CS 103: Mathematical Foundations of Computing
Problem Set #7

August 9, 2019

All questions due Wednesday, August 14th at 3:00PM.

Note: Because this problem set is due on the last day of class, no late days may be used and no late submissions will be accepted. Sorry about that! On the plus side, we’ll release solutions as soon as the problem set comes due.

This final problem set is designed as a capstone of the past seven weeks in this course. We’ll be concluding some long-running threads as well as exploring our newfound intuitions for the landscape of computability.

Before attempting the last problem on this problem set, we strongly recommend reading over the Guide to the Lava Diagram that is available on the course website, which provides a ton of extra background that you might find useful here.

As always, please feel free to drop by office hours or ask questions on Piazza if you have any questions. We’d be happy to help out.

Good luck, and have fun!
1 A Visit to the Fourth Dimension

This question explores a class of graphs called the \textbf{hypercube graphs} whose nodes and edges correspond to the vertices and edges of squares, cubes, tesseracts, and even more exotic higher-dimensional objects. Hypercube graphs have a ton of applications throughout CS theory and if you study high-performance computing, parallel programming, binary encoding schemes, or combinatorics, you’ll likely see them make an appearance.

We’ll begin this exploration by asking you to translate a statement into first-order logic.

i. Given the predicates
   \[ x \in y, \] which states that \( x \) is an element of \( y \), and
   \[ x \notin y, \] which states that \( x \) is not an element of \( y \),
   along with the constant symbols \( S \) and \( T \), which represent some two sets, translate the statement \( |S\Delta T| = 1 \) into first-order logic.

In zero dimensions, we have a point. In one dimension, we have a line. In two dimensions, we have a square. In three dimensions, we have a cube. And in four dimensions, we have a \textit{tesseract}.

How do we even think about what a tesseract is? For this, we can turn to graph theory. The \textbf{hypercube graph of order} \( k \), denoted \( Q_k \), is defined as follows:

- The nodes of \( Q_k \) are the elements of \( \wp([k]) \). (Refer to Problem Set Three for the definition of \( [k] \).)
- There is an edge between a pair of nodes \( S \) and \( T \) if (and only if) \( |S\Delta T| = 1 \).

That definition might seem like a mouthful, but it makes a lot more sense once you have a visual intuition.

ii. Draw the graphs \( Q_0, Q_1, Q_2, \) and \( Q_3 \). Label each node with the set it corresponds to. Then, explain why \( Q_0 \) is a good approximation of a point, \( Q_1 \) is a good approximation of a line, \( Q_2 \) is a good approximation of a square, and \( Q_3 \) is a good approximation of a cube.

\textit{You may need to shuffle around the nodes of} \( Q_3 \text{ } \text{to see why it’s a good proxy for a cube}.}

Though we won’t make you draw this one out, the tesseract is modeled by \( Q_4 \). There are many ways to draw \( Q_4 \); one of them is shown to the right. We’ve omitted the labels on the nodes for convenience. (Whoa! It’s an \( \{8/3\} \) star inside another \( \{8/3\} \) star inside a \( \{8/1\} \) star!)

Isn’t that just wild? You took this dense mathematical definition that we gave you and extracted from that a new intuition for shapes in the fourth dimension and beyond! Imagine if we had given you this problem on the first day of class and think about how far you’ve come since then.
2 Equivalence Classes and Regular Languages, Part Two

On Problem Set Six, you explored the \textit{indistinguishability} relation for $L$, denoted $\equiv_{L}$, defined as

\[ x \equiv_{L} y \iff \forall w \in \Sigma^*. (xw \in L \iff yw \in L). \]

You specifically proved that for any language $L$, the relation $\equiv_{L}$ is an equivalence relation and that any DFA for $L$ must have at least $I(\equiv_{L})$ states. In this problem, you’re going to prove an amazing result:

**Theorem:** If $L$ is a language where $I(\equiv_{L})$ is finite, then $L$ is regular.

In other words, if you know absolutely nothing about a language other than there are finitely many equivalence classes of the $\equiv_{L}$ relation, then somewhere out there, there must be a DFA for $L$!

