COM S 5310 Theory of Computing, Homework 2

Mobina Amrollahi

Due: October 8^{th}, 11:59pm on Gradescope.

Problem 1.

Prove that, for any two sets, A and B, the following two conditions are equivalent.

1. There is a function $f : A \xrightarrow{1-1} B$
2. $A = \phi$ or there is a function $g : B \xrightarrow{onto} A$

Conclude that a set A is countable (as we have defined in class) if and only if there is a function $f : A \xrightarrow{1-1} \mathbb{N}$.

Answer.

($\Rightarrow$) (1) $\implies$ (2): Suppose for A, B, there is a function $f : A \xrightarrow{\text{1-1}} B$. If the function f is empty, then $A = \varnothing$. Hence, suppose $A \neq \varnothing$, and so $\exists a_0 \in A$.

Consider $f^{-1}(b)$ as the pre-image of f, with the following definition:

$$f^{-1}(b) = \{a \mid a \in A,\ f(a) = b\}.$$

Since f is one-to-one, $\forall b \in B$, we have $|f^{-1}(b)| \leq 1$.

Define $g(y)$ for $y \in B$ as follows:

$$g(y) = \begin{cases} f^{-1}(y) & \text{if } |f^{-1}(y)| = 1, \\ a_0 & \text{if } |f^{-1}(y)| = 0. \end{cases}$$

To make sure that g is onto, consider the element $a_1 \in A$. Then $f(a_1) \in B$. Let $f(a_1) = b_1$. Then $g(b_1) = f^{-1}(b_1) = a_1$.

($\Leftarrow$) (2) $\implies$ (1): If $A = \varnothing$, we can consider the codomain of the function f to be empty. Then $f : \varnothing \xrightarrow{\text{1-1}} \varnothing$, and the one-to-one condition is vacuously true.

Now, consider $A \neq \varnothing$. Consider $g^{-1}(a)$ as the pre-image of g, with the following definition:

$$g^{-1}(a) = \{b \mid b \in B,\ g(b) = a\}.$$

Since g is onto, $\forall a \in A$, $g^{-1}(a) \neq \varnothing$. Additionally, for $a_1, a_2 \in A$, we know that $g^{-1}(a_1) \cap g^{-1}(a_2) = \varnothing$, since every element in B is mapped to only one element in A through the function g.

Now, for any $a \in A$, consider an element from $g^{-1}(a)$ denoted by b_a, where $b_a \in g^{-1}(a)$. Define

$$f(a) = b_a.$$

f is one-to-one, since for $a_1, a_2 \in A$, if $f(a_1) = f(a_2)$, then $b_{a_1} = b_{a_2}$. However, we have shown that the sets $g^{-1}(a)$ are disjoint; hence $a_1 = a_2$.

Consider $B = \mathbb{N}$. We know a set A is countable if $A = \varnothing$ or there is a function $g : \mathbb{N} \xrightarrow{\text{onto}} A$, which is equivalent to there being a function $f : A \xrightarrow{\text{1-1}} \mathbb{N}$.

Problem 2.

Let $\mathcal{C}_n$ be a set of languages $A \subseteq \{0,1\}^*$ for each $n \in N$. Prove: If each of the sets $\mathcal{C}_n$ is countable, then so is the set $\bigcup_{n=0}^{\infty} \mathcal{C}_n$.

Answer.

If $\mathcal{C}_n = \varnothing$ for all n, then

$$\bigcup_{n=0}^{\infty} \mathcal{C}_n = \varnothing,$$

and it is countable.

Now, suppose there exists n_0 such that $\mathcal{C}_{n_0} \neq \varnothing$. Then $\exists\, c_{n_0} \in \mathcal{C}_{n_0}$. To show that $\bigcup_{n=0}^{\infty} \mathcal{C}_n$ is countable, we must show that there exists a function

$$f : \mathbb{N} \xrightarrow[\text{onto}]{} \bigcup_{n=0}^{\infty} \mathcal{C}_n.$$

Since for every $i \in \mathbb{N}$, $\mathcal{C}_i$ is countable, it follows that $\exists\, f_i : \mathbb{N} \xrightarrow[\text{onto}]{} \mathcal{C}_i$.

