

## Problem Set 7

---

What can you do with regular expressions? What are the limits of regular languages? In this problem set, you'll explore the answers to these questions along with their practical consequences.

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

Good luck, and have fun!

**Due Friday, March 2<sup>nd</sup> at 2:30PM**

## Problem One: Designing Regular Expressions

Below are a list of alphabets and languages over those alphabets. For each language, write a regular expression for that language.

**Please use our online tool to design, test, and submit your regular expressions. Typed or handwritten solutions will not be accepted.** To use it, visit the CS103 website and click the “Regex Editor” link under the “Resources” header. As before, if you submit in a pair, please make a note in your GradeScope submission of which partner submitted your answers to this question so that we know where to look. Also, as a reminder, please test your submissions thoroughly, since we'll be grading them with an autograder.

- i. Let  $\Sigma = \{a, b\}$  and let  $L = \{ w \in \Sigma^* \mid w \text{ does not contain } ba \text{ as a substring} \}$ . Write a regular expression for  $L$ .
- ii. Let  $\Sigma = \{a, b\}$  and let  $L = \{ w \in \Sigma^* \mid w \text{ does not contain } bb \text{ as a substring} \}$ . Write a regular expression for  $L$ .

*You may want to draw out some sample strings in this language and look for a pattern.*

- iii. Suppose you are taking a walk with your dog on a leash of length two. Let  $\Sigma = \{y, d\}$  and let  $L = \{ w \in \Sigma^* \mid w \text{ represents a walk with your dog on a leash where you and your dog both end up at the same location} \}$ . For example, we have  $yydddyy \in L$  because you and your dog are never more than two steps apart and both of you end up four steps ahead of where you started; similarly,  $ddydy \in L$ . However,  $yyyyddd \notin L$ , since halfway through your walk you're three steps ahead of your dog;  $ddy \notin L$ , because your dog ends up two steps ahead of you; and  $ddydyyy \notin L$ , because at one point your dog is three steps ahead of you. Write a regular expression for  $L$ .
- iv. Let  $\Sigma = \{a, b\}$  and let  $L = \{ w \in \Sigma^* \mid w \neq ab \}$ . Write a regular expression for  $L$ .
- v. Let  $\Sigma = \{M, D, C, L, X, V, I\}$  and let  $L = \{ w \in \Sigma^* \mid w \text{ is number less than 2,000 represented in Roman numerals} \}$ . For example,  $CMXCIX \in L$ , since it represents the number 999, as are the strings  $L$  (50),  $VIII$  (8),  $DCLXVI$  (666),  $CXXXVII$  (137),  $CDXII$  (412), and  $MDCXVIII$  (1,618). However, we have  $VIIII \notin L$  (you'll never have four I's in a row; use  $IX$  or  $IV$  instead), that  $MM \notin L$  (it's a Roman numeral, but it's for 2,000, which is too large), that  $VX \notin L$  (this isn't a valid Roman numeral), and that  $IM \notin L$  (the notation of using a smaller digit to subtract from a larger one only lets you use  $I$  to prefix  $V$  and  $X$ , or  $X$  to prefix  $L$  and  $C$ , or  $C$  to prefix  $D$  and  $M$ ). The Romans didn't have a way of expressing the number 0, so to make your life easier we'll say that  $\epsilon \in L$  and that the empty string represents 0. (Oh, those silly Romans.) Write a regular expression for  $L$ .

(As a note, we're using the “standard form” of Roman numerals. You can see a sample of numbers written out this way via [this link](#).)