Let $L$ be an arbitrary language over some alphabet $\Sigma$ where $I(\equiv_{L})$ is finite. We are going to prove that $L$ is regular by defining a 5-tuple $(Q, \Sigma, \delta, q_{0}, F)$ for this language $L$. The key insight behind this proof is how to choose $Q$. Specifically, we will choose $Q$ to be the set of equivalence classes of $\equiv_{L}$:

\[ Q = \{[w]_{\equiv_{L}} \mid w \in \Sigma^* \}. \]

It might seem strange to have the states of a DFA be sets, but then again, you’ve seen something like this before. When we worked through the subset construction in lecture, we created a DFA whose states literally were sets of states of some particular NFA.

i. Explain why $Q$ is finite. This should take you at most a sentence or two.

We now need to figure out how to pick a start state and wire up our transitions. Our goal will be to define $q_{0}$ and $\delta$ so that our DFA has the following property: if you run $w$ through this DFA, the state you end up in corresponds to $[w]_{\equiv_{L}}$. It turns out that choosing $q_{0}$ and $\delta$ as follows makes this work:

\[ q_{0} = [\varepsilon]_{\equiv_{L}} \quad \delta([x]_{\equiv_{L}}, a) = [xa]_{\equiv_{L}}. \]

Of course, you shouldn’t take our word for it. You should prove that these choices make everything work!

ii. Prove that for any string $w \in \Sigma^*$, we have $\delta^{*}(w) = [w]_{\equiv_{L}}$.

\textit{Need a refresher on the definition of $\delta^{*}$? Check Problem Set Six. Perhaps the proof strategy you used there to reason about $\delta^{*}$ could come in handy here as well.}

To seal the deal, we need to choose our set of accepting states. We’ll define $F$ as follows:

\[ F = \{[w]_{\equiv_{L}} \mid \exists x \in [w]_{\equiv_{L}}. x \in L \}. \]

In other words, $F$ is the set of all equivalence classes containing at least one string in $L$.

iii. On Problem Set Six, you saw that we can formally define $L(D) = \{ w \in \Sigma^* \mid \delta^{*}(w) \in F \}$. Prove that with this choice of $F$, we have $L(D) = L$.

\textit{There is a ton of formal notation here, but at the end of the day, this question is just asking you to prove that two sets are equal. Think way back to Problem Set One. What’s the easiest way to do this?}

\textit{Your proof should use the formal definitions provided here and not higher-level concepts like “the DFA accepts $w$” or “run the DFA on $w$.” Perhaps a result from Problem Set Six would be useful here?}

By combining the two theorems you’ve explored about indistinguishability – the one you proved last time, and the one from above – we get this fundamental result:

**Theorem (Myhill-Nerode):** A language $L$ is regular if and only if $I(\equiv_{L})$ is finite. Furthermore, if $I(\equiv_{L})$ is finite, the smallest possible DFA for $L$ has exactly $I(\equiv_{L})$ states.

This result formalizes the intuition we’ve had about regular languages corresponding to problems you can solve with only finite memory. The “memory” you need corresponds to remembering which equivalence class the string you’ve seen so far happens to fall into. If you talk to CS theory folk and mention the Myhill-Nerode theorem,” they’ll assume you’re talking about the above theorem! The version we saw in lecture is just a special case of this more general one.
3 The Star-Drawing Saga, Part V: The Grand Finale

A powerful technique in any mathematician’s toolbox is the concept of dual objects. The formal definition of a dual object varies by context, but the key idea is to take an object made from two different parts, swap the meanings of those parts, and end up with a new object related to the one we started with.

Every star \{p/s\} has an associated dual star \{s/p\}. In other words, starting with a \(p\)-pointed star with a step size of \(s\), we create a new star with \(s\) points and a step size of \(p\). That new star might have a step size that’s much, much larger than its number of points, but that’s nothing to worry about. It’s still perfectly mathematically valid! For example, here’s the \{12/5\} star and its dual, the \{5/12\} star. You might notice that the \{5/12\} star looks a lot like the \{5/2\} star – more on that later – but it is indeed \{5/12\}.

