Homework 8: Finite Automata; Fundamentals Review

Due date: Wednesday March 6th at 10 PM.

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

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.
We sometimes describe approximately how long our explanations are. These are intended to help you understand approximately how much detail we are expecting.

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

1. Like Divides, but more complicated [10 points]

Define the following relation R on the positive integers. $(a, b) \in R$ if and only if a/b is an even integer.

So $(2, 4) \notin R$, because $2/4 = 1/2$ is not an integer.
$(30, 5) \in R$ because $30/5 = 6 = 2 \cdot 3$.

(a) Prove that R is a transitive relation. Be sure to explicitly introduce arbitrary variables as arbitrary. [7 points]

(b) Prove that R is not a symmetric relation (Hint: since 'symmetric' is a $\forall$ statement, you're proving an $\exists$ statement here!) [3 points]

2. Taking this homework in a new direction [22 points]

Call a directed graph **acyclic** if it has no cycles. Choose some acyclic directed graph G. For vertices u and v, we'll say $u \prec_G v$ if there is a path from u to v in G. Notice that the graph G might be very different from the graph representation of $\prec_G$. Don't confuse them below!

Recall the definition of a path: A path is a list vertices: $v_0, v_1, ..., v_k$ (with $k \geq 0$) such that (v_i, v_{i+1}) is an edge for all $0 \leq i \leq k-1$. Vertices (and edges) are allowed to repeat in a path. Note that if you have a path with $k = 0$ (i.e. with only one vertex), then the requirement for edges is vacuously true.

(a) Prove that if G is acyclic then $\prec_G$ is reflexive. Most of this proof is in the formatting! Be sure to introduce variables carefully [3 points]

(b) Prove that if G is acyclic then $\prec_G$ is antisymmetric (Hint: This proof is easiest if done by contradiction; be careful about where you start the proof!) [8 points]

(c) Prove that if G is acyclic then $\prec_G$ is transitive. [7 points]

A total-ly new direction

From parts a,b,c together we have that (for acyclic graphs, G) $\prec_G$ is a partial order (that was the "behaves kinda like $\leq$" prototype relation). Recall that for partial orders, some pairs of elements might not relate in either direction (for example: divides is a partial order, but 2 does not divide 3 and 3 does not divide 2).

A **total order** is a partial order where every element can be compared to every other (i.e. $\forall a \forall b (a \prec b \vee b \prec a)$).

(d) Draw (or otherwise describe) an acyclic graph G so that $\prec_G$ is not a total order. Give a 1 sentence description of why it is not a total order (you do not need a full proof). [2 points]

(e) Draw (or otherwise describe) an acyclic graph G so that $\prec_G$ is a total order. Give a 1 sentence description of why it is a total order (you do not need a full proof). [2 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) Binary strings with at least two 1s.

(b) Binary strings that have at least one 1 and an even number of 0s.

(c) Binary strings such that none of their runs of 1s have odd length.
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. "11" contains exactly one run of 1s (of length 2).
"111101" contains two runs of 1s (one of length 4, the other of length 1).
For example 1110011 is not accepted because the first three characters are an odd-length run of 1s. But 110011110 is accepted the first run of 1s has length two, and the second run of 1s has length 4.

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) The set of binary strings that contain 11 **and** do not contain 00. [7 points]

(b) The set of binary strings that contain 11 **or** do not contain 00. [8 points]

5. We're here to PUMP [:clap:] you up [11 points]

The pumping lemma for regular languages is a tool for showing languages are not regular. In its usual statement it says this for all languages L:

If L is a regular language, then:
there exists a natural number p such that
for all $w \in L$ with len$(w) > p$,
w can be broken up[1] into three substrings x, y, z such that
$y \neq \varepsilon$, and for all natural numbers i, replacing y with y^i yields a string that is also in L.

Recall that y^i means y concatentated with itself i times (and $y^0 = \varepsilon$). Let your domain of discourse be natural numbers, strings, and languages. Use the predicates `naturalNumber()`, `language()`, `string()` to do domain restriction. You can use standard string and set operations (like string concatenation, $=$, $\neq$, $\in$, etc.) and the predicate `elementOfL`(x) that returns true if and only if $x \in L$.

(a) Translate just the last two lines (starting with "w can be broken up..." to the end) into a single predicate logic statement. (You do not need to translate the lines before "w can be broken up..."). Since you are only translating the last two lines, treat L and w as constants in your translation (i.e., don't put quantifiers in front of them) You'll have to think about how to translate "can be broken up." [5 points]

(b) Negate the translation you wrote in part (a). 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. [3 points]

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

(d) Ponder the statement as a whole. So many quantifiers. You do not have to write anything for this part, but may find this pondering useful for attempting the extra credit. [0 points]

6. Some closure at the end of the quarter [16 points]

We say that a property of languages is "closed" under an operation if applying the operation to languages with the property must produce another language with the property. For example, being regular is closed under union – for any regular languages L_1, L_2, the language $L_1 \cup L_2$ is also regular.

When taking the complement, let the universe be Σ^*.

(a) Prove that being regular is closed under complement, that is: for any regular language L, $\overline{L}$ is also regular. Recall that we have multiple equivalent definitions of regular, some are better suited to this problem than others! [7 points]

(b) Prove that Σ^* is a regular language. [2 points]

(c) Prove that it is not the case that being irregular (i.e., not regular) is closed under both complement and union.[2] For this problem, you should use proof by contradiction. (Hint: our proof uses the result of part b!) [7 points]

[1] by which we mean every character of w appears in exactly one of x, y, z in order. For example, if $w = 111011$ then $x = 111$, $y = 011$, $z = \varepsilon$ or $x = 11$, $y = 10$, $z = 11$ would be valid, but $x = 101$, $y = 11$, $z = 1$ would not be valid.

[2] That is show $\neg$(closed under complement $\wedge$ closed under union). If irregular languages were closed under complement and union, then the complement of every irregular language would be irregular and the union of any two irregular languages would be irregular.

7. Buggy [9 points]

Consider the following claim. "Irregular languages are closed under intersection."

(a) Your friend writes the following proof. Their proof is incorrect. Identify the bug. State both the step where the biggest bug is, and why it is incorrect. [4 points]

Proof:
① We rephrase the original claim as "if L_1 and L_2 are irregular then $L_1 \cap L_2$ is irregular."
② We argue by contradiction. Suppose that if L_1 and L_2 are irregular, then $L_1 \cap L_2$ is regular.
③ Consider $L_1 = \{a^n b^n c^n : n \geq 0\}$ and $L_2 = \{a^n b^n c^n : n \geq 0\}$ (i.e. the same language).
④ We recall from class that L_1 (and thus L_2) is irregular.
⑤ Applying the statement in step ②, we get that $L_1 \cap L_2 = L_1$ must be regular.
⑥ But we know L_1 is irregular! That's a contradiction.
⑦ Thus we have irregular languages are closed under intersection.

(b) Is the claim true? Say "true" or "false" then prove your answer. [5 points]

8. 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." (See the statement in Question 5)

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 irrecular, 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 $\{a^{311}b^n c^m : n = 311+m\}$ 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. Extra Credit: Finishing Strong [0 points]

Let L be a regular language over the alphabet Σ. Define $\mathsf{ends}(L) = \{v \in \Sigma^* : \exists u \in \Sigma^* (uv \in L)\}$. Prove that $\mathsf{ends}(L)$ is regular.