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, $\textbf{graph}\,(f)$, is the set $\{\langle n, k\rangle \mid f(n) = k\}$. If f is total, $\textbf{graph}\,(f)$ is computable if and only if f is computable.

No, the lemma does not hold.

$\Longleftarrow$ ✓

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

TM **1:** Compute $f(n)$

for $k = 1, 2, 3, \ldots$ **do**
  if $\langle n, k\rangle \in \textbf{graph}\,(f)$ **then**
    output k

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

$\Longrightarrow$ ✗

However, f being partial computable *does not* imply that $\textbf{graph}\,(f)$ is computable. Suppose we have a turing machine M simulating f. Then the approach for determining if $\langle n, k\rangle \in \textbf{graph}\,(f)$ 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 $\mathbb{N} \setminus \textbf{dom}\,(f)$, which is a nonempty set). Thus, we cannot compute $\chi_{\textbf{graph}(f)}$, so $\textbf{graph}\,(f)$ is not necessarily computable.

Problem 2.

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 X is c.e. if and only if X is finite or the range of a total, computable, injective function.

$\Longrightarrow$

Proof. Let X be c.e. so that $X = W_e$ for some e.

If X is finite, then the condition is satisfied.

Suppose X is infinite. We shall construct a total, computable, injective function $f : \mathbb{N} \to \mathbb{N}$ whose range is X. Precisely, we construct f such that, given input n, f returns the $n+1$th member of X in the sequence $0, 1, 2, 3, \ldots$

TM 2: f: enumerate X

On input n:
$S \leftarrow \varnothing$
for $i = 0, 1, 2, 3, \ldots$ **do**
    if $\varphi_e(i)\downarrow$ **then**
        $S \leftarrow S \cup \{\varphi_e(i)\}$
        if $|S| = n + 1$ **then**
            output $\varphi_e(i)$

$\square$

$\Longleftarrow$

Proof. For the two cases:

Suppose X is finite, then $X = \{x_1, x_2, \ldots, x_n\}$ for some $n \in \mathbb{N}$. Define $g : \mathbb{N} \to \mathbb{N}$ by:

$$g(i) = \begin{cases} k & \text{if } i = x_k \text{ for some } k < n \text{ and } x_k \in X \\ \uparrow & \text{otherwise.} \end{cases}$$

Let e be the code of the turing machine that computes g. Then $\varphi_e(i) \downarrow$ *if and only if* $i \in X$. Therefore, $\mathbf{dom}\,(\varphi_e) = W_e = X$, so X is c.e.

Suppose X is infinite and it is the range of a total, computable, injective function $f : \mathbb{N} \to \mathbb{N}$.

We show that f is the domain of some function $g : \mathbb{N} \to \mathbb{N}$. Define g as follows:

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

TM 3: $g : \mathbb{N} \to \mathbb{N}$

```
On input n:
for i = 0, 1, 2, 3, ... do
    if f(i) ↓= n then
        output i
```

Let e be the code of the turing machine that computes g as outlined above. Then $\varphi_e(n) \downarrow$ *if and only if* $n = f(i)$ for some i, so $n \in \mathbf{range}\,(f) = X$. Therefore, $\mathbf{range}\,(f) = \mathbf{dom}\,(\varphi_e) = W_e = X$, and X is c.e. $\square$

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 4: Enumerate X in increasing order

for $i = 0, 1, 2, \ldots$ **do**
    if $\chi_X(i) = 1$ **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 5: Compute $\chi_X(n)$

for $x = 0, 1, 2, \ldots$ **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.

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 the machine that computes f, then

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

We can do this because of he recursion theorem. 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 $k \neq e$, $\varphi_e(k)\uparrow$ since $\varphi_e(k)\uparrow$. Therefore, $k \notin K$.

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

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

does not hold.

Problem 5.

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 $Y = \{y_0, y_1, y_2, \ldots\} \subseteq P$ as follows:

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

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 **6:** $\varphi_e : Y \to \mathbb{N}$

On input n:
Initialize $Y \leftarrow \varnothing$
for $i = 0, 1, 2, \ldots$ **do**
    Select e_i as above
    Compute $y_i = \psi(e_i)$
    if $y_i = n$ **then**
        output 1

On input n, the turing machine halts and outputs 1 if $n \in Y$. If $n \notin Y$, the turing machine will continue to loop, ad infinitum.

Therefore, $\mathbf{dom}\,(\varphi_e) = Y$, so Y is c.e.