Solutions for Week Eight

Problem One: Designing Regular Expressions

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

i. Let $\Sigma = \{a, b, c\}$ and let $L = \{ w \in \Sigma^* \mid w \text{ ends in cab} \}$. Write a regular expression for $L$.

One option is

$$\Sigma^* \text{cab}$$

This matches any string that begins with some number of characters of any type, then ends with cab. Isn't that easier than the NFA?

ii. Let $\Sigma = \{a, b\}$ and let $L = \{ w \in \Sigma^* \mid w \neq \epsilon \text{ and the first and last character of } w \text{ are the same} \}$. Write a regular expression for $L$.

Here's one option:

$$a \cup b \cup a\Sigma^*a \cup b\Sigma^*b$$

This says "either match the single characters $a$ and $b$, or match something that starts and ends with $a$ with any number of characters in the middle, or match something that starts and ends with $b$ with any number of characters in the middle."

You can condense this a bit using the $?$ operator:

$$a(\Sigma^*a)? \cup b(\Sigma^*b)?$$

This means "if you see an $a$, either you're done, or you're going to see some number of characters and then another $a$" (and the same for $b$.)

iii. Let $\Sigma = \{a, b\}$ and let $L = \{ w \in \Sigma^* \mid w \text{ contains two } b's \text{ separated by exactly five characters} \}$. Write a regular expression for $L$.

Here's one option:

$$\Sigma^*b\Sigma^5b\Sigma^*$$

This means "match any number of characters, then read a $b$, five characters, and another $b$, then as much as you'd like after that."
iv. Let Σ = {a, b} and let \( L = \{ w \in \Sigma^* \mid w \text{ is a nonempty string whose characters alternate between a's and b's} \} \). Write a regular expression for \( L \).

One possibility is
\[
a(ba)^*b? \cup b(ab)^*a?
\]

If the characters alternate, they either start with an a or they start with a b. If they begin with a, then we'll then have some number of copies of the string ba, optionally ending with a final b. If they begin with a b, then we'll then have some number of copies of the string ab, optionally ending with a final a.

v. Let Σ = {a, b, c} and let \( L = \{ w \in \Sigma^* \mid w \text{ contains every character in } \Sigma \text{ exactly once} \} \). Write a regular expression for \( L \).

Alas, there's no easy way to write this one because those characters can appear in any order.

Here's one option:
\[
abc \cup acb \cup bac \cup bca \cup cab \cup cba
\]

Why we asked this question: This question was designed to show off some common patterns and techniques in the design of regular expressions. Part (i) shows off how regexes match patterns at the beginning or end of a string (namely, by inserting \( \Sigma^* \) to cover prefixes or suffixes.) Part (ii) explores how regular expressions keep track of information like which character was used; namely, by repeating different patterns in different combinations. Part (iii) was designed to get you to see how a language looks when expressed in different formats: PS6 shows off an NFA for the language, PS7 asks you to prove there's no simple DFA for the language, and this problem shows that regexes nicely capture this particular language. Part (iv) shows off an edge case whereby you sometimes have to insert a suffix to a regex to capture the idea that a pattern might end abruptly. Finally, part (v) of this problem was there to show that regexes do have some weaknesses, namely the inability to remember what they've seen without just exhaustively listing cases.
Problem Two: State Elimination

Using the state-elimination algorithm, convert this NFA into a regular expression. (You could just directly design a regular expression for this language, but we want you to specifically use the state elimination algorithm).

We begin by introducing a new start and accept state with the appropriate new ε-transitions:

Let’s eliminate state $q_3$ first. The only state with a transition into $q_3$ is $q_2$, and the only state $q_2$ has a transition into is $q_f$. Therefore, we’ll add a direct link from $q_3$ to $q_f$ labeled with the regular expression $\Sigma \varepsilon$ ($\Sigma$ for the entry into $q_3$ and $\varepsilon$ for the exit from $q_3$ to $q_f$). This yields this setup:

Eliminating states $q_2$ and $q_1$, in that order, gives this configuration:

Now, we eliminate $q_0$. We can enter $q_0$ from $q_s$, stay around using the $\Sigma$ self-loop, then leave by following the $a \Sigma \varepsilon$ transition. This gives the following:

The final regex is $\varepsilon \Sigma^* a \Sigma \varepsilon$, which easily simplifies to $\Sigma^* a \Sigma$. This makes sense – it means that the regex matches any number of characters, then something that ends with $a$ followed by any two characters.