## Problem Two: Finite and Cofinite Languages

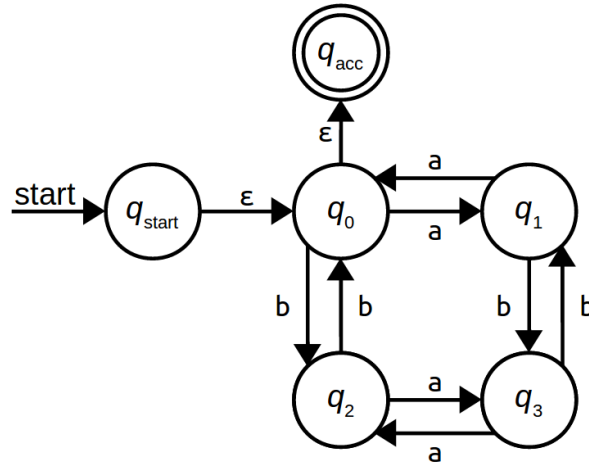
A language  $L$  is called *finite* if  $L$  contains finitely many strings (that is,  $|L|$  is a natural number.) A language  $L$  is *cofinite* if its complement is a finite language; that is,  $L$  is cofinite if  $|\bar{L}|$  is a natural number.

- i. Given a finite language  $L$ , explain how to write a regular expression for  $L$ . Briefly justify your answer; no formal proof is necessary. This shows that all finite languages are regular.
- ii. Look up the *trie* data structure on Wikipedia. Explain, in your own words, how a trie can be thought of as a finite automaton. This gives another view on why finite languages are regular.
- iii. Prove that any cofinite language is regular.

### Problem Three: State Elimination

The state elimination algorithm gives a way to transform a finite automaton (DFA or NFA) into a regular expression. It's a really beautiful algorithm once you get the hang of it, so we thought that we'd let you try it out on a particular example.

Let  $\Sigma = \{a, b\}$  and let  $L = \{ w \in \Sigma^* \mid w \text{ has an even number of } a\text{'s and an even number of } b\text{'s} \}$ . Below is a finite automaton for  $L$  that we've prepared for the state elimination algorithm by adding in a new start state  $q_{start}$  and a new accept state  $q_{acc}$ :



We'd like you to use the state elimination algorithm to produce a regular expression for  $L$ .

- i. Run two steps of the state elimination algorithm on the above automaton. Specifically, first remove state  $q_1$ , then remove state  $q_2$ . Show your result at this point.

*Go slowly. Remember that to eliminate a state  $q$ , you should identify all pairs of states  $q_{in}$  and  $q_{out}$  where there's a transition from  $q_{in}$  to  $q$  and from  $q$  to  $q_{out}$ , then add shortcut edges from  $q_{in}$  to  $q_{out}$  to bypass state  $q$ . Remember that  $q_{in}$  and  $q_{out}$  can be the same state. If you've done everything right at the end of this stage, none of the transitions you have at this point should have Kleene stars in them.*

- ii. Finish the state elimination algorithm. What regular expression do you get for  $L$ ?

*You should start seeing Kleene stars appearing as you remove the remaining states.*

- iii. Without making reference to the original automaton given above, give an intuitive explanation for how the regular expression you found in part (ii) works.

## Problem Four: Distinguishable Strings

The Myhill-Nerode theorem is one of the trickier and more nuanced theorems we've covered this quarter. This question explores what the theorem means and, importantly, what it *doesn't* mean.

Let  $\Sigma = \{a, b\}$  and let  $L = \{ w \in \Sigma^* \mid |w| \text{ is even} \}$ .

- i. Show that  $L$  is a regular language.

*There are lots of ways you can go about doing this. Pick whatever is easiest.*

- ii. Prove that there is an infinite set  $S \subseteq \Sigma^*$  where there are infinitely many pairs of distinct strings  $x, y \in S$  such that  $x \not\equiv_L y$ .

*An analogy that might help: do you understand the difference between "a group of six people, any two of which are friends" versus "a group of six people, of which six pairs of those people are friends?"*

- iii. Prove that there is *no* infinite set  $S \subseteq \Sigma^*$  where *all* pairs of distinct strings  $x, y \in S$  satisfy  $x \not\equiv_L y$ .

The distinction between parts (ii) and (iii) is important for the Myhill-Nerode theorem. A language is nonregular not if you can find infinitely many pairs of distinguishable strings, but rather if you can find infinitely many strings that are all *pairwise* distinguishable.

## Problem Five: At Least Three Point Five

The Myhill-Nerode theorem is a powerful tool for finding nonregular languages, but it can take some adjusting to get used to.

