Homework Assignment 6 Answers

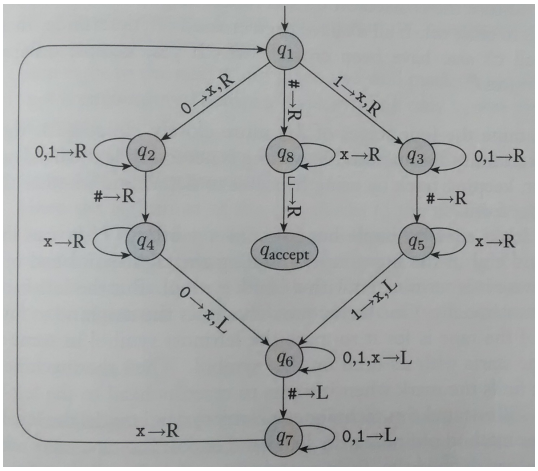
CSCI 2670 Introduction to Theory of Computing, Fall 2016

December 2, 2016

This homework assignment is about Turing machines, decidable languages, Turing recognizable languages, and (un)decidability

There are 7 questions, but Q6 is not required to be answered, with total 100 points.

1. Consider the following Turing machine, which recognizes language $L = \{w\#w : w \in \{0,1\}^*\}$.



For each part in (a) and (b), give the sequence of configurations that the TM enters when started on the indicated string.

(a) 10#11 (b) 10#10

Answers: (15 points in total)

Two kinds of knowledge are needed to answer this question.

- configurations of a TM on an input,
- how transitions work in a TM.

(a) The Turing machine has the following configuration changes on input 10#11:

$q_1 10\#11$
$\times q_1 0\#11$
$\times 0 q_3 \#11$
$\times 0 q_3 \#11$
$\times 0\# q_5 11$
$\times 0 q_6 \# \times 1$
$\times q_7 0\# \times 1$
$q_7 \times 0\# \times 1$
$\times q_1 0\# \times 1$
$\times \times q_2 \# \times 1$
$\times \times \# q_4 \times 1$
$\times \times \# \times q_4 1$ (The TM cannot proceed, because expecting 0. So reject the input)

(b) The Turing machine has the following configuration changes on input 10#11:

$q_1 10\#10$
$\times q_1 0\#10$
$\times 0 q_3 \#10$
$\times 0 q_3 \#10$
$\times 0\# q_5 10$
$\times 0 q_6 \# \times 0$
$\times q_7 0\# \times 0$
$q_7 \times 0\# \times 0$
$\times q_1 0\# \times 0$
$\times \times q_2 \# \times 0$
$\times \times \# q_4 \times 0$
$\times \times \# \times q_4 0$
$\times \times \# q_6 \times \times$
$\times \times q_6 \# \times \times$
$\times q_7 \times \# \times \times$
$\times \times q_1 \# \times \times$
$\times \times \# q_8 \times \times$
$\times \times \# \times q_8 \times$
$\times \times \# \times \times q_8 \sqcup$
$\times \times \# \times \times \sqcup q_{\text{accept}} \sqcup$

2. Design a Turing machine that recognizes the following language

$$\{w : w \in \{0,1\}^* \text{ contains at least two same symbols, with one at the end }\}$$

You need to answer this question by taking the following steps:

(1) outline how a TM may accept such a string w;

(2) give more details to each of the steps outlined in (1);

(3) draw a Turing machine diagram based on (2).

Answers: (20 points in total)

This and Q3 questions are designed to "force" the students to work toward designing a TM with the good habit of algorithm crafting.

(1) **(3 points)** This part should be statements that describe briefly how the TM recognizes such strings.

example answer:

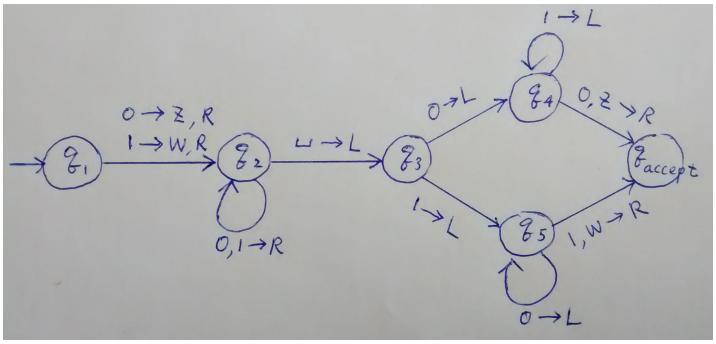
- *scan the input all the way to the rightmost symbol and memorize it;*
- *scan leftward to find a matched symbol, accept if found or reject else.*