Why we asked this question: The beauty of the automata transformations we’ve seen so far is that it lets us study the same class of objects – the regular languages – using one of three totally different lenses. The constructions are a bit tricky, though, and to help you get a handle for how they work, we wanted you to practice working through them on a few special cases. This particular case, in our opinion, wasn’t too tough of a sample input.
Problem Three: The Myhill-Nerode Theorem

i. What is the formal definition of the statement $x \not\equiv_L y$? Explain it in plain English. Give an example of two strings $x$ and $y$ along with a language $L$ where $x \not\equiv_L y$ holds.

The formal definition of $x \not\equiv_L y$ is

$$x \not\equiv_L y \quad \text{if} \quad \exists w \in \Sigma^* \ (xw \in L \iff yw \notin L).$$

In plain English, this means there’s a string you can tack onto the end of both $x$ and $y$ such that exactly one of $xw$ and $yw$ will be in $L$.

As an example, the strings $a$ and $aa$ are distinguishable relative to $\{a^nb^n \mid n \in \mathbb{N}\}$; tacking on $bb$ makes $aabb \in L$ and $abb \notin L$.

ii. The proof hinges on the fact that if $x \not\equiv_L y$, then $x$ and $y$ cannot end in the same state when run through any DFA for a language $L$. We sketched a proof of this in class. Explain intuitively why this is the case.

Let $x$ and $y$ be strings where $x \not\equiv_L y$, and let $w$ be a string that distinguishes $x$ and $y$. If $x$ and $y$ end up in the same state in a DFA $D$ for the language $L$, then the strings $xw$ and $yw$ will end up in the same state in $D$ because once $x$ and $y$ have “collided,” reading the same characters after reading $x$ and $y$ will cause the DFA to follow the same series of transitions. If the state that $xw$ and $yw$ end up in is accepting, then both $xw$ and $yw$ are accepted even though one of those strings isn’t in $L$, and if that state is rejected the both $xw$ and $yw$ are rejected even though one of those strings is in $L$. Either way, we contradict the fact that $D$ is a DFA for the language $L$.

iii. Explain, intuitively, why $S$ has to be an infinite set for this proof to work.

The proof of the Myhill-Nerode theorem works by arguing that no matter how large of a DFA we build for a language $L$, we can always find a larger number of pairwise distinguishable strings. If we have infinitely many strings in $S$, we can always ensure that we have more strings in $S$ than there are states in any proposed DFA for $L$. On the other hand, if $S$ is finite, this line of reasoning only works on DFAs that have fewer than $|S|$ states.

iv. Does anything in the proof require that $S$ be a subset of $L$?

Nope! Not at all. We just need $S$ to be a subset of $\Sigma^*$. This is really important, actually – when you’re trying to show that a language isn’t regular, you don’t need to limit your search for distinguishable strings purely to strings in $L$. You can use any strings you’d like.

Why we asked this question: This question was designed to get you engaged with the proof of the Myhill-Nerode theorem. The proof of the theorem is a bit tricky and nuanced, and the statement of the proof is quite complex. We hoped that these problems would make it easier for you to see what the theorem requires, what it doesn’t require, and why it works.
Problem Four: Nonregular Languages Warmup

Let $\Sigma = \{1, \geq\}$ and consider the language $L = \{1^m \geq 1^n \mid m, n \in \mathbb{N} \text{ and } m \geq n\}$.

i. Give some specific examples of strings from the language $L$.

Here are a few strings from $L$: $1\geq 1, 1111\geq 11, \geq, 111\geq$, and $11111\geq 11111$.

ii. Without using the Myhill-Nerode theorem, give an intuitive justification for why $L$ isn’t regular.

Intuitively, any DFA for this language will at some level need to keep track of how many 1’s appear on the left-hand side of the $\geq$ sign. If the DFA can’t do that, then when it sees the 1’s appearing on the right-hand side of the $\geq$ sign, it won’t be able to tell whether it saw more 1’s the first time than the second time. However, the DFA only has finitely many states, so there’s no way for it to exactly remember the number of 1’s that it’s seen.

iii. Use the Myhill-Nerode theorem to prove that $L$ isn’t regular. You’ll need to find an infinite set of strings that are pairwise distinguishable relative to $L$. As a hint, see if you can think of some strings that would have to be treated differently by any DFA for $L$, then see what happens if you gather all of them together into a set.