Let  $\Sigma = \{a, b\}$  and consider the following language:

$$L = \{ w \in \Sigma^* \mid \text{there are at least as many } a\text{'s as } b\text{'s in } w \}.$$

Below is a purported proof that  $L$  is not a regular language.

**Theorem:**  $L$  is not a regular language.

**Proof:** Consider the set  $S = \{ a^n \mid n \in \mathbb{N} \}$ . This set is infinite because it contains one string for each natural number. We will prove that any two distinct strings in  $S$  are distinguishable relative to  $L$ . To do so, consider any distinct strings  $a^m, a^n \in S$ , and assume without loss of generality that  $m > n$ . Then  $b^m a^m \in L$  because this string contains the same number of  $a$ 's and  $b$ 's, but  $b^m a^n \notin L$  because it contains  $m$   $b$ 's and  $n$   $a$ 's and  $m > n$ . Therefore, we see that  $a^m \not\equiv_L a^n$ . This means that  $S$  is an infinite set of strings that are pairwise distinguishable relative to  $L$ . Therefore, by the Myhill-Nerode theorem,  $L$  is not regular. ■

Unfortunately, this proof is incorrect.

- i. What's wrong with this proof? Be as specific as possible.

*The best way to identify a flaw in a proof is to point to a specific claim that's being made that's not true or not properly substantiated and to explain why.*

- ii. Is  $L$  a regular language? If so, show why it's regular. If not, prove that  $L$  is not a regular language.

*Think about the intuition behind what makes a language regular or not regular. If your intuition tells you that this language is regular, then use that intuition to guide your justification. If your intuition tells you that this language is not regular, use that intuition to guide your disproof. And if you don't have an intuition, take a look back at the lecture slides and feel free to stop by office hours!*

## Problem Six: Balanced Parentheses

Consider the following language over  $\Sigma = \{ (, ) \}$ :

$$L_1 = \{ w \in \Sigma^* \mid w \text{ is a string of balanced parentheses} \}$$

For example, we have  $() \in L_1$ ,  $(( )) \in L_1$ ,  $((())()) \in L_1$ ,  $\epsilon \in L_1$ , and  $((()))((())()) \in L_1$ , but  $)() \notin L_1$ ,  $(( \notin L_1$ , and  $((())) \notin L_1$ . This question explores properties of this language.

- i. Prove that  $L_1$  is not a regular language. One consequence of this result – which you don't need to prove – is that most languages that support some sort of nested parentheses, such as most programming languages and HTML, aren't regular and so can't be parsed using regular expressions.

*As with all problems involving nonregular languages, proceed with this one in stages. First, ask yourself: if you were reading an input string from left to right, what information would you necessarily have to keep track of? The Myhill-Nerode theorem asks you to find an infinite set of strings that are all pairwise distinguishable, so try creating an infinite set of strings, one for each possible value that this information could take on.*

Let's say that the *nesting depth* of a string of balanced parentheses is the maximum number of unmatched open parentheses at any point inside the string. For example, the string  $((()))$  has nesting depth three, the string  $((())())$  has nesting depth two, and the string  $\epsilon$  has nesting depth zero.

Consider the language  $L_2 = \{ w \in \Sigma^* \mid w \text{ is a string of balanced parentheses and } w \text{'s nesting depth is at most four} \}$ . For example,  $((())) \in L_2$ ,  $((())()) \in L_2$ , and  $((())())((())()) \in L_2$ , but  $((()))((())) \notin L_2$  because although it's a string of balanced parentheses, the nesting goes five levels deep.

- ii. Design a DFA for  $L_2$ , showing that  $L_2$  is regular. A consequence of *this* result is that while you can't parse all programs or HTML with regular expressions, you can parse programs with low nesting depth or HTML documents without deeply-nested tags using regexes. **Please submit this DFA using the DFA editor on the course website** and tell us on GradeScope who submitted it.
- iii. Look back at your proof from part (i) of this problem. Imagine that you were to take that exact proof and blindly replace every instance of " $L_1$ " with " $L_2$ ." This would give you a (incorrect) proof that  $L_2$  is nonregular (which we know has to be wrong because  $L_2$  is indeed regular.) Where would the error be in that proof? Be as specific as possible.