(2) **(5 points)** This part is a refinement of (1) to some technical steps that are detailed enough to allow a direct translation to a TM diagram for (3).

example answer:

- *remember the leftmost position, e.g., by changing 1 to W and 0 to Z;*
- *scan rightward to skip all 0s and 1s until seeing* $\sqcup$*;*
- *according to the symbol on the left of* $\sqcup$*, branch to two different states;*
- *scan leftward to find a match symbol (it could be the leftmost symbol);*
- *accept if there is a match and halt;*
- *reject if even the leftmost is not a match; halt.*

(3) **(12 points)** An example TM diagram, based on (2).



3. Give an implementation-level description of a Turing machine for the following language

$$\{w\#w^{\mathcal{R}} : w \in \{0,1\}^* \text{ and } w^{\mathcal{R}} \text{ is the reversed string of } w\}$$

You need to answer this question by taking the following steps:

(1) outline how a TM may accept string $w\#w^{\mathcal{R}}$;

(2) give more details to each of the steps outlined in (1);

(3) draw a Turing machine diagram based on (2).

Answers: (20 points in total)

Same criteria for Q2 are applied to Q3.

(1) **(3 points)** This part should be statements that describe briefly how the TM recognizes such strings.

example answer:

- *compare the pair of symbols on the left and right of #;*
- *repeat the comparison for all other pairs in the inside-out fashion;*
- *reject if any pair do not match, halt;*
- *accept if all pairs match, halt.*

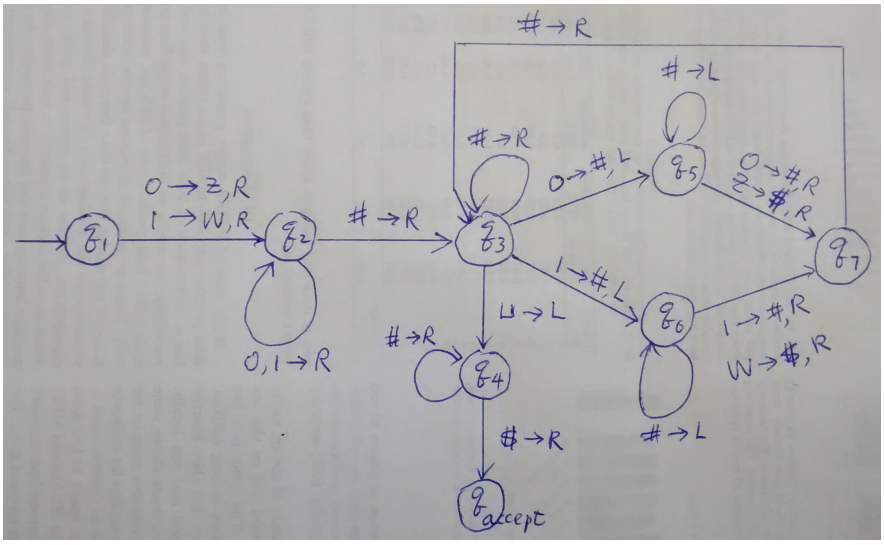
(2) **(5 points)** This part is a refinement of (1) to some technical steps that are detailed enough to allow a direct translation to a TM diagram for (3).

example answer:

0. replace the leftmost symbol, from 1 to W and from 0 to Z;
1. scan rightward to pass all # symbols to read the first non-# symbol;
2. **if it is ⊔, go to step 8***;*
3. otherwise, according to the symbol, branching to two different states;
and replace the symbol with #;
4. scan leftward to pass all # symbols to read the first non-# symbol;
5. compare it with the one memorized (including the cases of Z and W);
6. if not matched, reject and halt;
*7. otherwise replace the symbol with # (***but replace** Z **and** W **with** $\$$*),*
and go to step 1.

8. **scan leftward to pass all # symbols, reject if there are non-# symbols;**
9. otherwise accepts.

(3) **(12 points)** An example TM diagram, based on (2). See figure in the next page.



