Extra Practice Problems 2

Here's another compilation of practice problems you can use to review just about everything from this quarter.

Problem One: Cartesian Products and Subsets

Prove or disprove: if A, B, C, and D are sets where $A \times B \subseteq C \times D$, then $A \subseteq C$ and $B \subseteq D$.

Problem Two: 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 $RS(2, 10)$ as 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 $RS(x, y) = x^y$. *(Hint: use complete induction on y.)*

Problem Three: Partial Sums

Suppose that you have a set S of $n > 0$ natural numbers. Prove that there must be a nonempty subset of S where the sum of the numbers in that subset is a multiple of n. *(Hint: Number the elements of S as x_1, x_2, ..., x_n. Then, look at x_1, $x_1 + x_2$, $x_1 + x_2 + x_3$, etc.)*

Problem Four: Fun with DFAs and NFAs

Here's some true-or-false questions to ponder:

i. True or false: If D is a DFA over alphabet Σ and D has no accepting states, then $\mathcal{L}(D) = \varnothing$.

ii. True or false: If D is a DFA over alphabet Σ and D has no rejecting states, then $\mathcal{L}(D) = \Sigma^*$.

iii. True or false: If N is an NFA over alphabet Σ and N has no accepting states, then $\mathcal{L}(N) = \varnothing$.

iv. True or false: If N is an NFA over alphabet Σ and N has no rejecting states, then $\mathcal{L}(N) = \Sigma^*$.

Let $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}\}$ and let L be the following language:

$$L = \{ w \in \Sigma^* \mid \text{every character from } \Sigma \text{ appears at least once in } w \}$$

Any DFA for L must have at least 32 states (you don't need to prove this.)

v. Prove that any DFA for $\overline{L}$ must have at least 32 states.

vi. Design a reasonably-sized NFA for $\overline{L}$. This shows that even if you can't find a small NFA for a language, you might be able to find a small NFA for its complement.

Problem Five: Antitautonyms

Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and consider the language $L = \{ wx \mid w \in \Sigma^*, x \in \Sigma^*, |w| = |x|, \text{and } w \neq x \}$. Prove that L is not a regular language.

Problem Six: Closure Properties of CFGs

This question explores closure properties of CFLs.

i. Show that the context-free languages are closed under union, concatenation, and Kleene star.

ii. Although we didn't prove this, the context-free languages are not closed under complementation. In lecture, you saw a CFG for the language $\{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w \text{ is a palindrome} \}$, and on Problem Set Six you built a CFG for the complement of this language. Explain how this is possible even though the context-free languages aren't closed under complementation.

Problem Seven: Powers, Multiples, and 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 Eight: Strongly Connected 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 Nine: Closure Properties and 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 Ten: Properties of 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 Eleven: The Indistinguishability Relation

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:

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

Let L be a language over a set Σ. Prove that if $x \equiv_L y$, then for any string $z \in \Sigma^*$ we have $xz \equiv_L yz$.

Problem Twelve: Spin Me An Entree

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 Thirteen: Regular Languages and Parity

Consider the following language over $\Sigma = \{ \texttt{0}, \texttt{E} \}$:

$$PARITY = \{ w \mid w \text{ has even length and has the form } \texttt{E}^n \text{ or } w \text{ has odd length and has the form } \texttt{0}^n \}$$

For example, $\texttt{EE} \in PARITY$, $\texttt{00000} \in PARITY$, $\texttt{EEEE} \in PARITY$, and $\varepsilon \in PARITY$, but $\texttt{EEE} \notin PARITY$, $\texttt{E0} \notin PARITY$, and $\texttt{0000} \notin PARITY$.

i. Write a regular expression for *PARITY*.

ii. Design a DFA that accepts *PARITY*.

Problem Fourteen: Giant Balanced Strings

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 Fifteen: CFGs and Swedish Pop Music

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 Sixteen: Power Sets and Cartesian Products

Prove or disprove: there are sets A and B where $\wp(A \times B) = \wp(A) \times \wp(B)$.

