Math 29: Computability Theory **Spring 2024**

PSET 4 — 04/26/2024

Prof. Miller ***Student: Amittai Siavava***

Problem 1.

Does Lemma 1 from the Noncomputability lecture hold if we remove the word "total"? That is, f partial computable if and only if its graph is a computable set? Justify your answer.

Lemma 1.1. The graph of a (partial) function f, $\mathbf{graph}(f)$, is the set $\{\langle n, k\rangle \mid f(n) = k\}$. If f is total, $\mathbf{graph}(f)$ is computable if and only if f is computable.

No, the lemma does not hold.

$\Longleftarrow$ ✓

Assume that $\mathbf{graph}(f)$ is computable. Consider the following turing machine:

TM 1: Compute $f(n)$

```
for k = 1, 2, 3, ... do
    if ⟨n, k⟩ ∈ graph(f) then
        output k
```

For $n \in \mathbf{dom}(f)$, the turing machine will eventually check if $\langle n, f(n)\rangle \in \mathbf{graph}(f)$ and output k. However, for $n \notin \mathbf{dom}(f)$, the turing machine will never halt. Thus, f is partial computable.

$\Longrightarrow$ ✗

However, f being partial computable *does not* imply that $\mathbf{graph}(f)$ is computable. Suppose we have a turing machine M simulating f. Then the approach for determining if $\langle n, k\rangle \in \mathbf{graph}(f)$ would be to run M on input n, obtain the output k_2, and compare if $k = k_2$. But given f is partial, M may not halt for some inputs (specifically $\mathbb{N} \setminus \mathbf{dom}(f)$, which is a nonempty set). Thus, we cannot compute $\chi_{\mathbf{graph}(f)}$, so $\mathbf{graph}(f)$ is not necessarily computable.

Problem 2.

Recall that W_e is $\mathbf{dom}(\varphi_e)$, and that X is c.e. if $X = W_e$ for some e. Show that it is equivalent to define the c.e. sets as those that are either finite or the range of a total, computable, injective function $f : \mathbb{N} \to \mathbb{N}$.

We can show the equivalence of the two definitions by proving that one is true if and only if the other is true.

X is c.e. $\implies$ X is finite or the range of a total, computable, injective function.

Proof. Let X be c.e. so that $X = W_e$ for some e.
Then there exists a turing machine $\mathscr{E}_X$ that enumerates X without repeating elements. Define $f : \mathbb{N} \to \mathbb{N}$ to be the function that pairs each input n with the nth element of X that is enumerated by $\mathscr{E}_X$.

- If X is infinite, then $\mathscr{E}_X$ will never halt or repeat an element. Each $n \in \mathbb{N}$ will eventually be paired with an element of X, so f is total, injective, and computable.
- On the contrary, if X is not total then there must exist some $n \in \mathbb{N}$ that is not paired with an element of X. This means that $\mathscr{E}_X$ will halt after a finite number of elements, specifically before the nth element is enumerated. Therefore, X must be finite.
- Similarly, if X is not injective, then there must exist some 2 elements $n_1, n_2 \in \mathbb{N}$ that are paired to the same $k \in X$. This is a contradiction to the fact that E_X enumerates *unique* elements of X.

□

$\Longleftarrow$

If X is finite or the range of a total, computable, injective function, then X is c.e.

Proof. For the two cases:

1. If X is finite, then $X = \{x_1, x_2, \ldots, x_n\}$ for some $n \in \mathbb{N}$. We can define a turing machine $\mathscr{E}_X$ that outputs $x_1, x_2, \ldots, x_n$ in order and then halts. Similarly, given an input x, we can check if x occurs in the list $x_1, x_2, \ldots, x_n$ in finite time and halt if it does. Therefore, X is c.e.
2. If X is infinite and it is the range of a total, computable, injective function $f : \mathbb{N} \to \mathbb{N}$. Then $X = \{f(1), f(2), \ldots\}$. Since f is computable, $f = \varphi_e$ for some e. We can therefore enumerate X by running φ_e on $1, 2, 3, \ldots$ and outputting the corresponding values of $f(1), f(2), f(3), \ldots$ that are generated by φ_e.

□

Problem 3.

Prove that a c.e. set is computable *if and only if* it is the range of an increasing, total computable function.

