Extra Review Problems

This handout contains some extra review problems based on the topics people have asked us to cover in more detail. We hope that you find them useful when reviewing for the exam!

Problem One: Translating into Logic

i. Given the predicates

Person(p), which states that p is a person, and

ParentOf(p_1, p_2), which states that p_1 is the parent of p_2,

write a statement in first-order logic that says "someone is their own grandparent." (Paraphrased from an old novelty song.)

ii. Given the predicates

Natural(n), which states that n is a natural number, and

Integer(n), which states that n is an integer,

along with the function symbol $f(n)$, which represents some particular function f, write a statement in first-order logic that says "$f : \mathbb{N} \to \mathbb{Z}$ is a bijection."

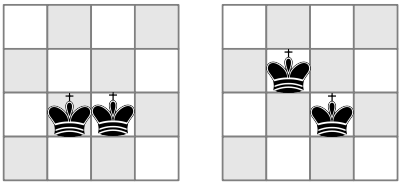
Problem Two: Translating Out Of Logic

For each first-order statement below, write an English sentence that describes what that sentence says. Then, determine whether the statement is true or false. No proofs are necessary.

- $\exists S.\ (Set(S) \wedge \forall x.\ x \notin S)$
- $\forall x.\ \exists S.\ (Set(S) \wedge x \notin S)$
- $\forall S.\ (Set(S) \to \exists x.\ x \notin S)$
- $\forall S.\ (Set(S) \wedge \exists x.\ x \notin S)$
- $\exists S.\ (Set(S) \wedge \exists x.\ x \notin S)$
- $\exists S.\ (Set(S) \to \forall x.\ x \in S)$
- $\exists S.\ (Set(S) \wedge \forall x.\ x \notin S \wedge \forall \mathrm{T}.\ (Set(T) \wedge S \neq T \to \exists x.\ x \in T))$
- $\exists S.\ (Set(S) \wedge \forall x.\ x \notin S \wedge \exists \mathrm{T}.\ (Set(T) \wedge \forall x.\ x \notin T \wedge S \neq T))$
- $\exists S.\ (Set(S) \wedge \forall x.\ x \notin S) \wedge \exists \mathrm{T}.\ (Set(T) \wedge \forall x.\ x \notin T))$

Problem Three: A Clash of Kings

Chess is a game played on an 8 × 8 grid with a variety of pieces. In chess, no two king pieces can ever occupy two squares that are immediately adjacent to one another horizontally, vertically, or diagonally. For example, the following positions are illegal:



Prove that it is impossible to legally place 17 kings onto a chessboard.

Problem Four: Coloring a Grid, Take Two

Suppose every point in a 3 × 7 grid is colored either red or blue. Prove that there must be four points of the same color that form a rectangle.

Problem Five: Repeated Squaring

In many applications in computer science, especially cryptography, it is important to compute exponents efficiently. For example, the RSA public-key encryption system, widely used in secure communication, relies on computing huge powers of large numbers. Fortunately, there is a fast algorithm called *repeated squaring* for computing x^y in the special case where y is a natural number.

The repeated squaring algorithm is based on the following function RS:

$$RS(x, y) = \begin{cases} 1 & \text{if } y = 0 \\ RS(x, y/2)^2 & \text{if } y \text{ is even and } y > 0 \\ x \cdot RS(x, (y-1)/2)^2 & \text{if } y \text{ is odd and } y > 0 \end{cases}$$

For example, we could compute 2^{10} using $RS(2, 10)$ follows:

In order to compute $RS(2, 10)$, we need to compute $RS(2, 5)^2$.
In order to compute $RS(2, 5)$, we need to compute $2 \cdot RS(2, 2)^2$.
In order to compute $RS(2, 2)$, we need to compute $RS(2, 1)^2$.
In order to compute $RS(2, 1)$, we need to compute $2 \cdot RS(2, 0)^2$.
By definition, $RS(2, 0) = 1$
so $RS(2, 1) = 2 \cdot RS(2, 0)^2 = 2 \cdot 1^2 = 2$.
so $RS(2, 2) = RS(2, 1)^2 = 2^2 = 4$.
so $RS(2, 5) = 2 \cdot RS(2, 2)^2 = 2 \cdot 4^2 = 32$.
so $RS(2, 10) = RS(2, 5)^2 = 32^2 = 1024$.

