Mathematical Logic Homework 02

Qiu Yihang

Sept.28 - 30, 2022

1 The Set of Real Numbers in $(-1, 0]$ is Uncountable

Proof. Assume R is countable.

Then we can find a listing of R without repetitions: $a_0, a_1, a_2, ...a_n, ...$ (which could be finite).

Meanwhile, any real number in the interval $(-1, 0]$ can be rewritten as a binary decimal, i.e. $-\left(\overline{0.h_0h_1h_2...h_m...}\right)_2$, where $h_i \in \{0, 1\}, i \in \mathbb{N}$.

For infinite binary decimal, its fraction part is obvious a sequence only containing 0s and 1s. For finite binary decimal, we can convert it to an infinite sequence by adding infinite 0s at its end.

Therefore, any number a_i can be rewritten as $-\left(\overline{0.a_{i0}a_{i1}a_{i2}...a_{ik}...}\right)_2$, where $a_{ij} \in \{0, 1\}, j \in \mathbb{N}$.

Then we construct a real number x by diagnal argument as follows. Let $x = -\left(\overline{0.x_0x_1x_2...x_k...}\right)_2$, where $x_i = 1 - a_{ii} \in \{0, 1\}.i \in \mathbb{N}$.

To further clarify, an example is given as below.

$$\begin{array}{rcccccccccc}
a_0 = & -0. & \mathbf{0} & 1 & 0 & 1 & 1 & 0 & 0 & 0 & ... \\
a_1 = & -0. & 1 & \mathbf{1} & 1 & 0 & 0 & 0 & 1 & 1 & ... \\
a_2 = & -0. & 0 & 0 & \mathbf{1} & 1 & 1 & 1 & 1 & 1 & ... \\
a_3 = & -0. & 1 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 & ... \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x = & -0. & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & 1 & 0 & 0 & 1 & ...
\end{array}$$

Then we know $x \neq a_n$ for any a_n in the listing. (Since the $(n+1)$-th bits are different).

Therefore, $x \notin (-1, 0]$. Meanwhile, by the construction of x, we know $x > -1$ and $x \leq 0$, i.e. $x \in (-1, 0]$. **Contradiction.**

Thus, R is uncountable. ■

2 An Algorithm for Determining Membership in $\mathbb{P}$

Solution. We design the following algorithm.

Algorithm 1: Algorithm for Determining Membership in Set of Prime Numbers

```
Algo. Prime Number Discriminator
begin
    on Input n;
    if n = 0 or n = 1 then Output: "NO";
    for i = 2 → (n − 1) do
        if i|n then    Output: "NO";
    end
    Output: "YES";
end
```

Obvious the algorithm will halt within finite steps. Now we prove its correctness.

If the input n is a prime number, we know for any $k \in \mathbb{N}, k \geq 2, k \neq n$, $k \nmid n$. Thus, the algorithm will output "YES".

If the input n is 0 or 1, neither of which is not a prime number, the algorithm outputs "NO".

If the input n is not a prime number, we know exists $k \in \mathbb{N}, 2 \leq k \leq (n-1)$ s.t. $k \mid n$. Then the algorithm will output "NO" at $i = k$.

Thus, the algorithm can correctly determine the membership of the input in $\mathbb{P}$.

In fact, $i = 2 \to \lfloor \sqrt{n} \rfloor$ is enough for the for-loop, considering $k \mid n \Leftrightarrow \frac{n}{k} \Big| n$.

Therefore, the **Algorithm 1** is an algorithm for determining membership in $\mathbb{P}$. ■

3 An Algorithm For Enumerating Prime Numbers

Solution. Based on the **Algorithm 1**, we design the algorithm as follows.

Algorithm 2: Algorithm for Enumerating Prime Numbers

```
Algo. Prime Number Enumerator
begin
    for n = 0, 1, 2, ... do
        Run Prime Number Discriminator on n;
        if the result is "YES" then    print: n;
    end
end
```

Let the numbers listed by the algorithm above be $a_0, a_1, a_2, ...a_n, ...$

For any $i \in \mathbb{N}$, a_i is a prime number, which is guaranteed by the correctness of **Algorithm 1**.

For any prime number p, let p be the k-th smallest prime number. Then exists k s.t. $p = a_k$.

Thus, **Algorithm 2** is an algorithm for enumerating prime numbers. ■

4 Range of Total Function f is Effectively Decidable

Proof. Since f is a total function, we know $\texttt{domain}(f) = \mathbb{N}$, i.e. for any $n \in \mathtt{N}, f(n)$ is defined.

Meanwhile, f is effectively computable.

Then exists an algorithm $\mathcal{A}$ s.t. on input n, $\mathcal{A}$ prints $f(n)$ within finite steps.

Considering f is strictly increasing, we know $x \in \texttt{range}(f) \iff$ exists $n \in \mathbb{N}$ s.t. $f(n) = x$ while $x \notin \texttt{range}(f) \iff$ exists $n \in \mathbb{N}$ s.t. $f(n) < x, f(n+1) > x$.

Thus, we can construct an algorithm for determining membership in $\texttt{range}(f)$ as follows.

Algorithm 3: Algorithm for Determining Membership in $\texttt{range}(f)$

Algo. *Strictly Increasing Total Function Range Discriminator*
begin
  on **Input** n;
  Run $\mathcal{A}$ on 0;   `// Since` f `is total,` $\mathcal{A}$ `will terminate in finite steps.`
  if *the result* $= n$ **then** **Output:** "YES";
  if *the result* $> n$ **then** **Output:** "NO";
  for $i = 1, 2, ...$ **do**
    Run $\mathcal{A}$ on i;   `// Since` f `is total,` $\mathcal{A}$ `will terminate in finite steps.`
    if *the result* $= n$ **then** **Output:** "YES";
    if *the result* $> n$ **then** **Output:** "NO";
  end
end

Now we prove the algorithm above is one for determining membership in $\texttt{range}(f)$.