Problem Seventeen: The Well-Ordering Principle

The ***well-ordering principle*** states that if $S \subseteq \mathbb{N}$ and $S \neq \varnothing$, then S contains an element n_0 that is less than all other elements of S. There is a close connection between the well-ordering principle and the principle of mathematical induction.

Suppose that P is some property such that

- $P(0)$
- $\forall k \in \mathbb{N}.\ (P(k) \to P(k+1))$

Using the well-ordering principle, *but without using induction*, prove that $P(n)$ holds for all $n \in \mathbb{N}$. This shows that if you believe the well-ordering principle is true, then you must also believe the principle of mathematical induction.

Problem Eighteen: Restricting and Manipulating Logic

Consider the following formula in first-order logic:

$$\forall x \in \mathbb{R}.\ \forall y \in \mathbb{R}.\ (x < y \to \exists p \in \mathbb{Z}.\ \exists q \in \mathbb{Z}.\ (q \neq 0 \wedge x < p/q \wedge p/q < y))$$

This question explores this formula.

i. Translate this formula into plain English. As a hint, there's a very simple way of expressing the concept described above.

ii. Rewrite this formula so that it doesn't use any universal quantifiers.

iii. Rewrite this formula so that it doesn't use any existential quantifiers.

iv. Rewrite this formula so that it doesn't use any implications.

v. Negate this formula and push the negations as deep as possible.

Problem Nineteen: Restrictions of Relations

Let R be a binary relation over a set A. For any set $B \subseteq A$, we can define the ***restriction of R to B***, denoted $R|_B$, to be a binary relation over the set B defined as follows:

$$x\ R|_B\ y \quad \text{if} \quad xRy.$$

In other words, the relation $R|_B$ behaves the same as R, but only on the elements of B.

i. Prove or disprove: if R is an equivalence relation over a set A and B is an arbitrary subset of A, then $R|_B$ is an equivalence relation over B.

ii. Prove or disprove: if R is a strict order over a set A and B is an arbitrary subset of A, then $R|_B$ is a strict order over B.

iii. Prove or disprove: there is a strict order R over a set A and a set $B \subseteq A$ such that $R|_B$ is an equivalence relation.

iv. Prove or disprove: there is an equivalence relation R over a set A and a set $B \subseteq A$ such that $R|_B$ is a strict order.

Problem Twenty: Functions and Relations, Together!

(Midterm Exam, Spring 2015)

In this question, let $A = \{1, 2, 3, 4, 5\}$. Let $f : A \to A$ be an arbitrary function from A to A that we know is ***not a surjection***. We can then define a new binary relation $\sim_f$ as follows: for any $a, b \in A$, we say $a \sim_f b$ if $f(a) = b$. Notice that this relation depends on the particular non-surjective function f that we pick; if we choose f differently, we'll get back different relations. This question explores what we can say with certainty about $\sim_f$ knowing only that its domain and codomain are A and that it is not a surjection.

Below are the six types of relations we explored over the course of this quarter. For each of the types, determine which of the following is true:

- The relation $\sim_f$ is ***always*** a relation of the given type, regardless of which non-surjective function $f : A \to A$ we pick.
- The relation $\sim_f$ is ***never*** a relation of the given type, regardless of which non-surjective function $f : A \to A$ we pick.
- The relation $\sim_f$ is ***sometimes, but not always*** a relation of the given type, depending on which particular non- surjective function $f : A \to A$ we pick.

Since these options are mutually exclusive, check only one box per row. *(Hint: Draw a lot of pictures.)*

$\sim_f$ is reflexive	☐ ***Always***	☐ ***Sometimes, but not always***	☐ ***Never***
$\sim_f$ is irreflexive	☐ ***Always***	☐ ***Sometimes, but not always***	☐ ***Never***
$\sim_f$ is symmetric	☐ ***Always***	☐ ***Sometimes, but not always***	☐ ***Never***
$\sim_f$ is asymmetric	☐ ***Always***	☐ ***Sometimes, but not always***	☐ ***Never***
$\sim_f$ is transitive	☐ ***Always***	☐ ***Sometimes, but not always***	☐ ***Never***
$\sim_f$ is an equivalence relation	☐ ***Always***	☐ ***Sometimes, but not always***	☐ ***Never***
$\sim_f$ is a strict order	☐ ***Always***	☐ ***Sometimes, but not always***	☐ ***Never***

Problem Twenty One: Permutation Parity*

Let n be an odd natural number and consider the set $S = \{1, 2, 3, \ldots, n\}$. A ***permutation*** of S is a bijection $\sigma : S \to S$. In other words, σ maps each element of S to some unique element of S and does so in a way such that no two elements of S map to the same element.