Proof: Let $S = \{1^n \mid n \in \mathbb{N}\}$. Notice that $S$ is infinite because it contains one string per natural number and there are infinitely many natural numbers. We also claim that any two distinct strings from $S$ are distinguishable relative to $L$. To see this, consider any two strings $1^m, 1^n \in S$ with $m \neq n$. Assume without loss of generality that $m > n$. Then $1^m \geq 1^n \in L$, but $1^n \geq 1^m \notin L$. Therefore, we see that $1^m \not\equiv_L 1^n$. Consequently, by the Myhill-Nerode theorem, we know that $L$ is not regular. ■

Why we asked this question: This question was designed to walk you through the thought process for showing that a particular language is nonregular. Part (i) asked you to take the (concise) definition of the language $L$ and get a feel for what the strings in $L$ actually look like. From there, we hoped you’d get a more concrete sense of what this language is. Part (ii) asked you to build an intuition for why the language isn’t regular, which is a good idea because it forces you to think about what a DFA for the language would have to look like and what it would have to do. In part (iii), we asked you to formalize your reasoning so that you could say with certainty that the language is definitely not regular. As you work through more proofs of nonregularity going forward, we hope that you use some of the techniques from this problem to help guide your efforts.
Problem Five: Nonregular Languages

Here are some more problems to help you get used to proving that certain languages aren’t regular.

i. Let $\Sigma = \{a, b\}$ and let $L = \{a^n b^m | n, m \in \mathbb{N} \text{ and } n \neq m\}$. Explain why this language is not the complement of the language $\{a^n b^n | n \in \mathbb{N}\}$.

This language is not the complement of the language $\{a^n b^n | n \in \mathbb{N}\}$ because the complement of $\{a^n b^n | n \in \mathbb{N}\}$ contains strings like $ba$ or $abba$ that don’t consist of a string of $a$’s followed by a string of $b$’s. However, the new language $\{a^n b^m | n, m \in \mathbb{N} \text{ and } n \neq m\}$ doesn’t contain strings like these, so it’s not the complement of $\{a^n b^n | n \in \mathbb{N}\}$.

ii. Let $\Sigma = \{a, b\}$ and let $L = \{a^n b^m | n, m \in \mathbb{N} \text{ and } n \neq m\}$. Prove that $L$ is not regular.

Proof: Let $S = \{a^n | n \in \mathbb{N}\}$. This set is infinite, since it contains one string per natural number. Moreover, we claim that all strings in $S$ are distinguishable relative to $L$. To see this, consider any strings $a^n, a^m \in S$ where $n \neq m$. Then $a^n b^n \not\in L$ but $a^m b^n \in L$. Thus $a^n$ and $a^m$ are distinguishable relative to $L$, and since they were chosen arbitrarily we conclude that any pair of strings in $S$ are distinguishable relative to $L$. Therefore, by the Myhill-Nerode theorem, $L$ is not regular. ■

iii. Let $\Sigma = \{a\}$ and let $L = \{w \in \Sigma^* | w \text{ is a palindrome}\}$. Prove that $L$ is regular.

Proof: All strings made only of $a$’s are palindromes, since they’re the same forwards and backwards. Thus $L = \Sigma^*$, which is a regular language. ■

iv. Let $\Sigma = \{a, b\}$ and let $L = \{w \in \Sigma^* | w \text{ is a palindrome}\}$. Prove that $L$ is not regular.

Proof: Let $S = \{a^n | n \in \mathbb{N}\}$. This set is infinite, since it contains one string per natural number. Moreover, we claim that all strings in $S$ are distinguishable relative to $L$. To see this, consider any strings $a^n, a^m \in S$ where $n \neq m$. Then $a^n b a^n \not\in L$ but $a^n b a^m \not\in L$. Thus $a^n$ and $a^m$ are distinguishable relative to $L$, and since they were chosen arbitrarily we conclude that any pair of strings in $S$ are distinguishable relative to $L$. Therefore, by the Myhill-Nerode theorem, $L$ is not regular. ■

Why we asked this question: Part (i) of this question was designed to highlight a nuance of complements of languages that hasn’t come up in any other contexts yet. Part (ii) was then designed as a (hopefully) straightforward application of the Myhill-Nerode theorem. Part (iv) was designed as a more complex Myhill-Nerode argument because you need to make sure that you can actually distinguish the strings in the middle of the proof. As you saw in part (iii), the language is regular if you just have one character in your alphabet, so the proof in part (iv) would have to use the character $b$ at least once.