Assignment 4

CS 311, Spring 2015
Due: May 20, 2014

Problem 1 A *Turing machine with left reset* is similar to an ordinary Turing machine, but the transition function has the form

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{R, RESET\}$$

If $\delta(q, a) = (r, b, RESET)$, when the machine is in state q reading an a, the machine's head jumps to the left-hand end of the tape after it writes b on the tape and enters state r. Note that these machines do not have the usual ability to move the head one symbol left. Show that Turing machines with left reset recognize the class of Turing-recognizable languages.

(Hint: Much like previous problems in this class, you'll need to describe some general construction that takes a Turing Machine-with-reset to an ordinary Turing Machine and visa-versa)

Problem 2 Give the informal descriptions for Turing machines that decide the following languages

a) $\{w \mid w \text{ contains twice as many 0s as 1s }\}$

$M =$ "On input string w:

1. Scan tape and mark the first unmarked 0. If no unmarked 0 is found, jump to step 4.
2. Continue to the right and mark the next unmarked 0, then return read-head to start of tape. If no unmarked 0 is found, *reject*.
3. Scan tape and mark the next unmarked 1. If no unmarked 1 is found, *reject*. Otherwise, go move read-head to start of tape, and go back to step 1.
4. Scan the tape to look for unmarked 1s. If there are any unmarked 1s left, *reject*. If there are not any unmarked 1s, *accept*."

b) $\{a + b = c \mid a, b, c \in \{0, 1\}*$ and the binary numbers represented by a and b sum to $c\}$

1. Scan tape to check if string is in the format: $\{0, 1\}^* + \{0, 1\}^* = \{0, 1\}^*$. If it isn't, *reject*. Move read-head back to start of tape.
2. Scan tape to check if all the 0s and 1s before the = symbol have been crossed off. If so, go to step 9. Scan right until either an X (X means crossed out symbol) or a + is found. Move left by one position. Cross off the symbol at the read-head. If the symbol was a 0, or no unmarked symbol was found, go to step 3. If the symbol was a 1, go to step 4.
3. Move past the + sign and scan until either an X or an = is found. Move left by one position. Cross off the symbol at the read-head. If the symbol was a 0 go to step 5. If the symbol was a 1, go to step 6. If no unmarked symbol was found between the + and the = sign, go to step 5.
4. Move past the + sign and scan until either an X or an = is found. Move left by one position. Cross off the symbol at the read-head. If the symbol was a 0 go to step 6. If the symbol was a 1, go to step 7. If no unmarked symbol was found between the + and the = sign, go to step 6 .
5. (case: $0 + 0 = 0$) Move past the = sign and scan until either an X or an *space* is found. Move left by one position. Cross off the symbol at the read-head. If the symbol was a 0, move read-head back to start of tape, and go to step 2. If the symbol was a 1, or no unmarked symbol was found, *reject*.

6. (case: $0+1=1$ or $1+0=1$) Move past the $=$ sign and scan until either an X or a *space* is found. Move left by one position. Cross off the symbol at the read-head. If the symbol at the read-head was a 1, move read-head back to start of tape and go to step 2. If the symbol was a 0, or no unmarked symbol was found, *reject*.
7. (case: $1+1=10$) Move past the $=$ sign and scan until either an X or a *space* is found. Move left by one position. Cross off the symbol at the read-head. If the symbol at the read-head was a 0, move to the left until arriving at the $+$ sign. If the symbol at the read-head was a 1, or no unmarked symbol was found, *reject*.
8. Scan to the left and find the first unmarked symbol. If the symbol is a 0, write a 1 in its place. If the symbol is a 1, write a 0 in its place, and repeat this step(step 8). If no unmarked symbol is found, shift the contents of the entire tape to the right by one position, and fill the vacant position at the start of the tape with 1. Go to step 2.
9. Scan right to check if all symbols after the $=$ have been crossed off. If they have, *accept*. Otherwise, *reject*."

Problem 3 Show that the Turing-decidable languages are closed under

a) union: Let L be the union of two Turing-decidable languages, L_1 and L_2. A language is decidable if and only if some non-deterministic Turing machine decides it(Corollary 3.19). Therefore it follows that for languages L_1 and L_2, there exist Turing machines M_1 and M_2 that decide them.

   $L = L_1 \cup L_2$ is Turing-decidable if and only if there is some non-deterministic Turing machine that decides it. Any non-deterministic Turing machine is automatically a deterministic Turing machine. So, to show that the Turing-decidable languages are closed under union, we describe a Turing machine M that decides L.

$$L = \{w \mid w \in L_1 \cup L_2, M_1 \text{ and } M_2 \text{ decide } L_1 \text{ and } L_2\}$$

   $M =$ "On input string w:

   1. Run M_1 on w. If it accepts, *accept*.
   2. Run M_2 on w. If it accepts, *accept*.
   3. If both M_1 and M_2 halt and reject, *reject*."

   Since M will always halt – end up in either an accept or reject state, L is decidable.

b) intersection: For any two Turing-decidable languages L_1 and L_2, there are Turing machines that decide them. Let these machines be M_1 and M_2. To prove that Turing-decidable languages are closed intersection, we construct a Turing machine M that decides $L_1 \cap L_2$.
   Let w be any string that contains symbols in languages L_1 and L_2. Then,
   1. Run M_1 on w. If it rejects, then reject.
   2. If M_1 accepts the input, run M_2 on w. If M_2 rejects, then reject.
   3. If both M_1 and M_2 accept the input, then accept.
   M accepts w if both M_1 and M_2 accept it. If either M_1 or M_2 rejects w, then M rejects w.

c) complement: For any Turing-decidable language L, there is a Turing machine M that decides it. To prove that Turing-decidable languages are closed under complement, we construct a Turing machine $\overline{M}$ that decides $\overline{L}$.
   Let w be any string that contains symbols in language L. Then,
   1. Run M on w. If it accepts, reject.
   2. If M rejects, accept.
   If M accepts w, $\overline{M}$ rejects it. If M rejects w, $\overline{M}$ accepts it.

d) set difference: For any two Turing-decidable languages L_1 and L_2, there are Turing machines that decide them. Let these machines be M_1 and M_2. To prove that Turing-decidable languages are closed intersection, we construct a Turing machine M that decides $L_1 \cap \overline{L_2}$.

Let w be any string that contains symbols in languages L_1 and L_2. Then,
1. Run M_1 on w. If it rejects, then reject.
2. If M_1 accepts the input, run M_2 on w. If M_2 accepts, then reject.
3. If M_1 accepts w and M_2 rejects w, then accept.

M accepts w if M_1 accepts and M_2 rejects. If M_1 rejects w or if M_2 accepts it, M rejects.

Problem 4 Show that the Turing-recognizable languages are closed under concatenation

This problem requires providing constructions that take individual Turing machines and combines them into a new machine that *recognizes* the new language. Remember, this is about Turing-recognizable languages not just decidable so that there's a possibility of non-termination.

Problem 5 For each of the following Turing machine variants determine if the machine is more powerful, equivalent, or less powerful than a single-tape Turing machine. If less powerful describe the class of languages recognized by the machine. Explain your answers.

a) A Turing Machine that can only make moves to the right and never left.

b) A Turing Machine that can move right one space or move left two spaces.

c) A Turing Machine that never writes a space on the tape that already contains a symbol.