Homework 8: Finite Automata; Fundamentals Review

This is the last homework :D. Unused late days will be converted to concept checks. For every late day you haven't used, we will convert a concept check to full credit (i.e., we assume you would have gotten full credit had you gotten extra time).

We will not be able to return feedback on HW8 until after the final.

If you work with others (and you should!), remember to follow the collaboration policy outlined in the syllabus.

In general, you are graded on both the clarity and accuracy of your work. Your solution should be clear enough that someone in the class who had not seen the problem before would understand it.

To help with formatting of English proofs, we've published a style guide on the website containing some tips.

Finally, be sure to read the grading guidelines for more information on what we're looking for.

Grin boxes will appear before Wednesday November 27. You can start to think about them now.

1. These are pretty l0000ng strings [20 points]

Let 0^n mean a string of n zeros. Let S be the set of strings defined as follows:

Basis Steps: $0^5 \in S$, $0^6 \in S$
Recursive Step: If $0^x, 0^y \in S$ then $0^x \cdot 0^y \in S$ where $\cdot$ is string concatenation.

Show that, for every integer $n \geq 20$ the set S contains the string 0^n.

Caution: Structural Induction is not the best tool for this problem. Structural induction shows $\forall x \in S(P(x))$. You're analyzing what the elements of S are in this problem, not proving a predicate holds for all elements of S.

2. Sets of work over time [12 points]

(a) Let A, B be arbitrary sets. Show that $\mathcal{P}(A \setminus B) \setminus \{\emptyset\} \subseteq \mathcal{P}(A) \setminus \mathcal{P}(B)$.

(b) Is it true that $\mathcal{P}(A \setminus B) \setminus \{\emptyset\} = \mathcal{P}(A) \setminus \mathcal{P}(B)$ for all sets A and B? If true, provide a proof (you may cite work from part (a)); otherwise, disprove the claim using a counterexample.

3. Context Is Everything. Except for Context-Free Grammars (Online) [15 points]

For each of the following languages, construct a context-free grammar that generates exactly the given language.

You will submit (and check!) your answers online at https://grin.cs.washington.edu/. Think carefully before entering a submission; you only have 10 guesses. Because these are auto-graded, we will not award partial credit.

(a) All strings in the form of $\{a^i b^j c^{i+3j} : i, j \geq 0\}$.

(b) All binary strings that contain at most two 1s and at least one 0.

(c) The set of all strings over $\{0, 1, 2\}$ such that the sum of its digits is an even number.
For example, the string 1102 is in this set since $1 + 1 + 0 + 2 = 4$ is even.

4. Build DFAs (Online) [15 points]

For each of the following languages, construct a **DFA** that recognizes **exactly** the given language. You should submit (and check!) your answers online at https://grin.cs.washington.edu/

Think carefully before entering your DFA; you have a limited number of guesses. Because these are auto-graded, we will not award partial credit.

(a) Binary strings that have an even number of 1's and end in a 0.

(b) Binary strings that start with a 1 and have at most three zeros.

(c) Binary strings that either contain the substring 10 or the substring 111 (The string can contain either substring or both to be accepted!)

5. Build NFAs (Online) [15 points]

For each of the following languages, construct an **NFA** that recognizes **exactly** the given set of strings. You should submit (and check!) your answers online at https://grin.cs.washington.edu/

Think carefully before entering your NFA; you have a limited number of guesses. Because these are auto-graded, we will not award partial credit.

(a) Binary strings that meet both of the following conditions:

- The string contains "11"
- every '0' is immediately followed by at least one '1'

(b) Strings matched by the regular expression $(0 \cup 010 \cup 0110)^*(\varepsilon \cup 1 \cup 11)$

(c) Binary strings that **either** end in "101" **or** contain "00" **but not both**.
Hint: Remember that two NFAs can be combined using ε-transitions to create a new NFA.

6. Proving Legendre's Conjecture...? [10 points]

Let the domain of discourse be positive integers. We've been using the `PRIME(x)` predicate many times but never formally defined it. We say x is a prime number if it is a positive integer not equal to one and for every pair of positive integers a, b where $ab = x$, then $a = 1$ or $b = 1$.

(a) Define `PRIME(x)` in predicate logic by translating the formal definition above. You must translate the English version above (do not use a different formulation, even if it's equivalent). You may use standard arithmetic notation (including $+, -, =, \leq$, etc.) in all parts of this problem [2 points]

(b) The Legendre Conjecture states that there is a prime number between n^2 and $(n+1)^2$ for every positive integer n. For example: 3 lies in between 1^2 and $(1+1)^2 = 4$, and so on (it is not known to the 311 staff if the conjecture is true).

Write a predicate logic statement which is true if and only if **the Legendre Conjecture** holds. Your answer should use the predicate `PRIME(x)` from part (a). [2 points]

(c) Negate the statement you wrote in part (b) (your answer should be in predicate notation). Show your work, but leave your answers as symbols. You don't have to name rules as you use them. Your final answer must have negations applied only to single predicates. You do not need to simplify to make domain restriction explicit in your translation [3 points]

(d) Translate your answer from (c) into English. You do not need to "take advantage of domain restriction" in your translation. [3 points]

7. Being Related [9 points]

This problem asks you to prove statements you haven't seen before. Even though the statements themselves are new, the structure of these statements **are** things you have seen before. Start by asking what the skeleton of this type of proof should be, and you'll be off to a good start.

