Problem Set 9 Solutions

Problem 1. As you did last week, classify each of the following languages as recursive, r.e. but not decidable, co-r.e. but not decidable, or neither r.e. nor co-r.e. Giving reductions where appropriate, prove your results.

1.1 \( A = \{ \langle M, k \rangle : M \text{ is a TM that accepts at least one string of length } k \} \).

r.e. An NTM can guess the string \( w \) of length \( k \) and verify that \( M \) accepts it.

However, \( A \) is not recursive. To see this, we show that \( A_{\text{TM}} \leq_m A \). This means that given a string \( \langle M, w \rangle \) we must construct \( \langle M', k \rangle \) by effective procedure such that \( M \) accepts \( w \) if and only if \( M' \) accepts some string of length \( k \). To do this, let \( k \) be arbitrary (eg, \( k = 1 \)) and let \( M' \) be the following machine: on input \( x \), \( M' \) ignores \( x \), runs \( M \) on \( w \), accepts if \( M \) does, and rejects if \( M \) rejects. Whenever \( M \) accepts \( w \) we will have that \( L(M') = \Sigma^* \), so \( M' \) will accept a string of length \( k \); while if \( M \) doesn't accept \( w \) then \( L(M') = \emptyset \) so \( M' \) will not accept any string of length \( k \).

1.2 \( B = \{ \langle M, k \rangle : M \text{ is a TM that runs forever on at least one string of length } k \} \).

co-r.e. The complement is the set of \( \langle M, k \rangle \) encodings such that \( M \) halts on every string of length \( k \). You can just try \( M \) on each string of length \( k \) and check that it halts on each of these finitely-many strings.

To show that \( B \) is not recursive, we show that \( \overline{A_{\text{TM}}} \leq_m B \). That is, given \( \langle M, w \rangle \), we must construct (by effective procedure) an \( \langle M', k \rangle \) such that \( M \) doesn't accept \( w \) if and only if \( M' \) diverges on some string of length \( k \). To carry out the mapping, let \( k = 0 \) and have machine \( M' \) on input \( x \) behave as follows: \( M' \) clears off \( x \), writes \( w \) on its input tape, and then behaves like \( M \), accepting if \( M \) accepts and looping if \( M \) rejects.

So if \( M \) does not accept \( w \), then \( M' \) diverges on some string of length 0 (namely, the empty string), while if \( M \) accepts \( w \) then \( M' \) accepts all strings of length 0 (namely, the empty string).

1.3 \( C = \{ \langle M, k \rangle : M \text{ is a TM that accepts a string of length } k \text{ and diverges on a string of length } k \} \).

Assume that the underlying alphabet has at least two characters.

neither. Let 0 and 1 name two characters in the underlying alphabet. First we show that \( A_{\text{TM}} \leq_m C \). This shows that \( C \) is not co-r.e. Given \( \langle M, w \rangle \) we must construct by effective procedure \( \langle M', k \rangle \) such that \( M \) accepts \( w \) if and only if \( M' \) accepts some string of length \( k \) and it diverges on some string of length \( k \). To do this, set \( k = 1 \) and have machine \( M' \) on input \( x \) if \( x \neq 0 \) then have \( M' \) diverge, while if \( x = 0 \) then let \( M' \) simulate \( M \) on input \( w \), accepting if \( M \) accepts \( w \) and looping if \( M \) rejects \( w \).

Now if \( M \) accepts \( w \) then \( M' \) accepts some string of length 1 (the string 0) and diverges on some string of length 1 (the string 1). If, instead, machine \( M \) does not accept \( w \), then \( M' \) diverges on all strings of length \( k = 1 \). Thus \( A_{\text{TM}} \leq_m C \).

