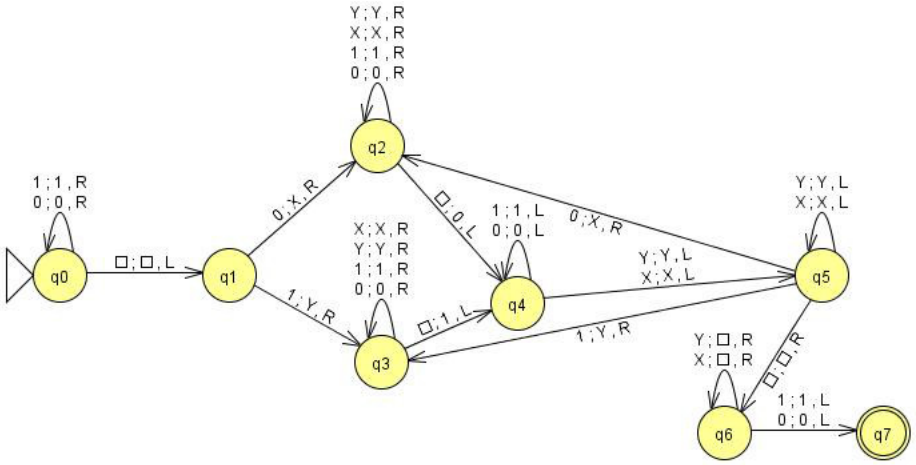
Theory of Computation

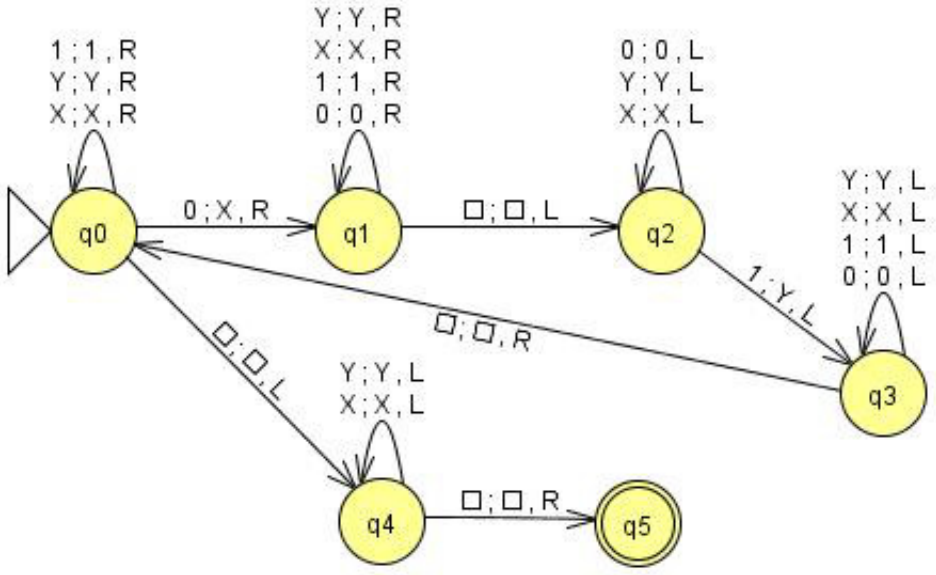
CS-312 **Summer 2014**
Exam 3 **Solution Set**

In all problems on this exam, let $\Sigma = \{0, 1\}$ be the input alphabet from which strings are produced.

1. *15 points. Give a detailed description of a Turing machine which, when started with the representation of a binary string w on its tape, halts with the binary representation of w^R on its tape, where w^R is the reversal of w. For example, when started with* 011101 *on its tape, the machine should end with* 101110 *on its tape.*



2. *15 points. Give a detailed description of a Turing machine which decides the language of binary strings which have an equal number of* 0*s and* 1*s. (Note that the* 0*s and* 1*s may appear in the string in any order.)*



3. *20 points. A* 2-PDA *is a pushdown automata that has two stacks. (You explored this model in a homework assignment this term.)*

   Show that 2-PDAs are equivalent to Turing machines. (Note that this an "iff" claim and thus requires two different proofs; each 2-PDA can be simulated by a Turing machine, and each Turing machine can be simulated by a 2-PDA.)