Hint: These proofs are usually much cleaner if you give variable names to the vertices in the path!

Let $G = (V, E)$ be an *undirected* graph. We discussed *directed* graphs in lecture 20 (see the slides for more details). Unlike directed graphs, an undirected graph has the property that if $(v, z) \in E$, then $(z, v) \in E$ (i.e., all edges are bidirectional).

Define the set

$$R = \{(v, z) : \text{There is a path from } v \text{ to } z \text{ in } G\}$$

Notice that $R \subseteq V \times V$, since it's a set of ordered pairs, where each element of the ordered pair is a vertex of G.

Recall that a path between vertices v_1 and v_n is a sequence of vertices $(v_1, v_2, \ldots, v_n)$ such that $(v_i, v_{i+1}) \in E$ for all $i \in \{1, \ldots, n-1\}$.

(a) Prove that for all vertices $v \in V$, we have $(v, v) \in R$. [3 points] This property is known as *reflexivity*.
Hint: This proof takes advantage of an edge case in the definition of path.

(b) Prove that for all $v, z \in V$, if $(v, z) \in R$, then $(z, v) \in R$. [3 points] This property is known as *symmetry*.

(c) Prove that for all $v, z, w \in V$, if $(v, z) \in R$ and $(z, w) \in R$, then $(v, w) \in R$. [3 points] This property is known as *transitivity*.

(d) A set of ordered pairs (like R) is called a **relation**. The three facts you showed, taken together, mean that R is a specific type of relation called an *equivalence relation*. With an equivalence relation, we can partition the set V into disjoint sets $C_1, \ldots, C_n$ such that $(v, z) \in R$ (i.e., there is a path from v to z) if and only if $v, z \in C_i$ for the same $i \in \{1, \ldots, n\}$. These are known as the *connected components* of G. This concept will show up again in classes such as CSE 332/421. **You do not need to submit anything for this part.** [0 points]

8. Prove a language is not regular [0 points]

This problem isn't required for this homework, because we won't discuss the topic until shortly before the homework is due. But this topic will show up on the final, so you may want to practice it! We'll release solutions for this problem like the others.

*This problem is **not** extra credit.*

Do not submit anything for this part.

Prove that the following language is not regular: The set of all strings over $\{a, b, c, ...z, \#\}$ of the form $x\#y$, with $x, y \in \{a, b, \ldots, z\}^*$ and y is the reversal of x.

The "reversal" of a string, is the string going from right-to-left (instead of left-to-right). For example

- abc#cba is in the language.
- aaab#baaa is in the language.
- ab###ba is not in the language (you can only have a single #).

- aabb#bba is not the language.
- abc#abc is not in the language.

9. Extra Credit: Going Back to the Well [0 points]

In class, we'll use proof by contradiction as a method of showing languages are not regular. Another common way to prove languages are not regular is "The Pumping Lemma."

Formally, the pumping lemma says: If L is a regular language, then there exists an integer p such that for all $w \in L$, if $\texttt{len}(w) \geq p$, then there is a way to divide w into substrings x, y, z such that:

- $w = xyz$ (i.e. x, y, z break w into pieces)
- $\texttt{len}(xy) \leq p$
- $\texttt{len}(y) \geq 1$
- $\forall i \in \mathbb{N}$, $xy^iz \in L$

The lemma says you can break any string of L into pieces, where the "middle" piece is length at least 1, such that you can "pump" the string w by inserting extra copies of y into the string while still having the new string also be in L

For some intuition: roughly, p corresponds to the number of states in a hypothetical DFA for L. Once a string w has more than p characters, you have to have a cycle when the machine processes w, but then you could go around the cycle as many times as you want (including 0 times) and still end up in the same accepting state at the end. So some y in the string (what's read when you go around the cycle) could be duplicated any number of times and still be accepted. The difficulty of using the lemma to prove irregularity is we don't know which part of the string the cycle would be in (we're doing proof by contradiction – the language is irregular, so there isn't even a DFA in real life!) so these proofs often have cases based on all the places y could be in the string.

Use the pumping lemma to show $\{a^{311}b^nc^m : n = 311+m\}$ is not regular. The hardest thing about using the pumping lemma is getting the quantifiers right (there are a lot of them...) make sure you're declaring and using the arbitrary variables as arbitrary, and saying what the existential variables are. Usually this proof is done by contradiction (suppose L is regular and use the pumping lemma's guarantee to derive a contradiction) or contrapositive (show the conclusion of the pumping lemma does not hold, and apply the contrapositive to get that the language is not regular). You should feel free to look online or in the textbook for an example proof or two to get a sense of how they usually go.

10. Extra Credit: An Odd Grammar? [0 points]

Let $L = \{0^i1^j : i, j \in \mathbb{N} \wedge i \neq 2j + 1\}$. This is an irregular language, but is it context-free (i.e., there's a context-free grammar representation for L)? If L is context-free, provide a context-free grammar and give a brief explanation of the purpose of each non-terminal. Otherwise, explain why no such context-free grammar exists (you do not have to provide a formal argument).

11. Feedback [1 point]

Answer these questions on the separate gradescope box for this question.

Please keep track of how much time you spend on this homework and answer the following questions. This can help us calibrate future assignments and future iterations of the course, and can help you identify which areas are most challenging for you.

- How many hours did you spend working on this assignment (excluding any extra credit questions, if applicable)? Report your estimate to the nearest hour.
- Which problem did you spend the most time on?

- Any other feedback for us?