Next we show that \( \overline{A_{\text{TM}}} \leq_m C \). This shows that \( C \) is not r.e. Given \( \langle M, w \rangle \), we must construct by effective procedure \( \langle M', k \rangle \) such that \( M \) fails to accept \( w \) if and only if \( M' \) accepts some string of length \( k \) and diverges on some string of length \( k \). To do this, set \( k = 1 \) and have machine \( M' \) on input \( x \) behave as follows: if \( x \neq 0 \) then \( M' \) accepts; otherwise, when \( x = 0 \), have \( M' \) simulate \( M \) on input \( w \), accepting if \( M \) accepts \( w \) and looping if \( M \) rejects \( w \). Now if \( M \) fails to accept \( w \) then \( M' \) diverges on some string of length 1 (the string 0) and \( M \) accepts some string of length 1 (the string 1). On the other hand, if \( M \) accepts \( w \) then \( M' \) accepts all strings, and so all strings of length 1. Thus \( \overline{A_{\text{TM}}} \leq_m C \).

1.4 \( D = \{ \langle M \rangle : M \text{ is a TM that accepts some palindrome} \} \).
r.e. An NTM can guess a palindrome \( w \) in \( L(M) \) and then verify that \( w \) is indeed a palindrome and \( w \in L(M) \). The language is undecidable by Rice's theorem, which gives us its classification. But to prove "from scratch" that \( D \) not co-r.e., we show that \( A_{TM} \leq_m D \). Given \( \langle M, w \rangle \) we must produce (in a Turing-computable way) a machine description \( \langle M' \rangle \) such that \( M \) accepts \( w \) if \( M' \) accepts some palindrome. Well, define \( M' \) (on input \( x \)) as follows: Runs \( M' \) on \( w \). If \( M \) accepts \( w \), then accept (the input \( x \) to \( M' \)). If \( M \) rejects \( w \), then reject (the input \( x \) to \( M' \)). Then if \( M' \) accepts \( w \) we will have that \( L(M') = \Sigma^* \), which certainly contains a palindrome. If \( M' \) doesn't accept \( w \) then \( L(M') = \emptyset \), so \( M' \) accepts no palindrome. We are done.

1.5 \( E = \{ \langle G_1, G_2 \rangle : G_1 \text{ and } G_2 \text{ are CFGs and } L(G_1) \oplus L(G_2) = \emptyset \}. \)

You may assume that \( L = \{ \langle G \rangle : G \text{ is a CFG and } L(G) = \Sigma^* \} \) is undecidable.

co-r.e. Two sets have empty symmetric difference iff they're the same; \( E \) is the set of all \( \langle G_1, G_2 \rangle \) where CFGs \( G_1 \) and \( G_2 \) denote the same language. An NTM can guess a string \( x \) in the symmetric difference of \( L(G_1) \) and \( L(G_2) \) and then verify this guess. To show that \( E \) is not decidable, we show that \( L \leq_m E \).

Given a CFG \( \langle G \rangle \), we map it to the pair \( \langle G, G_2 \rangle \) where \( G_2 \) is a fixed CFG the language of which is \( \Sigma^* \). Then \( L(G) \oplus L(G_2) = \emptyset \) iff \( L(G) = \Sigma^* \). The reduction is trivially computable.

1.6 \( F = \{ \langle M \rangle : \text{ \( M \) is a TM and } L(M) \text{ is recursive} \}. \)

neither. As the empty set \( \emptyset \) is recursive, Rice's theorem tells us that \( F \) is not r.e.. We must show that, in addition, it is not co-r.e.. To that end, let's reduce \( A_{TM} \leq_m F \). In particular, we must map an \( \langle M, w \rangle \) by a Turing-computable \( f \) to an \( M' \) such that \( M \) accepts \( w \) if \( L(M') \) is recursive. So let's have \( M' \), on input \( x \) behave as follows: first, run \( M \) on \( w \) for \( |x| \) steps. If \( M \) has accepted by this time, accept. Otherwise, parse \( x \) to a value \( (M'', w'') \) and run \( M'' \) on \( w'' \), accepting if \( M'' \) accepts and rejecting if \( M'' \) rejects. Now if \( M \) accepts \( w \) then \( L(M') \) is co-finite and therefore recursive; while if \( M \) doesn't accept \( w \) then \( L(M') = A_{TM} \), which is not recursive. The mapping \( f \) that takes \( \langle M, w \rangle \) to \( \langle M' \rangle \) is certainly Turing-computable, so we are done.