   - A 2-PDA can be simulated by a Turing machine with three tapes. The first tape will be used as a read-only input tape, where the tape head will only move to the right. The second two tapes will be used to hold the two stacks of the PDA (one per tape). Each move of the 2-PDA will be simulated by a corresponding move of the Turing machine, updating the tape heads and contents of the three tapes as appropriate.
   - A Turing machine can be simulated by a 2-PDA, using the stacks to contain the contents of the visible portion of the Turing's input tape. One stack will contain the symbols seen to the left of the tape head; the other will contain the symbols seen to the right of the tape head (including the tape head itself). Every move of the Turing machine's tape head can be simulated by popping a symbol from one stack and pushing it onto the other. If one stack becomes empty, blank symbols can be pushed onto it as needed.

4. A *useful symbol in a Turing machine is a symbol of the tape alphabet that can be written to the tape of that machine during its execution. That is, σ is a useful symbol of Turing machine M if there is some input w for which, when M is started on w, M eventually writes σ to the tape.*

   Let L be the set of pairs (M, σ), where M is a Turing machine, and σ a tape symbol of M, such that σ is a useful symbol of M.

   (a) *10 points. Show that L is recursively-enumerable (*i.e.*, Turing-acceptable).*

   Let $w_1, w_2, w_3, \ldots$ be an enumeration of the strings of Σ^*. Start by simulating M on one step with input w_1, two steps with inputs w_1 and w_2, and so on. In each simulation, monitor the behavior of M to see if it prints σ; if so, halt and accept. In all other cases, continue the simulation (even if M halts).

   Note that if σ is a useful symbol of M, there will eventually be some input w_i which will print σ after j steps. Let $n = \max(i, j)$. At step n of the algorithm above, when M simulates $w_1, \ldots, w_n$ for n steps each, it will simulate w_i for j steps and print σ, guaranteeing acceptance.

   (b) *20 points. Show that L is not recursive (*i.e.*, undecidable).*

   Suppose L were decided by some Turing machine D. Then we can decide the language $A_{TM} = \{(M, w) : M \text{ accepts } w\}$ as follows.

   Let (M, w) be a pair for which we want to decide if M accepts w. Create a new turing machine M' by modifing M as follows:

- For every transition that enters an accept state, add a new transition which also enters that state, but writes a special symbol x to the tape (where x is not in the original tape alphabet of M).
- M' begins by erasing its input, writing w on its input tape, and then begins simulating M (modified as noted above).

Notice that M' will print an x iff M accepts w. Thus, deciding if x is a useful symbol of M' is equivalent to deciding A_{TM}, which we know to be undecidable.

5. *Given a graph G, an* independent set *is a subset of the vertices of G such that no two vertices in the set are connected by an edge. The* independent set problem *is defined as follows: given a graph G and an integer k, does G contain an independent set of size at least k?*

   (a) *10 points. Show that the independent set problem is in $\mathcal{NP}$.*

   To see if (G, k) is a positive instance of the independent set problem, nondeterministically guess an independent set of size k, and then verify that all k^2 possible edges between members of that set are not present. Obviously, this takes quadratic time at most.

   (b) *10 points. Show that the independent set problem is $\mathcal{NP}$-complete, by reducing any known $\mathcal{NP}$-complete problem to the independent set problem.*

   We reduce from the k-clique problem. Suppose we want to decide if G has a clique of size k. Modify G into a new graph G' in which G is "inverted": (x, y) is an edge in G' iff (x, y) is not an edge of G. Observe that G has a clique of size k exactly when G' has an independent set of size k (the same set of nodes serves as the desired set in both cases).