On Problem Set Two, you proved that \{p/s\} is simple if and only if there is an integer \(t\) where \(1 \equiv_p st\). Technically, you only proved this theorem in the case where \(p > 0\) and \(t\) is a natural number, but for the purposes of this problem, we’re going to extend this theorem and assume it holds for all natural numbers \(p\) (not just the positive ones) and all integers \(t\) (not just the natural numbers).

i. Using the above theorem, prove that if \(p\) and \(s\) are natural numbers, then \{p/s\} is simple if and only if its dual star is simple.

You might have noticed that the \{5/12\} star looks a lot like the \{5/2\} star. To formalize why this is, we’ll need to use the division algorithm, which, despite the CS-sounding name, isn’t actually an algorithm. It’s the following fact about natural numbers:

**The Division Algorithm:** For any natural numbers \(m\) and \(n\) where \(n \neq 0\), there exist unique natural numbers \(q\) and \(r\) where \(m = nq + r\) and \(r < n\).

This is the fancy mathematical way of saying “you can divide \(m\) by \(n\) to get a quotient \((q)\) and a remainder \((r)\), and the remainder \(r\) is always less than \(n\).” I find it amusing that we have such a weighty name for something so simple, but hey, that’s the convention.

ii. Let \(p\) and \(s\) be natural numbers where \(s > 0\) and let \(r\) be the remainder you get via the division algorithm when dividing \(p\) by \(s\). Explain, intuitively, why \{s/p\} and \{s/r\} are the same star.

**Note that the roles of the variables \(s\) and \(p\) are reversed from their normal arrangement in this problem.**

iii. Let \(p\) and \(s\) be natural numbers where \(s > 0\) and let \(r\) be the remainder you get via the division algorithm when dividing \(p\) by \(s\). Prove that \{p/s\} is simple if and only if \{s/r\} is simple.

*We’re looking for a rigorous proof here involving modular congruence, so while the intuition from part (ii) might come in handy here, you still need to use a rigorous proof.*

iv. Using your result from part (iii), prove or disprove: \{4, 798, 014/423, 193\} is simple.

(This question is continued on the next page.)
And now the grand finale. Under what circumstances is a star simple? You know from Problem Set Three that if \( \{p/s\} \) is a simple star, then \( p \) and \( s \) are coprime. And based on what you’ve seen above, you now have enough to prove the converse of this statement.

v. Prove by induction on \( s \) that if \( p \) and \( s \) are coprime natural numbers, then \( \{p/s\} \) is a simple star. Feel free to use the fact that the only natural number coprime with 0 is 1.

Think about what you just did in part (iv) where you took this question of whether or not a star \( \{p/s\} \) is simple and converted it into a question of whether or not a smaller star \( \{s/r\} \) is simple. We can guarantee that \( r < s \), but we have no idea how much smaller \( r \) is going to be than \( s \).

There are a lot of variables flying around here so go slowly through the setup. What is your inductive hypothesis? What \( P(k+1) \) are you trying to prove? Importantly, your inductive hypothesis is an implication, so what do you need to prove in order to apply the inductive hypothesis?

You’ve come quite a long way since you first tried drawing a simple 7-pointed star! You’ve explored modular arithmetic, coprimality, and now the division algorithm and induction!

There are so many other questions you could ask about stars. Although \( \{8/2\} \) isn’t a simple star, you can draw an eight-pointed star as two copies of the \( \{4/1\} \) star. Is there some rule about how many copies of simple stars you’d need for a general \( \{p/s\} \), or what those simple stars would be? If you keep on exploring these questions – which we hope you do! – you’ll quickly run into concepts from group theory, number theory, and abstract algebra. For more on this, check out Math 120, Math 152, and CS255!
4 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 twelve 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 – the purpose of this problem is to help you build a better intuition for what makes a language regular, \( R \), \( \text{RE} \), or none of these.

We strongly recommend reading over the Guide to the Lava Diagram before starting this problem.

