Extra Practice Problems 10

Here's another batch of practice problems to work through. Please let us know if there are any topics you'd specifically like some more practice with. We'd be happy to provide extra practice problems on those topics!

Problem One: Set Theory

(Midterm Exam, Fall 2015)

We can use set-builder notation to describe a set by giving a rule that describes what elements are in the set. Specifically, if $P(x)$ is some predicate, then the set

$$\{ x \mid P(x) \}$$

is the set containing all objects x where $P(x)$ is true (and no elements besides these).

Let's suppose that we have a set S that is a set of sets (that is, every element of S is itself a set). Formally, this means we're talking about a set S where

$$\forall T.\ (T \in S \to \mathit{Set}(T)).$$

If S is a set of sets, then we can take the intersection of all of the sets contained in S. The resulting set, denoted $\cap S$, is called the ***intersection of S***. For example, if

$$S = \{ \{1, 2, 3, 4\}, \{2, 3, 4, 5\}, \{3, 4, 5, 6\} \},$$

then $\cap S = \{3, 4\}$.

Intuitively, an object x is an element of $\cap S$ if x belongs to every element of S. We can use this intuition to come up with a formal definition of $\cap S$, which is given below:

$$\cap S = \{ x \mid \forall T.\ (T \in S \to x \in T) \}$$

This is the standard definition of the intersection of $\cap S$ that's used throughout set theory. However, this definition of $\cap S$ has a pretty major edge case.

Prove that the set $\cap \varnothing$ does not exist. *(Hint: Think back to Problem Set Four.)*

Problem Two: Induction

Let $k \geq 1$ be any natural number. Prove, by induction, that $(k+1)^n - 1$ is a multiple of k for all $n \in \mathbb{N}$.

Problem Three: Graphs

A directed graph is called ***strongly connected*** if for any pair of nodes u and v in the graph, there's a path from u to v and from v to u. In a directed graph, the ***indegree*** of a node is the number of edges entering it, and its ***outdegree*** is the number of edges leaving it. Find a strongly-connected graph with 137 nodes where each node's ***indegree*** is equal to its ***outdegree***.

Problem Four: First-Order Logic

Given the predicates

- *TM*(*M*), which states that *M* is a TM;
- *String*(*w*), which states that *w* is a string; and
- *Accepts*(*M*, *w*), which states that *M* accepts *w*,

Write a statement in first-order logic that says "the **RE** languages are closed under union."

Problem Five: Functions

This question explores properties of special classes of functions.

i. Prove or disprove: if $f : \mathbb{R} \to \mathbb{R}$ is a bijection, then $f(r) \geq r$ for all $r \in \mathbb{R}$.

ii. Prove or disprove: if $f : \mathbb{N} \to \mathbb{N}$ is a bijection, then $f(n) = n$ for all $n \in \mathbb{N}$.

iii. Prove or disprove: if $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ are bijections, then the function $h : \mathbb{R} \to \mathbb{R}$ defined as $h(x) = f(x) + g(x)$ is also a bijection.

Problem Six: Binary Relations

Let L be an arbitrary language over an alphabet Σ. We'll say that two strings $x, y \in \Sigma^*$ are ***indistinguishable*** relative to L, denoted $x \equiv_L y$, if the following is true:

$$\forall w \in \Sigma^*. (xw \in L \leftrightarrow yw \in L).$$

i. Prove that if L is any language over Σ, then $\equiv_L$ is an equivalence relation over Σ^*.

ii. Prove that if $x \equiv_L y$ and $x \in L$, then $y \in L$.

iii. Let $L = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid |w| \equiv_3 2 \}$. What are all the equivalence classes of $\equiv_L$?

Problem Seven: The Pigeonhole Principle

Suppose that *n* people are seated at a round table at a restaurant. Each of the *n* people orders a different entrée for dinner. The waiter brings all of the entrées out and places one dish in front of each person. Oddly enough, the waiter doesn't put anyone's dish in front of them.

Prove that there is some way to rotate the table so that at least two people have their entree in front of them.

Problem Eight: DFAs, NFAs, and Regular Expressions

Consider the following language over $\Sigma = \{ \texttt{O}, \texttt{E} \}$:

$$PARITY = \{ w \mid w \text{ has even length and has the form } \texttt{E}^n \text{ or } w \text{ has odd length and has the form } \texttt{O}^n \}$$

For example, $\texttt{EE} \in PARITY$, $\texttt{OOOOO} \in PARITY$, $\texttt{EEEE} \in PARITY$, and $\varepsilon \in PARITY$, but $\texttt{EEE} \notin PARITY$, $\texttt{EO} \notin PARITY$, and $\texttt{OOOO} \notin PARITY$.

i. Write a regular expression for *PARITY*.

ii. Design a DFA that accepts *PARITY*.

Problem Nine: Nonregular Languages

Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and let $L = \{ w \in \Sigma^* \mid w \text{ has the same number of } \texttt{a}\text{'s and } \texttt{b}\text{'s and } |w| \geq 10^{100} \}$.

i. Prove or disprove: *L* is not a regular language.

ii. Prove or disprove: there is at least one infinite subset of *L* that is regular.

Problem Ten: Context-Free Grammars

Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and let $L = \{ w \in \Sigma^* \mid w \text{ is a palindrome and } w \text{ contains } \texttt{abba} \text{ as a substring} \}$. Write a context-free grammar for *L*.

Problem Eleven: Turing Machines

Let $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{=}\}$. Draw the state-transition diagram of a TM for the $\{ w\texttt{=}w \mid w \in \{\texttt{a}, \texttt{b}\}^* \}$.

Problem Twelve: R and RE Languages

Earlier in this packet of problems you translated the statement "the **RE** languages are closed under union" into first-order logic. It turns out that this statement is true, but a bit trickier to prove that you might expect.

If we take a language $L \in$ **RE**, we know that we can get a recognizer M for it. A recognizer for L, in software, would be a function

```
bool inL(string w)
```

that takes as input a string w. If $w \in L$, then `inL(w)` returns true. If $w \notin L$, then `inL(w)` *may* return false, or it may loop infinitely.

Let L_1 and L_2 be **RE** languages. Below is an ***incorrect*** construction that purportedly is a recognizer for $L_1 \cup L_2$:

```
bool inL1uL2(string w) {
      return inL1(w) || inL2(w);
}
```

Here, `inL1` and `inL2` are recognizers for L_1 and L_2, respectively.

i. Give concrete examples of languages L_1 and L_2 and implementations of methods `inL1` and `inL2` such that the above piece of code is not a recognizer for $L_1 \cup L_2$. Justify your answer.

To show that the **RE** languages are closed under union, it's easiest to think about combining together two verifiers for the input languages to produce a verifier for their union.

ii. Using the verifier definition of **RE**, prove that the **RE** languages are closed under union.

Problem Thirteen: Impossible Problems

(We will cover the topics necessary to solve this problem on Wednesday, May 31st.)

Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and let $L = \{ \langle M \rangle \mid M \text{ is a TM and } \mathscr{L}(M) \subseteq \texttt{a*} \}$. Prove that $L \notin$ **R**.