*Again, you should be able to point at a specific spot in the proof that contains a logic error and explain exactly why the statement in question is not true or not supported by the preceding statements. If you can't do this, it likely means you have an error in your proof from part (i)!*

- iv. Intuitively, regular languages correspond to problems that can be solved using only finite memory. Using this intuition and without making reference to DFAs, NFAs, or the Myhill-Nerode theorem, explain why  $L_1$  is nonregular while  $L_2$  is regular.

## Problem Seven: Tautonyms

A *tautonym* is a word that consists of the same string repeated twice. For example, “caracara” and “dikdik” are all tautonyms (the first is a bird, the second the cutest animal you'll ever see), as is “hotshots” (people who aren't very fun to be around). Let  $\Sigma = \{a, b\}$  and consider the following language:

$$L = \{ ww \mid w \in \Sigma^* \}$$

This is the language of all tautonyms over  $\Sigma$ . Below is an *incorrect* proof that  $L$  is not regular:

**Proof:** Let  $S = \{ a^n \mid n \in \mathbb{N} \}$ . This set is infinite because it contains one string for each natural number. We claim that any two strings in  $S$  are distinguishable relative to  $L$ . To see this, consider any two distinct strings  $a^m$  and  $a^n$  in the set  $S$ , where  $m \neq n$ . Then  $a^m a^m \in L$  but  $a^n a^m \notin L$ , so  $a^m \not\equiv_L a^n$ . This means that  $S$  is an infinite set of strings that are pairwise distinguishable relative to  $L$ . Therefore, by the Myhill-Nerode theorem,  $L$  is not regular. ■

- i. What's wrong with this proof? Be as specific as possible.

*This one is subtle, and it's not the same error that you identified earlier. As before, point out a specific claim that is made that isn't true or isn't supported by the preceding statements and justify why.*

- ii. Although the above proof is incorrect, the language  $L$  isn't regular. Prove this.

*The most common mistake we see people make in part (ii) of this problem is to make the exact same sort of mistake that they identified in part (i) of this problem (oops!) So go slowly here and double-check your proof to make sure it's airtight, especially in light of what you saw in part (i). You may want to take a sentence or two to precisely articulate why each claim that you're making is true, since as you saw above it's easy for an erroneous statement to slip in under the radar if you're not careful!*

## Problem Eight: State Lower Bounds

The Myhill-Nerode theorem we proved in lecture is actually a special case of a more general theorem about regular languages that can be used to prove lower bounds on the number of states necessary to construct a DFA for a given language.

- i. Let  $L$  be a language over  $\Sigma$ . Suppose there's a *finite* set  $S$  such that any two distinct strings  $x, y \in S$  are distinguishable relative to  $L$  (that is,  $x \not\equiv_L y$  for any two strings  $x, y \in S$  where  $x \neq y$ .) Prove that any DFA for  $L$  must have at least  $|S|$  states. (You sometimes hear this referred to as *lower-bounding* the size of any DFA for  $L$ .)

According to old-school Twitter rules, all tweets need to be 140 characters or less. Let  $\Sigma$  be the alphabet of characters that can legally appear in a tweet and consider the following language:

$$TWEETS = \{ w \in \Sigma^* \mid |w| \leq 140 \}.$$

This is the language of all legal tweets, assuming the empty string is a legal tweet. The good news is that this language is regular. The bad news is that any DFA for it has to be pretty large.

- ii. Using your result from part (i), prove that any DFA for  $TWEETS$  must have at least 142 states.

*It might be easier to tackle this problem if you consider replacing 140 and 142 with some smaller numbers (say, 2 and 4) to build up an intuition. And work backwards – what will you need to do to invoke part (i)?*

- iii. Refer back to the formal, 5-tuple definition of a DFA given in Problem Set Six. Define an 142-state DFA for  $TWEETS$  using the formal 5-tuple definition. Briefly explain how your DFA works. No formal proof is necessary.