$\Longrightarrow$

Suppose X is c.e. and computable. Then $X = W_e$ for some e. Since X is computable, we can specify a turing machine to print the elements of X in increasing order:

TM 2: Enumerate X in increasing order

```
for i = 0, 1, 2, ... do
    if χ_X(i) = 1 then
        print i
```

Define a function f that, given input n, outputs the nth element listed in the increasing-order enumeration of X. Then f is an increasing, total computable function whose range is X.

$\Longleftarrow$

Suppose X is the range of an increasing, total, computable function $f : \mathbb{N} \to \mathbb{N}$. We shall show that X is computable.

Since f is total and increasing, we have

$$\forall\, n_1, n_2 \in \mathbb{N},\ n_1 \leq n_2 \implies f(n_1) \leq f(n_2).$$

Furthermore, since f is computable, $f = \varphi_e$ for some e.

We can compute the characteristic function of X, χ_X, as follows:

TM 3: Compute $\chi_X(n)$

```
for x = 0, 1, 2, ... do
    if f(x) = n then
        output 1
    else if f(x) > n then
        output 0
```

Since f is total and increasing, the turing machine will eventually either reach an x such that $f(x) = n$ and output 1, or encounter a value of x such that $f(x) > n$ and output 0. Therefore, X is computable.

Problem 4.

Prove that K (the halting set) is ***not*** an index set.

1. $K = \{e \mid \varphi_e(e)\downarrow\}$.

2. An index set is a set X such that, for all e and k, if $\varphi_e = \varphi_k$ then $e \in X$ if and only if $k \in X$.

To show that K is not an index set, we shall find a code $e \in K$ and show that $k \notin K$ for some $\varphi_k = \varphi_e$.

Define a function f that converges only on its own code and diverges for all other $n \in \mathbb{N}$. That is, if e is the code of f, then

$$f(n) = \begin{cases} 1 & \text{if } n = e \\ \uparrow & \text{otherwise.} \end{cases}$$

Note that $e \in K$ since $\varphi_e(e)\downarrow$.
By the ***Padding Lemma***, for any e, there are infinitely many $k \neq e$ such that $\varphi_e = \varphi_k$. Pick one such k. What happens when we run $\varphi_k(k)$? Since $k \neq e$, $\varphi_e(k)\uparrow$. Therefore, $\varphi_k = \varphi_e$ since $\varphi_k(k)\uparrow$. Therefore, $k \notin K$.

This means that K must not be an index set, since the condition that

$$\varphi_e = \varphi_k \implies (e \in K \leftrightarrow k \in K)$$

does not hold for K.

Problem 5.

Show that if P is productive then P contains an infinite c.e. set.

Definition 5.1. A set P is productive if it has a productive function — a (partial) computable function ψ such that, whenever $W_e \subseteq P$, $\psi(e)\downarrow$ and $\psi(e) \in P \setminus W_e$. That is, a productive function is able to produce a witness to the fact that $P \neq W_e$ whenever $W_e \subseteq P$. Then it is immediate that productive sets are not c.e., so finding a c.e. set whose complement is productive will necessarily be a noncomputable c.e. set.

Let P be a productive function, with ψ as its productive function. We shall enumerate an infinite set $Y = \{y_0, y_1, y_2, \dots\} \subseteq P$ as follows:

1. Take e_0 to be the smallest index with $W_{e_0} = \varnothing \subseteq P$. Then $\psi(e_0) \downarrow= y$ for some $y \in P \setminus \varnothing = P$. Set $y_0 = y$.
2. Inductively, for $n \geq 1$, find a function ψ_{e_n} select e_n to be the smallest index such that $W_{e_n} = \{y_0, y_1, \dots, y_{n-1}\} \subseteq Y$.
   Then $\psi(e_n) \downarrow= y$ for some $y \in P \setminus \{y_0, y_1, \dots, y_{n-1}\}$. Thus, $y \neq y_i$ for any $i < n$.
   Set $y_n = y$.

To show that Y is c.e., we need to show that $Y = W_e$ for some e. Consider the following turing machine $\mathscr{E}_Y$ that enumerates Y:

TM **4:** Compute $f(y) : Y \to \mathbb{N}$

Initialize $Y \leftarrow \varnothing$
for $n = 0, 1, 2, \dots$ **do**
  Compute $y_n = \psi(e_n)$
  if $y_n = y$ **then**
    output 1

The turing machine halts and outputs 1 only when the input y is a member of Y. If not, it will continue to search loop, ad infinitum.

Let e be the code of the turing machine. Then $Y = W_\varepsilon$, so Y is c.e.