1 Countability: True or False

(a) The set of all irrational numbers $\mathbb{R}\backslash\mathbb{Q}$ (i.e. real numbers that are not rational) is uncountable.

(b) The set of integers x that solve the equation $3x \equiv 2 \pmod{10}$ is countably infinite.

(c) The set of real solutions for the equation $x+y=1$ is countable.

For any two functions $f : Y \to Z$ and $g : X \to Y$, let their composition $f \circ g : X \to Z$ be given by $f \circ g = f(g(x))$ for all $x \in X$. Determine if the following statements are true or false.

(d) f and g are injective (one-to-one) $\implies$ $f \circ g$ is injective (one-to-one).

(e) f is surjective (onto) $\implies$ $f \circ g$ is surjective (onto).

Solution:

(a) **True.** Proof by contradiction. Suppose the set of irrationals is countable. From Lecture note 10 we know that the set $\mathbb{Q}$ is countable. Since union of two countable sets is countable, this would imply that the set $\mathbb{R}$ is countable. But again from Lecture note 10 we know that this is not true. Contradiction!

(b) **True.** Multiplying both sides of the modular equation by 7 (the multiplicative inverse of 3 with respect to 10) we get $x \equiv 4 \pmod{10}$. The set of all intergers that solve this is $S = \{10k+4 : k \in \mathbb{Z}\}$ and it is clear that the mapping $k \in \mathbb{Z}$ to $10k+4 \in S$ is a bijection. Since the set $\mathbb{Z}$ is countably infinite, the set S is also countably infinite.

(c) **False.** Let $S \in \mathbb{R} \times \mathbb{R}$ denote the set of all real solutions for the given equation. For any $x' \in \mathbb{R}$, the pair $(x', y') \in S$ if and only if $y' = 1 - x'$. Thus $S = \{(x, 1-x) : x \in \mathbb{R}\}$. Besides, the mapping x to $(x, 1-x)$ is a bijection from $\mathbb{R}$ to S. Since $\mathbb{R}$ is uncountable, we have that S is uncountable too.

(d) **True.** Recall that a function $h : A \to B$ is injective iff $a_1 \neq a_2 \implies h(a_1) \neq h(a_2)$ for all $a_1, a_2 \in A$. Let $x_1, x_2 \in X$ be arbitrary such that $x_1 \neq x_2$. Since g is injective, we have $g(x_1) \neq g(x_2)$. Now, since f is injective, we have $f(g(x_1)) \neq g(g(x_2))$. Hence $f \circ g$ is injective.

(e) **False.** Recall that a function $h : A \to B$ is surjective iff $\forall b \in B, \exists a \in A$ such that $h(a) = b$. Let $g : \{0,1\} \to \{0,1\}$ be given by $g(0) = g(1) = 0$. Let $f : \{0,1\} \to \{0,1\}$ be given by $f(0) = 0$ and $f(1) = 1$. Then $f \circ g : \{0,1\} \to \{0,1\}$ is given by $(f \circ g)(0) = (f \circ g)(1) = 0$. Here f is surjective but $f \circ g$ is not surjective.

2 Counting Cartesian Products

For two sets A and B, define the cartesian product as $A \times B = \{(a,b) : a \in A, b \in B\}$.

(a) Given two countable sets A and B, prove that $A \times B$ is countable.

(b) Given a finite number of countable sets $A_1, A_2, \ldots, A_n$, prove that

$$A_1 \times A_2 \times \cdots \times A_n$$

is countable.

Solution:

(a) As shown in lecture, $\mathbb{N} \times \mathbb{N}$ is countable by creating a zigzag map that enumerates through the pairs: $(0,0), (1,0), (0,1), (2,0), (1,1), \ldots$. Since A and B are both countable, there exists a bijection between each set and a subset of $\mathbb{N}$. Thus we know that $A \times B$ is countable because there is a bijection between a subset of $\mathbb{N} \times \mathbb{N}$ and $A \times B : f(i,j) = (A_i, B_j)$. We can enumerate the pairs (a,b) similarly.

(b) Proceed by induction.
Base Case: $n = 2$. We showed in part (a) that $A_1 \times A_2$ is countable since both A_1 and A_2 are countable.
Induction Hypothesis: Assume that for some $n \in \mathbb{N}$, $A_1 \times A_2 \times \cdots \times A_n$ is countable.
Induction Step: Consider $A_1 \times \cdots \times A_n \times A_{n+1}$. We know from our hypothesis that $A_1 \times \cdots \times A_n$ is countable, call it $C = A_1 \times \cdots \times A_n$. We proved in part (a) that since C is countable and A_{n+1} are countable, $C \times A_{n+1}$ is countable, which proves our claim.

3 Undecided?

Let us think of a computer as a machine which can be in any of n states $\{s_0, \ldots, s_n\}$. The state of a 10 bit computer might for instance be specified by a bit string of length 10, making for a total of 2^{10} states that this computer could be in at any given point in time. An algorithm $\mathscr{A}$ then is a list of k instructions $(i_0, i_1, \ldots, i_{k-1})$, where each i_ℓ is a function of a state c that returns another state u and a number j describing the next instruction to be run. Executing $\mathscr{A}(x)$ means computing

$$(c_1, j_1) = i_0(x), \qquad (c_2, j_2) = i_{j_1}(c_1), \qquad (c_3, j_3) = i_{j_2}(c_2), \qquad \ldots$$

until $j_\ell \geq k$ for some ℓ, at which point the algorithm halts and returns $s_{\ell-1}$.

(a) How many iterations can an algorithm of k instructions perform on an n-state machine (at most) without repeating any computation?

(b) Show that if the algorithm is still running after $nk+1$ iterations, it will loop forever.

(c) Give an algorithm that decides whether an algorithm $\mathscr{A}$ halts on input x or not. Does your contruction contradict the undecidability of the halting problem?

Solution:

(a) Each of the k instruction can be called on at most n different states, therefore there are at most $n \cdot k$ distinct computations that can be performed during any execution. After $n \cdot k + 1$ iterations we must have repeated one of these computations.

(b) Since $nk + 1 > n \cdot k$, by the Pigeonhole Principle, $\mathscr{A}$ must repeat a computation $i_m(s_t)$ for some $(m,t) \in \{1,\ldots,n\} \times \{0,\ldots,k-1\}$. But we know that when $i_m(s_t)$ is performed the second time, its consective computations will be precisely the same that followed the first evaluation of $i_m(s_t)$. In particular, we will see $i_m(s_t)$ a third time, and hence a fourth, fifth time etc.

(c) From our solution to part (b) it follows that we only need to check whether after $nk + 1$ iterations, $\mathscr{A}(x)$ is still running or not. If it is, $\mathscr{A}(x)$ does not halt, otherwise it does. This does not contradict the undecidability of the halting problem, since it only states the inability to decide whether an *arbitrary* algorithm halts. Here we only proved the decidability for algorithms that can be run on an n-state machine, of which there are only finitely many!

4 Code Reachability

Consider triplets (M,x,L) where

```
M is a Java program
x is some input
L is an integer
```

and the question of: if we execute $M(x)$, do we ever hit line L?

Prove this problem is undecidable.

Solution:

Suppose we had a procedure that could decide the above; call it `Reachable(M, x, L)`. Consider the following example of a program deciding whether $P(x)$ halts:

```
Halt(P, x):
    def M(t):
        run P(x) #line 1 of M
        return #line 2 of M
    return Reachable(M, 0, 2)
```

Program M reaches line 2 if and only if $P(x)$ halted. Thus, we have implemented a solution to the halting problem — contradiction.