Homework 7 Solutions

1. Give an implementation-level description of a Turing machine that decides the language $B = \{ 0^n1^n2^n \mid n \geq 0 \}$.

Answer:

M = "On input string w:

1. Scan the input from left to right to make sure that it is a member of $0^*1^*2^*$, and *reject* if it isn't.
2. Return tape head to left-hand end of tape.
3. Repeat the following until no more 0s left on tape.
4. Replace the leftmost 0 with x.
5. Scan right until a 1 occurs. If there are no 1s, *reject.*
6. Replace the leftmost 1 with x.
7. Scan right until a 2 occurs. If there are no 2s, *reject.*
8. Replace the leftmost 2 with x.
9. Return tape head to left-hand end of tape, and go to stage 3.
10. If the tape contains any 1s or 2s, *reject.* Otherwise, *accept.*"

2. (a) Show that the class of decidable languages is closed under union.

Answer: For any two decidable languages L_1 and L_2, let M_1 and M_2, respectively be the TMs that decide them. We construct a TM M' that decides the union of L_1 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.* Otherwise, *reject.*

To see why M' decides $L_1 \cup L_2$, first consider $w \in L_1 \cup L_2$. Then w is in L_1 or in L_2 (or both). If $w \in L_1$, then M_1 accepts w, so M' will eventually accept w. Similarly, if $w \notin L_1$ but $w \in L_2$, then M_1 will reject w because M_1 is a decider (i.e., M_1 never loops), and M_2 will accept w, so M' will eventually accept w. On the other hand, if $w \notin L_1 \cup L_2$, then $w \notin L_1$ and $w \notin L_2$. Thus, both M_1 and M_2 reject w, so M' rejects $w \notin L_1 \cup L_2$. Hence, M' decides $L_1 \cup L_2$.

(b) Show that the class of Turing-recognizable languages is closed under union.

Answer: For any two Turing-recognizable languages L_1 and L_2, let M_1 and M_2, respectively, be TMs that recognize them. We construct a TM M' that recognizes the union $L_1 \cup L_2$:

M' = "On input string w:

1. Run M_1 and M_2 alternately on w, one step at a time. If either accepts, *accept*. If both halt and reject, *reject*.

To see why M' recognizes $L_1 \cup L_2$, first consider $w \in L_1 \cup L_2$. Then w is in L_1 or in L_2 (or both). If $w \in L_1$, then M_1 accepts w, so M' will eventually accept w. Similarly, if $w \in L_2$, then M_2 accepts w, so M' will eventually accept w. On the other hand, if $w \notin L_1 \cup L_2$, then $w \notin L_1$ and $w \notin L_2$. Thus, neither M_1 nor M_2 accepts w, so M' will also not accept w. Hence, M' recognizes $L_1 \cup L_2$. Note that if neither M_1 nor M_2 accepts w and one of them does so by looping, then M' will loop, but this is fine because we only needed M' to *recognize* and not *decide* $L_1 \cup L_2$.

3. In Theorem 3.21 we showed that a language is Turing-recognizable iff some enumerator enumerates it. Why didn't we construct the following simpler enumerator E' from an existing TM M for the forward direction of the proof? As before, $s_1, s_2, \ldots$ is a list of all strings in Σ^*, and the construct the following enumerator:

E' = "Ignore the input.

1. Repeat the following for $i = 1, 2, 3, \ldots$

2. Run M on s_i.

3. If it accepts, print out s_i."

Answer: The problem with the proof is that M on s_i might loop forever. If it loops forever, then E' doesn't print out s_i. More importantly, E' isn't going to move on to test the next string. Therefore, it won't be able to enumerate any other strings in L. For this reason, we need to simulate M on each of the strings for a fixed length of time so that no looping can occur.

4. A Turing machine with doubly infinite tape is similar to an ordinary Turing machine, but its tape is infinite to the left as well as to the right. The tape is initially filled with blanks except for the portion that contains the input. Computation is defined as usual except that the head never encounters an end to the tape as it moves leftward. Show that this type of Turing machine recognizes the class of Turing-recognizable languages.

Answer: A TM with doubly infinite tape can simulate an ordinary TM. It marks the left-hand end of the input to detect and prevent the head from moving off of that end.

To simulate the doubly infinite tape TM by an ordinary TM, we show how to simulate it with a 2-tape TM, which was already shown to be equivalent in power to an ordinary

TM. The first tape of the 2-tape TM is written with the input string, and the second tape is blank. We cut the tape of the doubly infinite tape TM into two parts, at the starting cell of the input string. The portion with the input string and all the blank spaces to its right appears on the first tape of the 2-tape TM. The portion to the left of the input string appears on the second tape, in reverse order.