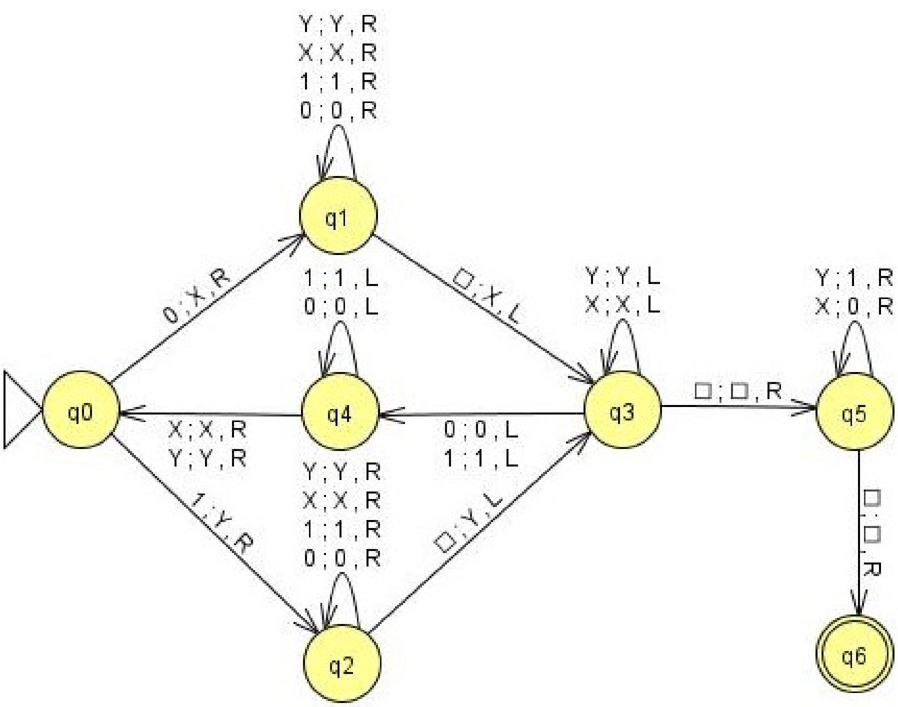
**Theory of Computation**

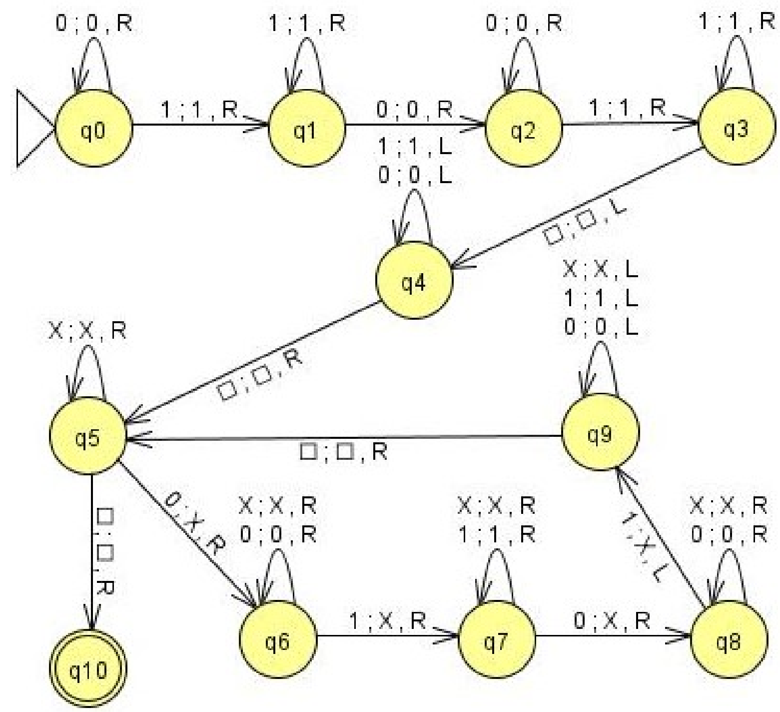
**CS-312** **Fall 2023**

**Exam 3** **Solution Set**

1. *15 points. Let $w \in \{0,1\}^*$ be a non-empty binary string. Give a detailed description (*i.e. *a state-machine diagram) of a deterministic one-tape Turing machine which, when started with $w$ on the tape as input, halts with the binary string $ww$ on its tape. For example, when started with* 01101 *on its tape, the machine should halt with* 0110101101 *on its tape.*



2. *15 points. Give a detailed description (*i.e. *a state-machine diagram) of a deterministic one-tape Turing machine which decides the language* $\{0^n1^n0^n1^n : n \geq 0\}$. *For example,* 000111000111 *is in this language.*



3. *15 points. A* Chalkboard Turing Machine (CTM) *is a Turing machine with the following restriction: the machine may never overwrite a non-blank symbol with a blank symbol. A CTM may leave an existing blank symbol unchanged; however, once a blank symbol has been overwritten by a non-blank symbol, it cannot be restored to a blank symbol again. (Symbolically: if $\delta(q, \alpha) = (r, \Box, d)$ is a transition of the CTM, then $\alpha = \Box$.)*