The RS function is interesting because it can be computed much faster than simply multiplying x by itself y times. Since RS is defined recursively in terms of RS with the y term roughly cut in half, RS can be evaluated using approximately $\log_2 y$ multiplications. (You don't need to prove this).

Prove that for any $x \in \mathbb{R}$ and any $y \in \mathbb{N}$, that $RS(x, y) = x^y$.

Problem Six: The Harmonic Series

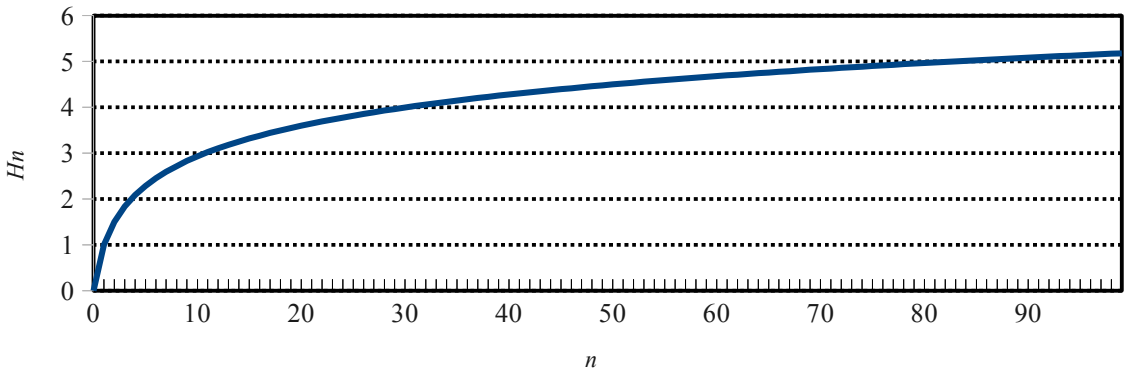
The *harmonic sequence* is the sequence $^1/_1$, $^1/_2$, $^1/_3$, $^1/_4$, … $^1/_n$, …. This sequence has numerous applications in mathematics and computer science. For example, it can be used both to describe the way that strings vibrate in musical instruments and to analyze the expected running time of the quicksort algorithm.

We can sum up the first n terms of the harmonic sequence to get the nth *harmonic number*:

$$H_n = \sum_{k=1}^{n} \frac{1}{k}$$

The first few harmonic numbers are $H_0 = 0$, $H_1 = 1$, $H_2 = {}^3/_2$, $H_3 = {}^{11}/_6$, $H_4 = {}^{25}/_{12}$, …

Since each harmonic number is the sum of progressively more terms of the harmonic series, we know that each harmonic number is bigger than the previous one. However, the terms of the harmonic series only get slightly larger each time; the difference between H_n and H_{n+1} is only $^1/_{n+1}$. As a result, the harmonic numbers grow very slowly, as seen below:



It seems like the harmonic numbers might get closer and closer to some limiting real number without ever crossing it. Amazingly, though, this is not the case. As n goes to infinity, H_n goes to infinity as well. It just does so very slowly.

Prove, by induction on n, that

$$H_{2^n} \geq \frac{n}{2} + 1$$

Problem Seven: Increasing Functions

A function $f : \mathbb{R} \to \mathbb{R}$ is called monotonically increasing iff for every $x, y \in \mathbb{R}$, if $x < y$, then $f(x) < f(y)$. Prove that any monotonically-increasing function must be injective.

Problem Eight: Infinity is Strange!

For any $n \in \mathbb{N}$, let $S_n = \{ k \in \mathbb{N} \mid k \geq n \}$. Prove for any $n \in \mathbb{N}$ that $|\mathbb{N}| = |S_n|$.