

Sample Solutions for CMPS132 Mid-Term 1

1. (a) Obviously, there are many examples of such languages. One given in the textbook is NSA . (See Definition 10.4 on page 305 for its definition.)
- (b) Similarly, there are many examples of such languages. One given in the textbook is SA . (See Definition 10.4 on page 305 for its definition.)
- (c) A number of variations are possible. Let S be the start symbol. Then,

$$\begin{aligned} S &\rightarrow abcdS | \lambda \\ xy &\rightarrow yx \quad \text{where } x, y \in \{a, b, c, d\} \text{ and } x \neq y \end{aligned}$$

- (d) Let L be a language accepted by an LBA A . We will show that L is decidable by constructing a TM T that recognizes L .

Define T as follows. Given an input string x , T simulates A on x . If A accepts, then T accepts. If A crashes, then T rejects. To account for the possibility that A may loop forever, note that its space usage is bounded. This means that there are only finitely many LBA configurations that A could be in. (The number of head positions is $|x| + 2$, the number of states is $|Q|$, and the number of possible tape contents is $|\Gamma|^{|x|}$. Thus, the total number of LBA configurations is $(|x| + 2) \times |Q| \times |\Gamma|^{|x|}$.) Thus, if the simulation of A after that many steps does not accept x , we can conclude that A is in an infinite loop and reject x .

2. (a) We are given that the languages L_1 and L_2 are both Turing-acceptable. This means that there are TMs T_1 and T_2 that accept L_1 and L_2 , respectively. We will show that $L_1 \cup L_2$ is also Turing-acceptable by constructing a TM T that accepts the language.

Define T as follows. Given an input string x , T copies x onto tape 2 and then simulates T_1 on x (on tape 1) and T_2 on x (on tape 2) concurrently. (This means that T will alternate between one simulated step of T_1 and one simulated step of T_2 .) If either T_1 or T_2 accepts, then T accepts.

Alternate: Non-deterministically guess $i \in \{1, 2\}$ and run T_i on the input. If either T_1 or T_2 accepts, then there is a computation of the non-deterministic machine that accepts and vice versa.

- (b) We are given that the languages L_1 and L_2 are both Turing-acceptable. This means that there are TMs T_1 and T_2 that accept L_1 and L_2 , respectively. We will show that $L_1 \cap L_2$ is also Turing-acceptable by constructing a TM T that accepts the language.

Define T as follows. Given an input string x , T saves the input on tape 2. It then simulates T_1 on x . If T_1 accepts, then T simulates T_2 on x (on tape 2). If T_2 accepts, then T accepts.

3. Suppose the set of all infinite sequences of bits are countably infinite. This means that all such sequences can be listed as in s_1, s_2, \dots . Let $s_i[j]$ denote the j th bit in s_i . Consider the sequence of bits s^* that is defined as follows.

$$s^*[j] = \begin{cases} 1 & \text{if } s_j[j] = 0 \\ 0 & \text{if } s_j[j] = 1 \end{cases}$$

Because there are infinitely many s_i s, s^* is infinitely long. Thus, it must be the case that $s^* = s_k$ for some k . However, this cannot be the case. If $s^*[k] = 1$, then $s_k[k] = 0$ by definition of s^* . But this contradicts with the fact that $s_k = s^*$. Similarly, if $s^*[k] = 0$, then $s_k[k] = 1$ by definition of s^* . But this also contradicts with the fact that $s_k = s^*$.

4. The Halting Problem is, given a TM T and a string x , decide whether running T on x will halt or not.

The problem is known to be undecidable. This means that there is no TM that given a string of the form $(e(T), x)$, where $e(T)$ is the encoding of TM T and x is an input to T , decides whether running T on x will halt or not.

5. Suppose that it is decidable. This means that there is a TM T_H that accepts input strings of the form (T, x) if running T on x halts, and rejects all other input strings.

First, construct a TM T_D as follows. Given a TM T as input, T_D simulates T_H on (T, T) . Note that T_H always halts. If T_H accepts, then T_D loops for ever; otherwise, if T_H rejects, then T_D halts.

Next, consider running T_D on T_D . There are two possibilities. If it halts, it implies that T_H rejected (T_D, T_D) . This means that running T_D on T_D loops for ever, which contradicts with our assumption that it halted. If, on the other hand, it loops for ever, it implies that T_H accepted (T_D, T_D) . This means that running T_D on T_D halts, which contradicts with our assumption that it loops for ever.

Thus, the Halting Problem cannot be decidable.

6. (a) Erase the input tape.
 (b) Simulate M on the now empty input tape.
7. (a) *Trivial* properties of RE languages. These are properties that apply to all RE languages or none of the RE languages.
 (b) Both L_1 and L_2 decide non-trivial properties of RE languages. Thus, by Rice's Theorem, both are undecidable.
 (c) L_1 is RE. We show this by constructing a TM T that accepts L_1 . TM T works as follows: It nondeterministically guesses 481 distinct words and simulates M (the machine whose encoding was given as input) on each of the 481 words. If M accepts all of them, T accepts.
 (d) L_2 is not RE. To show that it is not, suppose that L_2 is RE. Since L_1 and L_2 are complements of each other, by Theorem 10.4 (which says that if both L and \bar{L} are RE, then both L and \bar{L} are recursive/decidable), both L_1 and L_2 must be decidable. This contradicts with our conclusion earlier (7(b)) that neither is decidable.