COMS 3003A
HW 5

Dmitry Shkatov

22 March, 2024

Watching:
Videos for week 5 (see Moodle).

(1) **In this question, we're looking at bijections, injections, and surjections. If $f\colon X \to Y$ and $g\colon Y \to Z$, then the *composition* of f and g is the function $g \circ f\colon X \to Z$ defined by**

$$(g \circ f)(x) \;=\; g(f(x)) \qquad \textbf{whenever } x \in X.$$

(a) **Prove that a composition of bijections is a bijection.**

Suppose that f and g are bijections. Hence f and g are both surjections and injections.

First, we show that $g \circ f$ is an injection. We need to show that $x_1 \neq x_2$ implies $(g \circ f)(x_1) \neq (g \circ f)(x_2)$, for every $x_1, x_2 \in X$. Suppose that $x_1, x_2 \in X$ and $x_1 \neq x_2$. Since f is an injection, $f(x_1) \neq f(x_2)$. Since g is an injection, $g(f(x_1)) \neq g(f(x_2))$, i.e., $(g \circ f)(x_1) \neq (g \circ f)(x_2)$.

Next, we show that $g \circ f$ is a surjection. We need to show that, for every $z \in Z$, there exists $x \in X$ such that $(g \circ f)(x) = z$. Pick an arbitrary $z \in Z$. Since g is a surjection, there exists $y \in Y$ such that $g(y) = z$. Since f is a surjection and $y \in Y$, there exists $x \in X$ such that $f(x) = y$. Then, $(g \circ f)(x) = g(f(x)) = g(y) = z$. Since z is an arbitrary element of Z, the statement follows.

(b) **Prove that a composition of injections is an injection.**

The proof is a verbatim repetition of the first half of the proof of (a).

(c) **What can we say about surjections?**

The proof is a verbatim repetition of the second half of the proof of (a).

(2) **Prove that the set of prime numbers is countable.**

Proof #1: Since every listable set is countable, it is enough to produce an infinite list of prime numbers. Take a list of all natural number and remove from it all the numbers that are not prime. This gives us a required list.

Proof #2: Every prime number can be represented in the finite alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Since the set of strings of a finite alphabet is countable, the set of primes is countable.

(3) **Prove that the set of points on the real plane that have integer coordinates is countable.**

Proof #1: Notice that we can traverse the real plane so that we pass through every point with integer coordinates—for example, by visiting points in this order: $(0,0)$, $(0,1)$, $(1,1)$, $(1,0)$, $(1,-1)$, $(0,-1)$, $(-1,-1)$, $(-1,0)$, $(-1,1)$, $(-1,2)$, $(0,2)$, $(1,2)$, etc. Thus, the set of points with integer coordinates is listable; hence, it is countable.

Proof #2: Every point of the real plane with integer coordinates can be represented in the finite alphabet $\Sigma = \{0,1,2,3,4,5,6,7,8,9,),(,-,,\}$. Since the set of strings of a finite alphabet is countable, the set of points of the real plane with integer coordinates is countable.

(4) **Prove that the set of Turing machines is countable.**

As we have seen, every Turing machine can be represented as a binary string. We have also seen that the set of binary strings is countable. Hence, the set of Turing machines is countable.

(5) **We have shown in lecture that, for every set X, the set of Boolean-valued functions with domain X is strictly larger than X. Prove, using this fact and (4), that there exist undecidable decision problems.**

A decision problem is a function from $\{0,1\}^*$ into $\{0,1\}$. By our result, the set P of all decision problems is strictly larger than the set $\{0,1\}^*$ itself, i.e. $|\{0,1\}^*| < |P|$. On the other hand, the set of Turing machines that are deciders has the same cardinality as $\{0,1\}^*$, since both are countable sets; i.e., $|D| = |\{0,1\}^*|$. Therefore,

$$|D| < |P|. \qquad (*)$$

On the other hand, if every decision problem were solvable by a Turing machine, then the set of deciders would have to be at least as large as the set of problems. To see this, define a function $f\colon D \to P$ so that, if $M \in D$, then $f(M)$ is the problem that M solves. If every problem is solvable by some machine, then f is a surjection, which means that $|P| \leqslant |D|$. This, however, contradicts $(*)$. Therefore, it's impossible for every decision problem to be solvable by a Turing machine.

(6) **Prove that the set of all subsets of a set X is strictly larger than X itself.**

Let 2^X be the set of all Boolean-valued functions with domain X and let $\mathcal{P}(X)$ be the set of all subsets of X. We know that $|X| < |2^X|$. To prove that $|X| < |\mathcal{P}(X)|$, it would be enough to show that $|\mathcal{P}(X)| = |2^X|$. For that, we need a bijection between $\mathcal{P}(X)$ and 2^X. Such a bijection does exist: define : $\mathcal{P}(X) \to 2^X$ so that, if $Y \subseteq X$, then $f(Y)$ is the function $g \in 2^X$ such that $g(x) = 1$ if, and only if, $x \in Y$.

To show that f is a bijection, we need to show that $Y_1 \neq Y_2$ implies $f(Y_1) \neq f(Y_2)$ and that, for every $g \in 2^X$, there exists $Y \subseteq X$ such that $f(Y) = g$. First, suppose that $Y_1 \neq Y_2$. Then there exists $x \in X$ such that $x \in Y_1$ and $x \notin Y_2$ (or vice versa, in which case the argument is symmetrical). Then, by definition of g, both $f(Y_1)(x) = 1$ and $f(Y_2)(x) = 0$, which impliest that $f(Y_1) \neq f(Y_2)$. Second, suppose that $g \in 2^X$. Define $Y_g = \{x \in X \mid g(x) = 1\}$. Then, $f(Y_g) = g$, by definition of f. Thus, f is a bijection, and hence $|\mathcal{P}(X)| = |2^X| > |X|$.

(7) **As we have seen, every Turing machine can be represented as a binary string. Thus, every Turing machine can be input to a Turing machine with a binary input alphabet. Which questions do you think we might like to ask about Turing machines that we might want to be answered by Turing machines?**

Some examples:

- Does a TM halt on all inputs?
- Does a TM halt on at least one input?
- Does a TM halt on the empty string?
- Does a TM accept any input?

(8) What kind of questions we might like to ask Turing machines about themselves? I.e., think of a meaningful question we might pose to a Turing machine M about M.

Some examples:

- Does you halt on all inputs?
- Does you halt on at least one input?
- Does you halt on the empty string?
- Does you accept any input?
- Do you accept yourself?