4. A 2-PDA is a Push Down Automaton equipped with two stacks, in which the tops of the two stacks (instead of one stack) may be updated at each step. Prove that 2-PDAs are as powerful as Turing machines. (*Hint*: place the two stacks head-to-head to simulate the work tape of a Turing machine so the joint point of the two stacks simulates the position of the read-head of the TM.)

Answers: **(10 points in total)**

A hint has been given. You just need to add a bit of additional details about how to use the two stacks to simulate every transition of the given TM. A typical transition is

$$a \to b, M \quad \text{where } a, b \in \Sigma \text{ and } M \in \{L, R\}$$

which means the current symbol a is replaced with b and move the read-head to direction M.

A bit of details:

- the tape content to the left of the read-head (inclusive) is stored in stack 1 (S_1); with the current symbol being the top of S_1;
- the tape content to the right of the read-head (exclusive) is stored in stack 1 (S_2); with the symbol on the right of the current symbol being the top of S_2;
- simulation of $a \to b, R$ can be done with

$$POP(S_1); PUSH(S_1, b); PUSH(S_1, POP(S_2))$$

- while simulation of $a \to b, L$ can be done with

$$POP(S_1); PUSH(S_2, b)$$

5. Show that the class of decidable languages is closed under the following operations:

   (a) concatenation,

   (b) complementation,

   (c) intersection,

   Is the class of Turing recognizable languages also closed under all the above operations? Why?

Answers: (20 points in total)

5 points for each of (a)-(c) and the additional question.

Answers:

(a) Let L_1 and $L2$ be two decidable languages and M_1 and M_2 be two deciders (i.e., Turing machines that halt) for L_1 and L_2, respectively. The following decider M recognizes L_1L_2:

> *Given input* w,
> M *nondeterministically partitions* $w = w_1w_2$,
> M *calls* M_1 *to run on* w_1 *and calls* M_2 *to runs on* w_2,
> M *accepts* w *iff both* M_1 *accepts* w_1 *and* M_2 *accepts* w_2.

Because M_1 and M_2 halt, so does M.

(b) Let L_1 be a decidable language and M_1 be a decider for L_1. The following decider M recognizes $\overline{L_1}$:

> *Given input* w,
> M *calls* M_1 *to run on* w,
> M *accepts* w *iff* M_1 *rejects* w_1.

Because M_1 halts, so does M.

(c) Let L_1 and $L2$ be two decidable languages and M_1 and M_2 be two deciders (i.e., Turing machines that halt) for L_1 and L_2, respectively. The following decider M recognizes $L_1 \cap L_2$:

> *Given input* w,
> M *calls* M_1 *to run on* w *and calls* M_2 *to runs on* w,
> M *accepts* w *iff both* M_1 *accepts* w *and* M_2 *accepts* w.

Because M_1 and M_2 halt, so does M.

- *Answer to the additional question*: The class of Turing-recognizable languages is closed under operations concatenation and intersection because in the proof (a) and (c), M runs forever iff either M_1 or M_2 runs forever.

  But the class is not closed under complementation. This is because in (b) M is supposed to accept the input if M_1 runs forever (on the input) but M does not know if M_1 is going to stop eventually.

6. Question 4.1, page 210 (**exercise only**, you do NOT need to turn in your answers).

   Answers: Answers are in the book. You would like to do the exercise because you may see this type of questions again in the final exam!

7. Question 4.2, page 211 (*Hint*: use theorem 4.5 and theorem 1.54 (or lemma 1.55)).

Answers: **(15 points in total)**

First, we express the problem as a language:

$EQ_{DFA,RE} = \{\langle A, E\rangle |$ DFA A recognizes the same language described by regular expression $E\}$

Then we prove that Language $EQ_{DFA,RE}$ is decidable because we can construct a decider TM M to recognizes $EQ_{DFA,RE}$.

> *On input* $\langle A, E\rangle$,
> M *transforms* E *to an DFA* B (Lemma 1.55, Theorem 1.54),
> M *then calls the decider* C *for language* EQ_{DFA} *on input* $\langle A, B\rangle$ (Theorem 4.5)
> M *accepts* $\langle A, E\rangle$ *iff* C *accepts* $\langle A, B\rangle$.

Since all above steps are finite, M always halts.