Let σ be an arbitrary permutation of S. Prove that there is some $r \in S$ such that $r - \sigma(r)$ is even.

Problem Twenty Two: Reversing Regular Languages

If w is a string, then w^R represents the reversal of that string. For example, the reversal of "table" is "elbat." If L is a language, then L^R is the language $\{ w^R \mid w \in L \}$ consisting of all the reversals of the strings in L.

It turns out that the regular languages are closed under reversal.

i. Give a construction that turns an NFA for a language L into an NFA for the language L^R. No proof is necessary.

ii. Give a construction that turns a regular expression for a language L into a regular expression for the language L^R. No proof is necessary.

Problem Twenty Three: Centrist Languages

Prove that the language $\{ w \in \{\texttt{a}, \texttt{b}\}^* \mid |w| \equiv_3 0$ and the middle third of the characters in w contains at least one $\texttt{a}$ $\}$ is not regular.

Problem Twenty Four: Parenthesis Parity

Let $\Sigma = \{(,)\}$ and let $L = \{ w \in \Sigma^* \mid w$ is a string of balanced parentheses and w has an even number of open parentheses $\}$. Write a CFG for L.

Problem Twenty Five: Nonregular Languages via a Different Path

The Myhill-Nerode theorem is a powerful tool for proving that languages aren't regular, but it might not be the easiest way to prove that a given language isn't regular. This problem explores a different route you can take to prove that various languages aren't regular.

i. Prove that if L_1 is a language, L_2 is a regular language, and $L_1 \cap L_2$ is not regular, then L_1 is not regular.

ii. Using your result from part (i), but without using the Myhill-Nerode theorem, prove that the language $L = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w$ has the same number of $\texttt{a}$'s as $\texttt{b}$'s $\}$ is not regular.

* Adapted from http://www.cut-the-knot.org/do_you_know/pigeon.shtml.

Problem Twenty Six: Just a Few More Grammars

Below is a list of alphabets and languages over those alphabets. Design a CFG for each language.

i. Let $\Sigma = \{1, \geq\}$ and let $L = \{ 1^m \geq 1^n \mid m, n \in \mathbb{N} \text{ and } m \geq n \}$. Write a CFG for L.

ii. On Problem Set 6, you explored the language $L_1 = \{ 1^m+1^n=1^{m+n} \mid m, n \in \mathbb{N} \}$ over the alphabet {1, +, =} Consider the following generalization of this language, which we will call L_2, which consists of all strings describing unary encodings of two sums that equal one another. For example:

1 + 3 = 4 would be encoded as `1+111=1111`

4 = 1 + 3 would be encoded as `1111=1+111`

2 + 2 = 1 + 3 would be encoded as `11+11=1+111`

2+0+2+0=0+4+0 would be encoded as `11++11+=+1111+`

0=0 would be encoded as `=`

Notice that there can be any number of summands on each side of the `=`, but there should be exactly one `=` in the string; thus `1=1=1` $\notin L_2$. Write a CFG for L_2.

iii. Let $\Sigma = \{(,), [,]\}$ and let $L = \{w \in \Sigma^* \mid w$ is a string of balanced parentheses and brackets$\}$. This means that all parentheses and brackets must match one another, and collectively they must obey the appropriate nesting rules. For example, `([])[]` $\in L$, but `([)]` $\notin L$. Write a CFG for L.

Problem Twenty Seven: Translating Out Of Logic

For each first-order statement below, write a short English sentence that describes what that sentence says. While you technically *can* literally translate these statements back into English, you'll probably have better luck translating them if you try to think about what they really mean. Then, determine whether the statement is true or false based on what you know about sets and set theory.

