Homework 8: Finite Automata; Fundamentals Review

This homework comes in two parts.

Part 1 Due date: Wednesday March 6th at 11:59 PM
Part 2 Due date: Friday March 8th at 11:59 PM

You may use up to **3** late days for Part 1 and up to **1** late day for Part 2. We will have two separate grade scope submission boxes. We calculate late days for this assignment as **max(part 1 lateness, part 2 lateness)**, therefore using one late day allows you to submit **both** parts one day later (e.g. one late day lets you submit part 1 on Thursday and part 2 on Saturday). We will not accept Part 2 past Saturday.

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 (assuming 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.

Part I

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: $0^4 \in S$, $0^5 \in S$
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 $n \geq 12$ 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. Bijectivity [10 points]

Let the function $f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z} \times \mathbb{Z}$ be given by $f((x, y)) = (-y, x)$.

(a) Prove that f is one-to-one. [5 points]

(b) Prove that f is onto. [5 points]

3. Build DFAs (Online) [15 points]

For each of the following languages, construct a DFA that accepts 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 DFA; you have a limited number of guesses. Because these are auto-graded, we will not award partial credit.

(a) Construct a DFA that recognizes the language of binary strings that contain the substring 11 and an even number of 0s.

(b) Construct a DFA that accepts binary strings with at most two 1's and an odd number of 0's

(c) Construct a DFA that accepts binary strings such that none of their runs of 1s have an odd length > 2.

A "run" of 1s is a string of consecutive 1s with edges at the start of the string, end of the string, or adjacent to a 0. "1" contains exactly one run of 1s (of length 1).

- "111101" contains two runs of 1s (one of length 4, the other of length 1).
- For example, "1100111" is not accepted because the last three characters are an odd-length run of 1s. But "0110001" is accepted because the first run of 1s has length 2, and the second run of 1s has length 1 (even though this is odd, it is fine since the run is not > 2). "00000" is also accepted because it contains no run of 1s.

4. Build NFAs (Online) [15 points]

For each of the following languages, construct an **NFA** that accepts 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) Construct an NFA which recognizes the set of binary strings that contain "11" **and** do not contain "00".

(b) Construct an NFA which recognizes the same language as the regular expression $(1 \cup 01 \cup 001)^*(\epsilon \cup 0 \cup 00)$

(c) Binary strings that **either** have every occurrence of a 0 immediately followed by a 1 **or** contain at least two 1s **but not both**.

5. Proving Goldbach'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 every pair of positive integers a, b where $ab = x$, then that means $a = 1$ or $b = 1$.

(a) Define `PRIME(x)` in predicate logic by translating the formal definition above. You may use standard arithmetic notation (including $+, -, =, \leq$, etc.) in all parts of this problem [2 points]

(b) The Goldbach Conjecture states that every even integer greater than 2 can be expressed as the sum of two prime numbers. For example: $4 = 2 + 2$, $6 = 3 + 3$, $8 = 3 + 5$, $10 = 3 + 7 = 5 + 5$, and so on (it is not known to the 311 staff if the conjecture is true)[1].

Write a predicate logic statement which is true if and only if **the Goldbach Conjecture** holds. Your answer should use your predicate from part (a). You can also use the the predicate `EVEN(x)` in your answer. [2 points]

[1]The conjecture has been shown to hold for all integers less than $4 \cdot 10^{18}$ but remains unproven despite considerable effort; we're willing to offer extra credit to anyone who can prove the Goldbach Conjecture. We'd also happily write and publish the paper with you. Maybe also give you a PhD and a Fields Medal as well.

(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 "take advantage of domain restriction." [3 points]

(d) Translate your answer from (c) into English. [3 points]

6. Another Set Proof [14 points]

Prove that for any sets, if $A \subseteq B$, then $A \setminus B = \emptyset$

Part II

7. Prove a language is not regular [15 points]

This problem uses the technique from Monday's lecture. You can look ahead at the slides or wait until Monday.

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.

8. Extra Credit: Going Back to the Well [0 point]

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.

9. 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?