To submit your answers, edit the file LavaDiagram.h in the src/ directory of the starter files for this problem set.

\[
\begin{align*}
1. \, & \Sigma^* \\
2. \, & L_D \\
3. \, & \{ \epsilon^n \mid n \in \mathbb{N} \} \\
4. \, & \{ \epsilon^n \mid n \in \mathbb{N} \text{ and is a multiple of 137} \} \\
5. \, & \{ 1^n1^m \geq 1^{n+m} \mid m, n \in \mathbb{N} \} \\
6. \, & \{ \langle M \rangle \mid M \text{ is a Turing machine and } L(M) \neq \emptyset \} \\
7. \, & \{ \langle M \rangle \mid M \text{ is a Turing machine and } L(M) = \emptyset \} \\
8. \, & \{ \langle M \rangle \mid M \text{ is a Turing machine and } L(M) = L_D \} \\
9. \, & \{ \langle M, n \rangle \mid M \text{ is a TM, } n \in \mathbb{N}, \text{ and } M \text{ accepts all strings in its input alphabet of length at most } n \} \\
10. \, & \{ \langle M, n \rangle \mid M \text{ is a TM, } n \in \mathbb{N}, \text{ and } M \text{ rejects all strings in its input alphabet of length at most } n \} \\
11. \, & \{ \langle M, n \rangle \mid M \text{ is a TM, } n \in \mathbb{N}, \text{ and } M \text{ loops on all strings in its input alphabet of length at most } n \} \\
12. \, & \{ \langle M_1, M_2, M_3, w \rangle \mid M_1, M_2, \text{ and } M_3 \text{ are TMs, } w \text{ is a string, and at least two of } M_1, M_2, \text{ and } M_3 \text{ accept } w. \} 
\end{align*}
\]
Optional Fun Problem: Back to the Fourth Dimension (Extra Credit)

Picking up right where we left off in Problem 1, we saw that the tesseract is modeled by $Q_4$. Here is again one way of drawing $Q_4$. It’s hard to look at that picture and to imagine what it would “feel like” to hold it in your hand. But by using automorphisms, which you saw in the Problem Set 4, we can get a better intuition.

From your lived experience you know that if you pick up a cube, you can twist and turn it around and it has all sorts of symmetries. From a graph theory perspective, that would lead us to think that $Q_3$ should have many automorphisms, since each automorphism corresponds to a symmetry. So here’s a question: does $Q_4$ have the same sort of symmetries that you’d expect of a cube or a square?

A graph $G = (V,E)$ is called node-symmetric if, for any two nodes $u,v \in V$, there is an automorphism $\sigma$ of $G$ where $\sigma(u) = v$. Intuitively, this means that all the nodes in $G$ “look the same,” since for any pair of nodes there’s a symmetry of the graph (an automorphism) that makes the first node look like the second.

Prove that for every natural number $k$, the graph $Q_k$ is node-symmetric. You can assume that symmetric difference is commutative ($S \Delta T = T \Delta S$) and associative ($(S \Delta T) \Delta R = S \Delta (T \Delta R)$).

This result is beautiful and ties together concepts from throughout the course. We’d like to encourage everyone to at least take a stab at solving it. Here’s some guidance to get started:

This problem is much, much easier to solve if you take the time to work through some examples. Can you find an automorphism of $Q_2$ that maps $\emptyset$ to $\{0,1\}$? To answer that question, try connecting it back to an actual square and its symmetries, see what happens to the corners, and see if you can use that to define an automorphism – there are two different choices here, one of which has a very simple interpretation. Then, find an automorphism of $Q_3$ that maps $\{1\}$ to $\{0,2\}$ by thinking about what that means in terms of symmetries of a square. You should aim to find a general pattern here before moving on.

Once you’ve found a pattern of what these automorphisms look like, find a general formula for an automorphism $\sigma$ of $Q_k$ that maps some set $S$ to some set $T$. It should be pretty short and shouldn’t require a piecewise definition. Then, write down a list of everything you need to prove in order to show that $\sigma$ is indeed an automorphism; you did this on Problem Set Four, so perhaps that would be useful as a starting point. Then, go prove all those properties. In doing so, look back to Problem Set One or Problem Set Two. Perhaps there are some nice results from there you could use here?

Oh, and there’s no need to use induction here. 😊

Grand Challenge Problem: $P \not\equiv NP$ (Worth an A+, $1,000,000, and a Ph.D)

Prove or disprove: $P = NP$.

Take fifteen minutes and try this. Seriously. And if you can’t crack this problem, feel free to submit your best effort, or the silliest answer you can think of.