- $\exists S.\ (Set(S) \wedge \forall x.\ x \notin S)$
- $\forall x.\ \exists S.\ (Set(S) \wedge x \notin S)$
- $\forall S.\ (Set(S) \rightarrow \exists x.\ x \notin S)$
- $\forall S.\ (Set(S) \wedge \exists x.\ x \notin S)$
- $\exists S.\ (Set(S) \wedge \exists x.\ x \notin S)$
- $\exists S.\ (Set(S) \rightarrow \forall x.\ x \in S)$
- $\exists S.\ (Set(S) \wedge \forall x.\ x \notin S \wedge \forall T.\ (Set(T) \wedge S \neq T \rightarrow \exists x.\ x \in T))$
- $\exists S.\ (Set(S) \wedge \forall x.\ x \notin S \wedge \exists T.\ (Set(T) \wedge \forall x.\ x \notin T \wedge S \neq T))$
- $\exists S.\ (Set(S) \wedge \forall x.\ x \notin S) \wedge \exists T.\ (Set(T) \wedge \forall x.\ x \notin T))$

Problem Twenty Eight: Rock, Paper, Scissors

(Final Exam, Winter 2012)

The number of characters in a regular expression is defined to be the total number of symbols used to write out the regular expression. For example, `(a U b)*` is a six-character regular expression, and `ab` is a two-character regular expression.

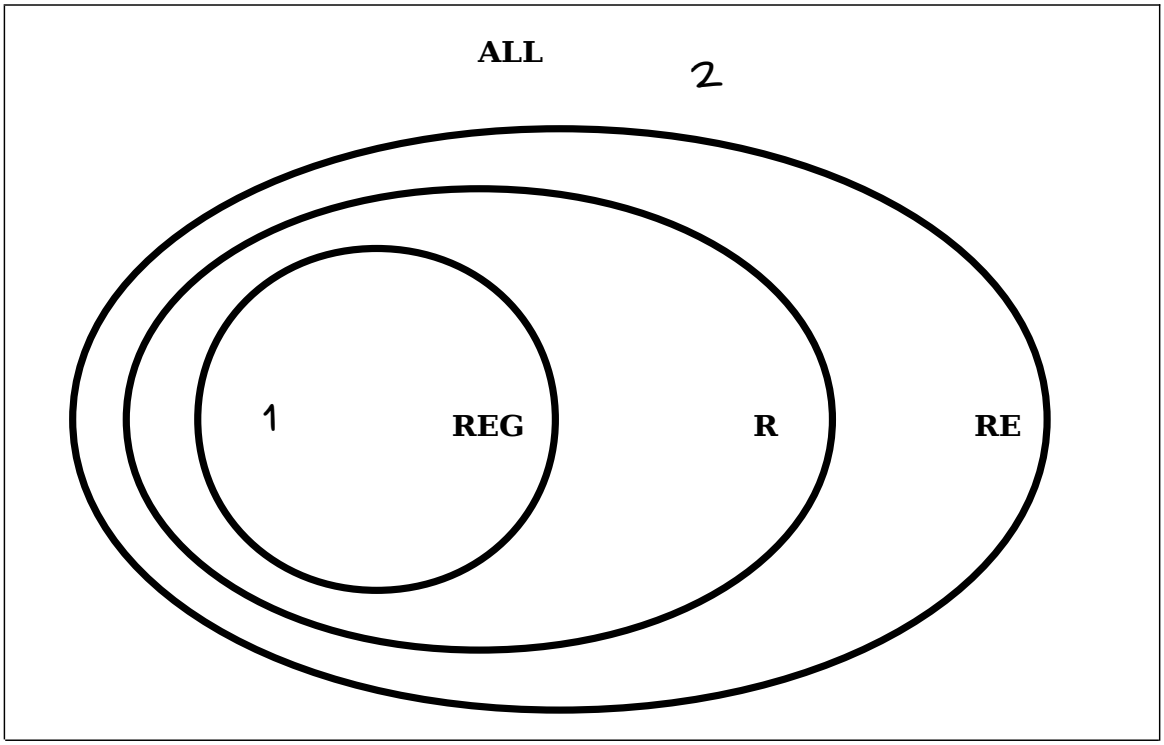
Let Σ = {`a`, `b`}. Find examples of all of the following:

- A regular language over Σ with a one-state NFA but no one-state DFA.
- A regular language over Σ with a one-state DFA but no one-character regular expression.
- A regular language over Σ with a one-character regular expression but no one-state NFA.

