1 Countability: True or False

(a) The set of all irrational numbers \( \mathbb{R} \setminus \mathbb{Q} \) (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 \( f : Y \rightarrow Z \) and \( g : X \rightarrow Y \), let their composition \( f \circ g : X \rightarrow Z \) 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).

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.
3 Undecided?

Let us think of a computer as a machine which can be in any of \( n \) states \( \{s_1, \ldots, s_n\} \). 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 \( \mathcal{A} \) then is a list of \( k \) instructions \((i_0, i_1, \ldots, i_{k-1})\), where each \( i_\ell \) is a function of a state \( c \) that returns another state \( u \) and a number \( j \) describing the next instruction to be run. Executing \( \mathcal{A}(x) \) means computing

\[
(c_1, j_1) = i_0(x), \quad (c_2, j_2) = i_{j_1}(c_1), \quad (c_3, j_3) = i_{j_2}(c_2), \quad \ldots
\]

until \( j_\ell \geq k \) for some \( \ell \), 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 \( \mathcal{A} \) halts on input \( x \) or not. Does your construction contradict the undecidability of the halting problem?

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.