rewrite this square with $h = 0$ to unmark it.*
- *If M would move left on tape 2, and the current square is not marked with $b = 1$, move left, and mark that square with $h = 2$. Otherwise, M would move right on tape 2, so move right and mark that square with $h = 2$.*
- *Transition to state (q_j, t_0), where q_j is the state that M would transition to.*

3) Define the language $A_{TM\times 2}$ as follows:

$$A_{TM\times 2} \;=\; \{\, \langle M_1, M_2, w\rangle \,|\, M_1 \text{ and } M_2 \text{ are Turing Machines that both accept the string } w \,\}$$

3a) [20 marks] Prove that $A_{TM\times 2}$ is recognizable.

$A_{TM\times 2}$ can be recognized by a TM that operates on input u as follows:

1) Reject if u is not a syntactically valid string of the form $\langle M_1, M_2, w\rangle$.

2) Simulate the operation of M_1 on input w. If M_1 rejects, reject. If M_1 accepts, go on to the next step. (Note that it is possible that this simulation will loop forever.)

3) Simulate the operation of M_2 on input w. If M_2 rejects, reject. If M_2 accepts, accept. (Note that it is possible that this simulation will loop forever.)

It is clear that this machine accepts its input if and only if this input is a valid description of the form $\langle M_1, M_2, w\rangle$ and both M_1 and M_2 accept w, which is what is needed for a recognizer of $A_{TM\times 2}$. (It doesn't matter whether the machine rejects or loops on strings that are not in $A_{TM\times 2}$.)

There are other possible answers. In particular, one could simulate M_1 and M_2 in parallel, and accept only if both accept. But sequential simulation as above is simpler and works just as well.

3b) [15 marks] Prove that $A_{TM\times 2}$ is not decidable.

One way of proving it: Adapt the proof that A_{TM} is not decidable.

Suppose that some Turing Machine, say H, decides $A_{TM\times 2}$. Define a Turing Machine, D, that operates as follows on input u:

1) *Reject if u is not a syntactically valid string of the form $\langle M\rangle$, where M is a Turing Machine.*
2) *Run H on the input $\langle M, M, \langle M\rangle\rangle$. Accept if H rejects, and reject if H accepts. (Note that since H is a decider, it never loops.)*

Now consider what D does when given $\langle D\rangle$ as input.

- *If D accepts $\langle D\rangle$, then H must reject $\langle D, D, \langle D\rangle\rangle$, which means that $\langle D\rangle$ is not accepted by both D and D, which means that $\langle D\rangle$ is not accepted by D. This is a contradiction.*
- *But if D rejects $\langle D\rangle$, then H must accept $\langle D, D, \langle D\rangle\rangle$, which means that $\langle D\rangle$ is accepted by both D and D, which means that $\langle D\rangle$ is accepted by D. This is also a contradiction.*
- *Finally, D never loops, since H is a decider.*

Since any action of D when given $\langle D\rangle$ leads to a contradiction, the assumption that a decider for $A_{TM\times 2}$ exists must be incorrect, and hence $A_{TM\times 2}$ is undecidable.

Another way of proving it: Reduce the problem of deciding

$$A_{TM} \;=\; \{\, \langle M, w\rangle \mid M \text{ is a TM that accepts } w \,\}$$

(which we know is undecidable) to that of deciding $A_{TM\times 2}$.

Suppose that $A_{TM\times 2}$ is decidable. Then some TM, say S, must decide it. Using S, we could decide A_{TM} by a TM that operates as follows on input u:

1) *Reject if u is not a syntactically valid string of the form $\langle M, w\rangle$.*
2) *Run S on the input $\langle M, M, w\rangle$. Accept if S accepts, and reject if S rejects. (Note that since S is a decider, it never loops.)*

This will decide A_{TM} because S will accept in stage (2) iff both M and M accept w, which is the same as M accepting w. But we know that A_{TM} is undecidable (Theorem 4.11), so we have a contradiction. So the assumption that some TM decides $A_{TM\times 2}$ must be wrong, and instead $A_{TM\times 2}$ must be undecidable.