Let p_i denote the i^{th} prime number ($p_i \in \mathbb{N}$), and define the sequence

$$\mathcal{G}_i = \{p_i^1, p_i^2, p_i^3, \ldots\}.$$

Define the function f by

$$f(p_i^j) = f_i(j),$$

and for all

$$n \in \mathbb{N} \setminus \bigcup_{i=0}^{\infty} \mathcal{G}_i, \qquad f(n) = c_{n_0}.$$

Then

$$f : \mathbb{N} \xrightarrow[\text{onto}]{} \bigcup_{n=0}^{\infty} \mathcal{C}_n$$

is onto.

To verify surjectivity, suppose $c \in \bigcup_{n=0}^{\infty} \mathcal{C}_n$. Then there exists i such that $c \in \mathcal{C}_i$, and there exists $j \in \mathbb{N}$, such that $f_i(j) = c$ (since f_i is onto). Therefore, $f(p_i^j) = c$.

Problem 3.

Design a Turing machine (as we defined in class) $M = (Q, \Gamma, \delta, s, H)$ that *decides* the language $\{0^n 1 0^{3n} | n \in \mathbb{N}\}$.

Answer.

Consider the Turing machine (TM) is a 5-tuple

$$M = (Q = \{s, q_1, \cdots, q_{10}, q_{11}, h\}, \Gamma = \{0, 1, \sqcup\}, \delta, s, H = \{h\})$$

where:

- $\delta : (Q \setminus H) \times \Gamma \to Q \times \Gamma \times \{-1, 0, 1\}$ is the transition function:

  $$\delta(q, a) = (\pi, b, d)$$

  where π is the new state of the machine, b is what happens to the cell that the tape head is pointing at time t, and $d = \{-1, 0, 1\}$ is the direction:

  - $1 \to$ one cell to the right
  - $0 \to$ hold still
  - $-1 \to$ one cell to the left

The following is the detailed description of the transition function δ: To help with checking the answers to this question, a Python code is attached. You can input your own strings, and it will show whether the machine halts and what the output on the tape would be: 0 for reject, 1 for accept.

q \ a	0	1	$\llcorner\!\lrcorner$
s	$(q_1, 0, 1)$	$(q_1, 1, 1)$	$(q_1, \llcorner\!\lrcorner, 1)$
q_1	$(q_1, 0, 1)$	$(q_2, 1, 1)$	$(q_{11}, \llcorner\!\lrcorner, 0)$
q_2	$(q_3, \llcorner\!\lrcorner, -1)$	$(q_2, 1, -1)$	$(q_9, \llcorner\!\lrcorner, 1)$
q_3	$(q_4, \llcorner\!\lrcorner, 1)$	$(q_4, 1, 1)$	$(q_{11}, \llcorner\!\lrcorner, 1)$
q_4	$(q_5, \llcorner\!\lrcorner, 1)$	$(q_4, 1, 1)$	$(q_{11}, \llcorner\!\lrcorner, 1)$
q_5	$(q_6, \llcorner\!\lrcorner, 1)$	$(q_5, 1, 1)$	$(q_{11}, \llcorner\!\lrcorner, 1)$
q_6	$(q_7, \llcorner\!\lrcorner, 1)$	$(q_6, 1, 1)$	$(q_{11}, \llcorner\!\lrcorner, 1)$
q_7	$(q_8, \llcorner\!\lrcorner, -1)$	$(q_7, 1, 1)$	$(q_{11}, \llcorner\!\lrcorner, 1)$
q_8	$(q_8, 0, -1)$	$(q_2, 1, 1)$	$(q_2, \llcorner\!\lrcorner, 1)$
q_9	$(q_{11}, 0, 1)$	$(q_9, 1, 1)$	$(q_{10}, \llcorner\!\lrcorner, -1)$
q_{10}	$(h, 1, 1)$	$(h, 1, 1)$	$(h, 1, 1)$
q_{11}	$(h, 0, 1)$	$(h, 0, 1)$	$(h, 0, 1)$
h	–	–	–

Table 1: Description of the Transition Function $\delta(q, a)$

Problem 4.