First we prove that for any input $n \in \mathbb{N}$, the algorithm will halt within finite steps.

Each time we run $\mathcal{A}$, it will terminate in finite steps.

Meanwhile, for input $n \in \mathbb{N}$, we run $\mathcal{A}$ for at most n times. Otherwise, exists $x \in \mathbb{N}$ s.t. $f(x) \geq f(x+1)$, which contradicts to that f is strictly increasing.

Thus, on any input $n \in \mathbb{N}$, the algorithm will terminate within finite steps. □

Then we prove the correctness of the algorithm.

When the algorithm returns "YES", either $f(0) = \mathcal{A}(0) = n \in \texttt{range}(f)$ or exists a number $i \in \mathbb{N}$ s.t. $f(i) = \mathcal{A}(i) = n \in \texttt{range}(f)$. Correct.

When the algorithm returns "NO", there exists two cases.

CASE 01. $f(0) > n$. Then for any $i \in \mathbb{N}$, $f(i) > f(0) > n$. Thus, $n \notin \texttt{range}(f)$.

CASE 02. f terminates at $i = k$. Then we know for $i < k$, $f(i) < n$ while $f(k) > n$. Thus, exists $x = k - 1 \in \mathbb{N}$ s.t. $f(x) < k < f(x+1)$, i.e. $n \notin \texttt{range}(f)$.

In conclusion, **Algorithm 3** gives the correct result. □

Therefore, **Algorithm 3** is an algorithm for determining membership in $\texttt{range}(f)$. ■

5 A is Effectively Decidable

Proof. A is effectively enumerable $\Longrightarrow$ exists algorithm $\mathcal{A}$ for enumerating members in A.

$\mathbb{N} \setminus A$ is effectively enumerable $\Longrightarrow$ exists algorithm $\mathcal{B}$ for enumerating members in $\mathbb{N} \setminus A$.

Let the output of $\mathcal{A}$ and $\mathcal{B}$ be $a_0, a_1, ...a_n, ...$ and $b_0, b_1, ...b_n, ...$ respectively.

Then we know

- $a \in A \Rightarrow a = a_n$ for some $n \in \mathbb{N}$, i.e. a will show up in the output of $\mathcal{A}$ after finite steps.
- $a \in \mathbb{N} \setminus A \Rightarrow a = b_n$ for some $n \in \mathbb{N}$, i.e. a will show up in the output of $\mathcal{B}$ after finite steps.

Then we con construct an algorithm $\mathcal{C}$ for determining membership in A as follows.

Algorithm 4: Algorithm for Determining Membership in A

```
Algo. Algorithm C
begin
    on Input n;
    for i = 1, 2, ... do
        Run A until it prints the i-th number;
        if the i-th output = n then    Output: "YES";
        Run B until it prints the i-th number;
        if the i-th output = n then    Output: "NO";
    end
end
```

Now we prove $\mathcal{C}$ is an algorithm for determining membership in A.

When $n \in A$, we know exists k s.t. $a_k = n$. Thus, $\mathcal{C}$ will terminate when $i = k$ with "YES", i.e. $\mathcal{C}$ returns "YES" within finite steps.

When $n \in \mathbb{N} \setminus A$, we know exists k s.t. $b_k = n$. Thus, $\mathcal{C}$ will terminate when $i = k$ with "NO", i.e. $\mathcal{C}$ returns "NO" within finite steps.

Thus, $\mathcal{C}$ is an algorithm for determining membership in A.

Therefore, A is effectively deicidable. ■

6 P is Effectively enumerable

Solution. $P = \{n \in \mathbb{N} \mid \forall x < n, x \in R\}$.

Since R is effectively enumerable, there exists algorithm $\mathcal{A}$ for enumerating members of R.

Then we can construct an algorithm $\mathcal{A}'$ for listing members in P as follows.

Algorithm 5: Algorithm for Enumerating Members in P

```
Algo. Algorithm 𝒜'
begin
    S ← ∅;
    print: 0;
    for i = 1, 2, 3, ... do
        Continue running 𝒜 until it prints the i-th number a_i;
        S ← S ∪ {a_i};
        for j = 0, 1, 2, ... do
            if j ∉ S then break;
            print: j + 1;
        end
    end
end
```

Now we prove $\mathcal{A}'$ is an algorithm for enumerating members in P.

Let the output of $\mathcal{A}'$ be $p_0, p_1, p_2, ...p_n, ...$. Obvious $p_0 = 0$.

When $n = 0$, $p_n = 0$. $0 \in P$.

For any $n \in \mathbb{N}, n \geq 1$, by the process of algorithm, we know $0, 1, ..., (p_n - 1) \in S$. Meanwhile, $S \subset R$. Thus, $p_n \in P$.

Suppose $x \in R$ will appear in the output of $\mathcal{A}$ after $\texttt{num}(x)$ steps.

Then for any $a \in P$, we know for any $x \in \mathbb{N}, x < a \Rightarrow x \in R$, i.e. x will appear in the output of $\mathcal{A}$ within finite steps. Thus, a will appear in the output of $\mathcal{A}'$ within $\sum_{k\in\mathbb{N},k<a} \texttt{num}(k)$ steps, i.e. within finite steps.

Therefore, $\mathcal{A}'$ is an algorithm for listing the members in P.

Thus, P is effectively enumerable. ■