Problem 2 Prove or disprove each of the following claims.

2.1 If \( A \leq_m A \). True. The identify function provides the needed mapping \( f \).

2.2 If \( A \leq_m B \) and \( B \leq_m C \), then \( A \leq_m C \). True. Given \( f \) many-one reducing \( A \) to \( B \) and \( g \) many-one reducing \( B \) to \( C \), their composition, \( g \circ f \), many-one reduces \( A \) to \( B \).

2.3 If \( A \leq_m B \) then \( \overline{A} \leq_m \overline{B} \). True. If \( f \) many-one reduces \( A \) to \( B \) then \( x \in A \) iff \( f(x) \in B \), which means that \( f \) itself many-one reduces \( \overline{A} \) to \( \overline{B} \), as \( x \notin A \) iff \( f(x) \notin B \).

2.4 If \( A \) is recursive, then \( A \leq_m a^*b^* \). True. Because \( A \) is decidable we can construct a Turing-computable function \( f \) where \( f(x) = \varepsilon \) if \( x \in A \) and \( f(x) = ba \) if \( x \notin A \). This function comprises a many-one reduction from \( A \) to \( a^*b^* \).

2.5 If \( A \leq_m B \) then \( B \leq_m A \). False. For example, \( a^*b^* \leq_m A_{TM} \) but \( A_{TM} \nleq_m a^*b^* \).

2.6 If \( A \leq_m B \) and \( B \leq_m A \) then \( A = B \). False. Eg, \( \{0\} \leq_m \{1\} \) and \( \{1\} \leq_m \{0\} \), but that doesn't mean \( \{0\} = \{1\} \).

Problem 3. Let us say that a nonempty set \( B \) is countable if you can list (possibly with repetitions) its elements \( B = \{a_1, a_2, a_3, \ldots \} \); more formally, there is a surjective\(^1\) function \( f \) from \( \mathbb{N} \) to \( B \). We'll say that the empty set is also countable. A set is uncountable if it is not countable.

\(^1\)Recall that a function \( f : A \rightarrow B \) is surjective (or onto) if for every \( b \in B \) there is an \( a \in A \) such that \( f(a) = b \).
3.1 Prove that any subset $A$ of a countable set $B$ is countable.

If $A = \emptyset$ this is trivially true, so suppose $A \neq \emptyset$ and fix $a \in A$. Let $B$ be countable and let $f : \mathbb{N} \to B$ be a surjective function that demonstrates this. Define $g : \mathbb{N} \to A$ by $g(x) = f(x)$ if $x \in A$ and $g(x) = a$ otherwise. Then $g$ is onto $A$, since for every $x \in A$ there is an $i \in \mathbb{N}$ such that $f(i) = x$ and, for this $i$, we also have that $g(i) = x$.

3.2 Fix an alphabet $\Sigma$. Prove that there are countably many finite languages over $\Sigma$.

Every finite subset of $\Sigma^*$ can be specified by a string over $\Sigma^* \cup \{,\}$ (just list its elements). There are countably many strings over any alphabet: you can list the strings in lexicographic order.

3.3 Fix an alphabet $\Sigma$. Prove that there are uncountably many infinite languages over $\Sigma$.

Assume for contradiction that there is an enumeration $L_1, L_2, \ldots$ of the infinite languages over $\Sigma$. Let $w_1, w_2, \ldots$ be the lexicographic enumeration of all odd-length strings over $\Sigma$. Construct the language $D$ as follows: if $|x|$ is even, say that $x \in D$; if $|x|$ is odd, define $i$ so that $w_i = x$ and say that $x \in D$ iff $x \not\in L_i$. Then, for all $i$, we know that $D \neq L_i$ because $w_i \in D$ iff $w_i \not\in L_i$. Also, $D$ is infinite because it contains all even-length strings.