Prove that a language $A \subseteq \{0,1\}^*$ is c.e. if and only if there is a computable partial function $f :\subseteq \{0,1\}^* \to \{0,1\}^*$ such that $dom\ f = A$.

Answer.

($\Rightarrow$) $A \subseteq \{0,1\}^*$ is c.e. $\implies$ exists a computable partial function $f :\subseteq \{0,1\}^* \to \{0,1\}^*$ such that $dom\ f = A$.

Since A is c.e., there exists a Turing machine M such that $L(M) = A$. Equivalently, there exists an *enumerator* E that enumerates all strings of A [1].

Now construct a partial function f corresponding to the output behavior of the enumerator E. Specifically, define:

$$f(x) = \begin{cases} 1, & \text{if } x \text{ is eventually printed by the enumerator } E, \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

Since E enumerates exactly the elements of A, the domain of f consists precisely of all strings x that are printed by E, i.e., all $x \in A$. Thus, $\mathrm{dom}(f) = A$, and f is computable (because it can be implemented by simulating E and outputting 1 whenever E prints x).

($\Leftarrow$) There is a computable partial function $f :\subseteq \{0,1\}^* \to \{0,1\}^*$ such that $dom\ f = A$ $\implies$ $A \subseteq \{0,1\}^*$ is c.e.

Since f is a computable partial function, there exists a Turing machine M such that

$$\mathrm{dom}(f) = \{\, x \in \{0,1\}^* \mid M(x) \downarrow \},$$

i.e., M halts exactly on those strings in $\mathrm{dom}(f)$. Because $\mathrm{dom}(f) = A$, M halts on every string in A. Now construct the Turing machine M' that, on input x, simulates $M(x)$ and accepts whenever $M(x)$ halts. Formally, M' accepts every string $x \in A$ for which M halts.

Then the language recognized by M' is

$$L(M') = A,$$

since for any $x' \in L(M')$, M must have halted on x', and therefore $x' \in A$.

Hence, A is computably enumerable.

Problem 5.

Prove that, for every language $A \subseteq \{0,1\}^*$, the following conditions are equivalent.

1. A is c.e.
2. There is a computable partial function $f :\subseteq \mathbb{N} \to \{0,1\}^*$ such that *range* $f = A$.
3. $A = \phi$ or there is a computable function $f : \mathbb{N} \to \{0,1\}^*$ such that *range* $f = A$.
4. A is finite or there is a computable function $f : \mathbb{N} \xrightarrow{1-1} \{0,1\}^*$ such that *range* $f = A$.

Notes: Functions as in (3) are called *enumerations* of A and are the reason for the "c.e." terminology. Functions as in (4) are called *enumerations* of A *without repetition.*

Answer.

$(1 \Rightarrow 2)$ Consider the structure of the enumerator E described in [1], which enumerates the set $A \subseteq \{0,1\}^*$. Let $\{s_0, s_1, s_2, \dots\}$ be the *standard enumeration* of all binary strings in $\{0,1\}^*$. The enumerator E prints out some subset of these strings. Specifically, those that belong to A. That is, the i^{th} string printed by E may be s_j for some $j \geq i$.

Define a partial function

$$f :\subseteq \mathbb{N} \to \{0,1\}^*$$

such that $f(i) = s_j$ whenever the i^{th} string printed by E is s_j. Since A may be finite, $\mathrm{dom}(f)$ may be a finite subset of $\mathbb{N}$, f is defined only for those indices i corresponding to actual outputs of E.