   *Show that Chalkboard Turing Machines are equivalent to classic Turing machines.*

   *(Note that this is an "iff" claim and thus requires two proofs.)*

   - Every Chalkboard Turing Machine can be directly simulated by a classic Turing machine without modification.
   - Every classic Turing machine can be transformed into a Chalkboard Turing Machine by doing the following:
     - Add a new symbol $\Box'$ to the tape alphabet.
     - Replace every transition that writes '$\Box$' to the tape with another that writes $\Box'$. That is, for every transition $(q, \alpha) \to (r, \Box, d)$, replace that transition with the similar transition $(q, \alpha) \to (r, \Box', d)$.
     - Duplicate every transition that reads $\Box$ with another that reads $\Box'$, performing the same action. That is, for every transition $(q, \Box) \to (r, \beta, d)$, add an additional transition $(q, \Box') \to (r, \beta, d)$.

     This new machine is a Chalkboard Turing Machine that performs the exact same computation as the original machine, except that the "official" blank symbol $\Box$ is gradually replaced by the pseudo-blank symbol $\Box'$.

4. *Let $L$ be the set of Turing machines that reject at least two different input strings.*

   (a) *15 points. Show that $L$ is recursively-enumerable (*i.e., *Turing-acceptable).*

   let $M$ be a Turing machine for which we want to decide if $M \in L$. Let $w_0, w_1, w_2, \ldots$ be a shortlex enumeration of the strings of the input alphabet of $M$. Simulate $M$ on strings $(w_0, w_1, \ldots w_i)$ for $i$ steps, increasing $i$ each time. If $M \in L$, then there are strings $(w_x, w_y)$ which $M$ rejects. Eventually, $i$ will reach a large enough value to run $w_x$ and $w_y$ long enough to reach a reject state on both strings; when this happens, halt the simulation and accept $M$. Otherwise, continue the simulation forever. This algorithm accepts every machine in $L$, and does not accept any other machines. Thus, this algorithm accepts $L$.

   (b) *15 points. Show that $L$ is not recursive (*i.e., *undecidable).*

   Suppose by contradiction that $L$ is decidable. We show how to use this decision algorithm to decide $A_{tm}$.

   Let $< M, w >$ be a pair for which we want to decide if $< M, w > \in A_{tm}$; that is, we want to decide if $M$ accepts $w$. Construct a Turing machine $D$ which operates by erasing its input (whatever it is), writes $w$ to its tape, and then runs $M$. If $M$ accepts $w$, $D$ halts and rejects. If $M$ rejects $w$, then $D$ enters an infinite loop.

Observe that:

- If $M$ accepts $w$, $D$ rejects all strings, and therefore rejects at least two strings.
- If $M$ does not accept $w$, $D$ rejects no strings.

Thus, $D \in L$ if and only if $< M, w > \in A_{tm}$. This is a decision algorithm for $A_{tm}$, which we know is undecidable.

5. *Recall the definition of the $\mathcal{NP}$-complete problem SATISFIABILITY: given a Boolean propositional logic formula $\phi$, does $\phi$ have a satisfying truth assignment?*

   *We define a related problem called DOUBLE-SAT: given a Boolean propositional logic formula $\phi$, does $\phi$ have two different satisfying truth assignments?*

   (a) *10 points. Show that DOUBLE-SAT is in $\mathcal{NP}$.*

   Let $\phi$ be a formula for which we want to decide if $\phi$ has two different satisfying truth assignments. Nondeterministically guest two different truth assignments, then verify that (a) they are different from each other and (b) each satisfies $\phi$. Both of these tasks can be completed in polynomial time.

   (b) *15 points. Show that DOUBLE-SAT is $\mathcal{NP}$-complete, by reducing SATISFIABILITY to DOUBLE-SAT. That is, show how to use an algorithm that solves DOUBLE-SAT to solve SATISFIABILITY.*

   Let $\phi$ be a formula for which we want to decide if $\phi$ has a satisfying truth assignment. Construct a new formula $\phi' = \phi \vee x$, where $x$ is a new variable not present in $\phi$.

   We claim $\phi'$ has two satsifying assignments iff $\phi$ has one satisfying assignment. Observe:

   - If $\phi$ is satisfiable, $\phi'$ has two satisfying assignments: $\phi$'s satisfying assignment with $x$ set to either *true* or *false*.
   - If $\phi$ is not satisfiable, $\phi'$ has at most one satisfying assignment (when $x$ is set to *true*).

   Thus, this construction reduces SATISFIABILITY to DOUBLE-SAT.

6. *Extra Credit (worth 1 point added to final course grade):*

   *The five homework groups for this course were named after colors: black, blue, green, red, and yellow. These five colors are used together in a commonly-recognized symbol. What is that symbol?*

   These are the colors of the Olympic rings.