Prove that all of your examples have the required properties.

Problem Twenty Nine: The Lava Diagram

Below is a Venn diagram showing the overlap of different classes of languages we've studied so far. We have also provided you a list of numbered languages. For each of those languages, draw where in the Venn diagram that language belongs. As an example, we've indicated where Language 1 and Language 2 should go. No proofs or justifications are necessary, and there is no penalty for an incorrect guess.



1. Σ^*
2. L_D
3. $\{ \langle M, w \rangle \mid M$ is a TM, w is a string, and M accepts w^n for ***every*** natural number $n \}$
4. $\{ \langle M, w \rangle \mid M$ is a TM, w is a string, and M accepts w^n for ***at least one*** natural number $n \}$
5. $\{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w$ contains the same number of copies of the substrings `aabb` and `bbaa` $\}$
6. $\{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w$ contains the same number of copies of the substrings `ab` and `ba` $\}$

   (*Careful, this one is tricky: try writing out some sample strings in this language before you answer. The substrings* `ab` *and* `ba` *are allowed to overlap.*)
7. $\{ \langle M, w \rangle \mid M$ is a TM, w is a string, M loops on w, and $|\langle M, w \rangle| \leq 10^{137} \}$
8. $\{ \langle M \rangle \mid M$ is a TM and M loops on $\langle M \rangle \}$

Problem Thirty: Arden's Lemma

When you were first learning algebra, you probably learned a family of techniques to solve equations in which a variable x was on both sides of an equals sign. For example, you probably learned how to look at a formula like

$$x^2 = ax + b$$

and to use the quadratic formula to solve for x.

It's also possible to set up equations involving some unknown that appears on both sides of an equals sign, but where the quantities involved are *languages* rather than numbers. For example, if A and B are languages, you may want to determine what languages X satisfy the equality

$$X = AX \cup B.$$

Just as the quadratic formula is a useful tool for solving for x given a quadratic equation, in formal language theory there's a result called ***Arden's lemma*** that's useful for solving for X in an equation of the above form. Specifically, Arden's lemma says that, given the equality $X = AX \cup B$, you are guaranteed that

$$A^*B \subseteq X.$$

In this problem, we're going to ask you to prove Arden's lemma.

Let's begin with a refresher of the key terms and definitions involved. As a reminder, if L_1 and L_2 are languages over an alphabet Σ, then the ***concatenation of L_1 and L_2***, denoted L_1L_2, is the language

$$L_1L_2 = \{ wx \mid w \in L_1 \text{ and } x \in L_2 \}.$$

From concatenation, we can define ***language exponentiation*** of a language L inductively as follows:

$$L^0 = \{\varepsilon\} \qquad\qquad L^{n+1} = LL^n$$

You may find these formal terms helpful in the course of solving this problem.

i. Let A and B be arbitrary languages over some alphabet Σ. Prove, by induction, that if $X = AX \cup B$, then $A^nB \subseteq X$ for every $n \in \mathbb{N}$. Please use the formal definitions of concatenation, language exponentiation, union, and subset in the course of writing up your answer.

If you'll recall, we formally defined the ***Kleene closure*** of a language L over Σ to be the language

$$L^* = \{ w \in \Sigma^* \mid \text{there is some } n \in \mathbb{N} \text{ such that } w \in L^n \}.$$

ii. Let A and B be arbitrary languages over some alphabet Σ. Using your result from part (i) of this problem and the formal definition of L^*, prove that if $X = AX \cup B$, then $A^*B \subseteq X$.