*Again, this might be a lot easier to do if you first reduce 140 and 142 to 2 and 4, respectively, and see what you come up with. Start by drawing out what the DFA would look like, then think about how you'd formalize your idea as a 5-tuple.*

Your results from parts (ii) and (iii) collectively establish that the smallest possible DFA for  $TWEETS$  has exactly 142 states. This approach to finding the smallest object of some type – using some theorem to prove a lower bound (“we need at least this many states”) combined with a specific object of the given type (“we certainly can’t do worse than this”) is a common strategy in discrete mathematics, algorithm design, and computational complexity theory. If you take classes like CS161, CS254, etc., you’ll likely see similar sorts of approaches!

## Problem Nine: Closure Properties Revisited

When building up the regular expressions, we explored several closure properties of the regular languages. This problem explores some of their nuances.

The regular languages are closed under complementation: If  $L$  is regular, so is  $\bar{L}$ .

- i. Prove or disprove: the *nonregular* languages are closed under complementation.

*In other words, if you have a nonregular language  $L$ , is  $\bar{L}$  necessarily nonregular?*

The regular languages are closed under union: If  $L_1$  and  $L_2$  are regular, so is  $L_1 \cup L_2$ .

- ii. Prove or disprove: the *nonregular* languages are closed under union.

We know that the union of any two regular languages is regular. Using induction, we can show that the union of any finite number of regular languages is also regular. As a result, we say that the regular languages are closed under *finite union*.

An infinite union is the union of infinitely many sets. Formally speaking, imagine that you have a sequence of infinitely many sets  $S_0, S_1, S_2, \dots$ , one for each natural number. Then the *infinite union* of those sets is defined as follows:

$$\bigcup_{i=0}^{\infty} S_i = \{ w \mid \exists n \in \mathbb{N}. w \in S_n \}.$$

For example, if  $L$  is a language, then  $L^*$ , which we defined as  $\{ w \in \Sigma^* \mid \exists n \in \mathbb{N}. w \in L^n \}$ , can be thought of as the infinite union

$$\bigcup_{i=0}^{\infty} L^i.$$

Similarly, the set of all rational numbers,  $\mathbb{Q}$ , can be thought of as

$$\mathbb{Q} = \bigcup_{i=0}^{\infty} \left\{ \frac{x}{i+1} \mid x \in \mathbb{Z} \right\}.$$

- iii. Prove or disprove: the regular languages are closed under infinite union.

*In other words, if you have an infinite list of regular languages  $L_0, L_1, L_2, \dots$ , is their infinite union necessarily a regular language?*

## Optional Fun Problem: Generalized Fooling Sets (1 Point Extra Credit)

In Problem Eight, you used distinguishability to lower-bound the size of DFAs for a particular language. Unfortunately, distinguishability is not a powerful enough technique to lower-bound the sizes of NFAs. In fact, it's in general quite hard to bound NFA sizes; there's a \$1,000,000 prize for anyone who finds a polynomial-time algorithm that, given an arbitrary NFA, converts it to the smallest possible equivalent NFA!

Although it's generally difficult to lower-bound the sizes of NFAs, there are some techniques we can use to find lower bounds on the sizes of NFAs. Let  $L$  be a language over  $\Sigma$ . A *generalized fooling set* for  $L$  is a set  $\mathcal{F} \subseteq \Sigma^* \times \Sigma^*$  is a set with the following properties:

- For any  $(x, y) \in \mathcal{F}$ , we have  $xy \in L$ .
- For any distinct pairs  $(x_1, y_1), (x_2, y_2) \in \mathcal{F}$ , we have  $x_1y_2 \notin L$  or  $x_2y_1 \notin L$  (this is an inclusive OR.)

Prove that if  $L$  is a language and there is a generalized fooling set  $\mathcal{F}$  for  $L$  that contains  $n$  pairs of strings, then any NFA for  $L$  must have at least  $n$  states.