Because the enumerator E eventually prints every element of A, each $a \in A$ appears as some printed string s_j. Hence, for each $a \in A$, there exists $i \in \mathrm{dom}(f)$ such that $f(i) = a$. Therefore,

$$\mathrm{range}(f) = A.$$

$(2 \Rightarrow 3)$ **Case 1:** $A = \varnothing$. In this case, the enumerator described in (2) will not print any strings. Hence, we can define

$$f : \varnothing \to \varnothing,$$

where the domain is a (possibly empty) subset of $\mathbb{N}$, and therefore

$$\mathrm{range}(f) = A = \varnothing.$$

Case 2: $A \neq \varnothing$. Then there exists some element $a_0 \in A$. With respect to the algorithm proposed for the enumerator E in [1], at each round i, if a string s_j with $j \leq i$ is printed, define

$$f(i) = s_j.$$

It might take several rounds before any string from A is printed, but since A is nonempty, there exists a first round i_0 at which some string $s_0 \in A$ is printed.

Assign all natural numbers up to i_0 to this first printed string:

$$f(i) = s_0, \quad \text{for } 0 \leq i \leq i_0.$$

Furthermore, if at any later round i no string is printed, set

$$f(i) = a_0.$$

Since every string printed by E belongs to A, we have:

$$\mathrm{range}(f) = A.$$

(3 ⇒ 4) **Case 1:** A is finite. Then A is either empty or not. If A is empty, we can follow exactly the same procedure described in (2).

If A is finite and nonempty, we can again follow the idea in (2): find the first string in A by looking at the first string printed by the enumerator. The process of assigning natural numbers to strings in A works as follows. At each round i, if the printer outputs a new string s_j, define

$$f(i) = s_j.$$

If no new string is printed at round i, set

$$f(i) = s_0,$$

where s_0 is the first string that appeared on the printer. If A is finite, then after some round i_0 no new strings will be printed. For all $i > i_0$, we keep assigning $f(i) = s_0$. This way, we make sure that $\text{range}(f) = A$.

Case 2: A is infinite. We have to make sure that when the enumerator E lists the strings in A, it does not print any string more than once. We can do this by building a new enumerator E' that simulates E as follows:

E'**:** Whenever E prints a string w:

(a) If w is not already on the list, print w and add it to the list.

(b) If w is already on the list, skip it.

Please note that at any point in time, the number of strings printed by E is finite. Hence, checking whether a newly printed string is already on the printer corresponding to E can be done in a finite amount of time.
Then, we define f by matching each natural number to the string E' prints at that position:

$$f(1) = \text{the first string printed by } E', \quad f(2) = \text{the second string printed by } E', \quad \text{and so on.}$$

This function is one-to-one, because every natural number is matched with a unique string that hasn't appeared before. Since E' prints every element of A exactly once:

$$\text{range}(f) = A.$$

(4 ⇒ 1) **Case 1:** A is finite. List all the strings in A. Construct a Turing machine M that, on input x, compares x with each string in A. If there is a match, M accepts; otherwise, M rejects.

Case 2: A is infinite. Then there is a computable function $f : \mathbb{N} \xrightarrow{1-1} \{0,1\}^*$ such that *range* $f = A$. Construct the following Turing machine M: Construct the following Turing machine M:

M**:** On input x, for $n = 0, 1, 2, \ldots$ (rounds):

(a) Dovetail over $i = 0, 1, \ldots, n$:

i. Compute $f(i)$ for one additional step

ii. If any computation halts with $f(i) = x$, then **accept**.

Problem 6.

Prove that a language $A \subseteq \{0,1\}^*$ is decidable if and only if A is finite or there is a computable function $f : \mathbb{N} \to \{0,1\}^*$ such that *range* $f = A$ and, for every $n \in \mathbb{N}$, $f(n)$ appears before $f(n+1)$ in the standard enumeration of $\{0,1\}^*$.

Answer.

(⇒) Suppose $A \subseteq \{0,1\}^*$ is decidable. Suppose A is infinite. Then there exists a Turing machine M such that (1) $L(M) = A$, and (2) M halts on every input $x \in \{0,1\}^*$.

