Homework 8: Finite Automata; Fundamentals Review

Due Date: Friday December 5th
Late deadline: Saturday December 6th.

Note the unusual late deadline; you can use at most one late day on this homework assignment; this will make sure we can release solutions on Sunday December 7th (before the final on Monday Dec 8).

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 by Wednesday November 26. You can start to think about them now.

1. Happy Holidays, Garfield! [20 points]

Garfield is preparing for the holidays! Since he's in a festive mood, Garfield wants to set up a holiday tree and decorate it with bells! To make sure his bell tree doesn't fall over, Garfield wants to prove some facts about the tree's number of bells compared to its height. In this problem, we'll refer to the following definition of a `bellTree`:

Basis Step: A single node is a rooted bell tree: $\bullet$.
Recursive Step: If A, B, C, D are `bellTrees`, then $(\bullet, A, B, C, D)$ is also a `bellTree`.

We will also use the following recursively defined function that sums the values in a `bellTree`:

$$\text{bells}(\bullet) = 1$$
$$\text{bells}((\bullet, A, B, C, D)) = \text{bells}(A) + \text{bells}(B) + \text{bells}(C) + \text{bells}(D) \qquad \text{for arbitrary bellTrees } A, B, C, D$$

$$\text{height}(\bullet) = 0$$
$$\text{height}((\bullet, A, B, C, D)) = 1 + \max(\text{height}(A), \text{height}(B), \text{height}(C), \text{height}(D)) \qquad \text{for arbitrary bellTrees } A, B, C, D$$

Show that $1 \leq \text{bells}(T) \leq 4^{\text{height}(T)+1}$ for every `bellTree` T.

2. Sets of work over time [12 points]

(a) Let A, B be arbitrary sets. Show that $\mathcal{P}(A \cap B) = \mathcal{P}(A) \cap \mathcal{P}(B)$.

For this proof you **must** introduce an arbitrary element of a set X in order to prove that it is a subset of another set Y.

(b) Is it true that $\mathcal{P}(A \cup B) = \mathcal{P}(A) \cup \mathcal{P}(B)$ 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.

Notes on Accessing Grin

If you're having trouble accessing Grin, please read this pinned post.
Submission boxes on Grin will appear before Thanskgiving.

3. Context... doesn't matter? (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.

Caution: Grin can be very slow at autograding CFGs in particular. For this problem, your solutions should be relatively simple. All of our solutions use at most 2 nonterminal characters. If Grin times out while trying to grade your work, look for ways to simplify it.

(a) All strings of the form $\{a^{2i+2j}b^jc^i : i, j >= 0\}$.

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

(c) All binary strings that start and end with the same number of 1's

For example, 1100011 is in this language (it starts and ends with 2 1's), but 111011 is not (it starts with 3 1's and ends with 2 1's).
We say that a string like 111 is in the language because it starts with 3 1's, and it ends with 3 1's.
Additionally, if a string starts and ends with 0 1's, it is in the language.

Note that this is not equivalent to "strings of the form 1^kx1^k where x is any binary string". That language actually includes all binary strings (consider $k = 0$).

4. An exercise in determinism (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 odd number of 1's and end in a 0.

(b) Binary strings that start with 01 and have at most three 0's.

(c) Binary strings that either end in 111 or contain 00 **but not both!**

5. The power of free will (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 contain the substring 101 or the substring 111 (a string that contains both should be accepted).

(b) Binary strings that contain the substring 100, and every 1 is immediately followed by at least one 0.

(c) Binary strings that match the regular expression $(001 \cup 00101)^*(11)^*$.

6. Modulation [9 points]

This problem asks you to prove statements you haven't seen before. Even though the statements themselves are new, the <u>structure</u> 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.

Define the set

$$[a]_m = \{n : a \equiv n (\text{mod } m)\}$$

This is a definition that is a *bit* different to what we might have seen before. For one, this seems to be a set that takes in two parameters: a mod base, and an element!

As it turns out, this set is actually an example of an incredibly useful construction with results that show up all across analytic philosophy (PHIL 470) algorithmic analysis (CSE332/CSE421), and programming languages (CSE341 / CSE505), called an **equivalence class**. All of the elements of an equivalence class are equivalent!

We are going to prove some useful properties about this!

(a) Prove that for all $x \in \mathbb{Z}$ and $m \in \mathbb{Z}$, $x \in [x]_m$ [3 points] This property is known as *reflexivity*.
Hint: This proof will be proving a statement with *two* for all's!

(b) Prove that for all $a, b, m \in \mathbb{Z}$, if $a \in [b]_m$, then $b \in [a]_m$. [3 points]. With this, we prove that the equivalence classes have **symmetry** - that is, if a is related to b, then b is related to a

(c) Prove that for all $a, b, c, m \in \mathbb{Z}$, if $a \in [b]_m$ and $b \in [c]_m$, then $a \in [c]_m$. [3 points] This property is known as *transitivity*.

(d) Congratulations! You've now worked with equivalence class formally!

To give a formal definition, a **relation** is a set of ordered pairs (a, b) such that a is related to b. For example, we can treat the $<$ operator as a relation $\{(a, b) : a < b\}$!

An **equivalence relation** R is a relation that is **reflexive** ($\forall a[(a, a) \in R]$), **symmetric** ($\forall a, b[(a, b) \in R \rightarrow (b, a) \in R]$), and **transitive** ($\forall a, b, c[(a, b) \in R \wedge (b, c) \in R \rightarrow (a, c) \in R]$). An equivalence class (what we worked with!) is a subset of an equivalence relation, such that all of the elements are equivalent!

Ponder: which of those properties apply to $<$? What about $\leq$?
You do not have to write anything for this part.

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

Prove that the following language is not regular: The set of all strings over $\{0, 1, 2...9, \#\}$ of the form $x\#1^{(len(x))}$, with $x \in \{0, 1 \ldots, 9\}^*$.

For example

- 1#1 is in the language.
- 1492#1111 is in the language.
- 311###111 is not in the language (you can only have a single #).
- 4#1111 is not the language.

8. 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?