Let $\{s_0, s_1, s_2, \ldots\}$ be the standard enumeration of all binary strings. We feed these strings to M in order and define a function f as follows:

$$f(n) = \text{the } n\text{th string (in standard order) that } M \text{ accepts.}$$

Because M halts on all inputs, we can effectively search for the next accepted string. Hence, f is computable, and by definition, $f(n)$ appears before $f(n+1)$ in the standard enumeration. And since the values of f are assigned only to those strings that M accepts, we have *range* $f = A$.

($\Leftarrow$) Suppose A is finite or there is a computable function $f : \mathbb{N} \to \{0,1\}^*$ such that *range* $f = A$ and, for every $n \in \mathbb{N}$, $f(n)$ appears before $f(n+1)$ in the standard enumeration of $\{0,1\}^*$. **Case 1:** A is finite. List all the strings in A. Construct a Turing machine M that, on input x, compares x with each string in A. If there is a match, M accepts; otherwise, M rejects. Since the list of strings in A is finite, this procedure halts on all inputs, and hence A is decidable.

Case 2: A is infinite. There exists a computable function $f : \mathbb{N} \to \{0,1\}^*$ such that range$(f) = A$ and $f(n)$ appears before $f(n+1)$ in the standard enumeration.

We define a Turing machine M that decides A as follows:

M**:** On input $x \in \{0,1\}^*$:

1. Initialize $n = 0$ and $S = \emptyset$.
2. Sequentially compute $f(n)$ for $n = 0, 1, 2, \ldots, N$, where $N = |x| + 1$.
   - If $f(n) = x$, **accept**.
3. **Reject**.

Since f is computable, we can generate $f(n)$ for any n. Because $f(n)$ appears before $f(n+1)$ in the standard enumeration, the comparison between x and $f(n)$ always proceeds in a consistent order.

The bound $N = |x|+1$ guarantees that after computing $f(0), \ldots, f(N)$, we will have compared x with every string that appears before or up to its length in the standard enumeration. Since A is infinite, the sequence $\{f(n)\}$ eventually produces a string longer than x, which is lexicographically greater than x. At this point, if x has not been encountered among the first N values, it cannot belong to A, and M rejects.

Therefore, A is decidable.

Problem 7.

Let Φ be the following statement. For every computable function $f : \{0,1\}^* \to \{0,1\}^*$, there is a computable function $g : \{0,1\}^* \to \{0,1\}^*$ such that, for all $x \in \{0,1\}^*$, $g(f(x)) = x$.

1. Prove that Φ is false by giving a counterexample.
2. Strengthen the hypothesis of Φ just enough to obtain a statement Φ' that is true.
3. Prove your statement Φ'.

Answer.

1. Consider the function

$$f : \{0,1\}^* \to \{0,1\}^*, \quad f(x) = 0$$

for all $x \in \{0,1\}^*$. This function is computable and corresponds to a Turing machine M that halts on every input and rejects (outputs 0).

For all $x_1, x_2 \in \{0,1\}^*$ with $x_1 \neq x_2$, we have

$$f(x_1) = f(x_2) = 0.$$

Hence, $g(f(x_1)) = g(f(x_2)) = g(0)$ takes the same value for every input, meaning g does not preserve the input string. Therefore, Φ is false.

2. Suppose
$$f : \{0,1\}^* \xrightarrow{\text{1-1}} \{0,1\}^*.$$
Define $g = f^{-1}$ as the inverse of f, where
$$\forall y \in \text{range}(f), \quad f^{-1}(y) = \{\, x \in \{0,1\}^* \mid f(x) = y \,\}.$$
To see that this pre-image always contains exactly one element, note that if we take any $y_0 \in \text{range}(f)$, then there must exist at least one x_0 such that $f(x_0) = y_0$. Moreover, if $x_1, x_2 \in f^{-1}(y_0)$, then $f(x_1) = f(x_2) = y_0$, and since f is one-to-one, it follows that $x_1 = x_2$. Therefore, $f^{-1}(y_0)$ contains exactly one element.

3. Let $y_0 \in \text{range}(f)$. Then there exists a unique $x_0 \in f^{-1}(y_0)$ such that $f(x_0) = y_0$. Therefore,
$$g(f(x_0)) = f^{-1}(f(x_0)) = f^{-1}(y_0) = x_0.$$
Hence, $g(f(x)) = x$ for all $x \in \text{domain}(f)$, and the strengthened statement Φ' is true.

Problem 8.

Prove that every infinite c.e. language $A \subseteq \{0,1\}^*$ has an infinite decidable subset.

Answer.

If $A \subseteq \{0,1\}^*$ is infinite, then by the result of Problem 5, there exists a computable one-to-one function
$$f : \mathbb{N} \xrightarrow{\text{1-1}} \{0,1\}^*$$
such that $\text{range}(f) = A$.
We will construct another computable function
$$f' : \mathbb{N} \xrightarrow{\text{1-1}} \{0,1\}^*$$
whose range defines a new language $A' = \text{range}(f')$, with $A' \subseteq A$, such that $f'(n)$ appears before $f'(n+1)$ in the standard enumeration of $\{0,1\}^*$. Then, by the result of Problem 6, any language whose elements can be enumerated in this way is decidable. Thus, showing that A' is infinite will complete the proof.
Since A is c.e., there exists a Turing machine M that enumerates A. We use M to define f' in the following:

Machine M':

1. Set $f'(0) = f(0)$.
2. For each $n \geq 1$, compare $f(n)$ with $f'(n-1)$ in the standard lexicographic order:
   - If $f(n)$ appears *after* $f'(n-1)$ in the standard enumeration, set $f'(n) = f(n)$.
   - Otherwise, continue scanning outputs $f(n+1), f(n+2), \ldots$ until a string $f(k)$ is found such that $f(k)$ is lexicographically greater than $f'(n-1)$, and set $f'(n) = f(k)$.

Because A is infinite, such a string $f(k)$ will always be found: for any n, let
$$M = \max\{\, |f'(1)|, |f'(2)|, \ldots, |f'(n-1)| \,\}.$$
There are only finitely many binary strings of length at most M, but since A is infinite, M cannot bound the lengths of all strings in A. Eventually, M' will reach a string of length greater than M, which is lexicographically larger than all previously selected strings. Hence, the process always terminates and defines $f'(n)$ for all n. Now, Each $f'(n)$ belongs to A, because every value is obtained from the enumeration f of A. Additionally, the range of f' is infinite, since we can always find a lexicographically larger element in A for the next step. Finally, The sequence $\{f'(n)\}$ is strictly increasing in the standard enumeration order. Thus, $A' = \text{range}(f') \subseteq A$ is infinite and can be enumerated by a computable function f' in lexicographic order. By the result of Problem 6, such a language is decidable.

Acknowledgment

I would like to acknowledge Ayesha Samreen. We specifically wrote a brief sketch of the transition function for Question 3, which I later asked ChatGPT to convert into a Python algorithm. This algorithm is attached to make the grading process easier. In writing the δ function, there was no involvement with ChatGPT or any other generative AI tools.
I would also like to acknowledge Abishek Jayan. We worked together on Questions 1, 4, 5, and 6, where we discussed the problems collaboratively but wrote our individual solutions.
I would like to thank Ryan Holt for discussing possible directions we could take for Question 6 as well as Jae Choi for their support in solving Question 8. I would also like to thank Dr. Lutz for clarifying which states of a Turing machine are halting, accepting, rejecting, or non-halting. The output of the transition function and the halting conditions are based on this discussion.
Last but not least, I would like to thank Saptarshi Biswas for extending office hours far beyond their scheduled time and for helping with almost all the problems, especially Question 5, where he mentioned the existence of an enumerator for any c.e. language, and Question 1, where he guided the approach to proving the existence of the proposed functions as left and right inverses.

References

[1] Michael Sipser. *Introduction to the Theory of Computation.* International Thomson Publishing, 1st edition, 1996.