Problems for Week Eight

Concept Checks

We've learned a TON about languages/automata since last time we met! As a warm up, let's do some review.

i. What is a language? (I.e., what "type" does it have—is it a string? A function?)

ii. Name at least two differences between DFAs and NFAs.

iii. Give three different definitions for what a regular language is.

iv. Name at least two closure properties of regular languages.

Designing Regular Expressions

Below are a list of alphabets Σ 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.

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

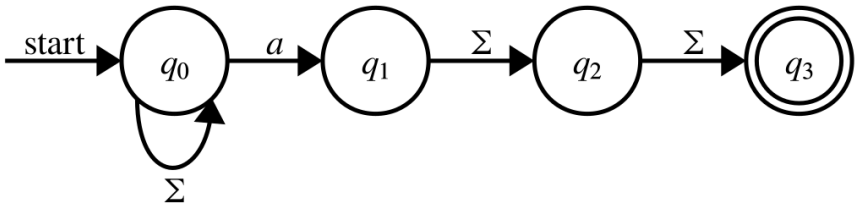
iii. Let $\Sigma = \{a, b\}$ and let $L = \{ w \in \Sigma^* \mid w$ contains two b's separated by exactly five characters $\}$. Write a regular expression for L.

iv. Let $\Sigma = \{a, b\}$ and let $L = \{ w \in \Sigma^* \mid w$ is a nonempty string whose characters alternate between a's and b's $\}$. Write a regular expression for L.

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

State Elimination

Below is an NFA for a language from the "Practice with Automata" handout (now up on the CS103A website!):



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).

Closure Properties

We've seen many examples of closure properties of regular languages. The proofs of those closure properties tend to fall into one of two different categories. First, there are proofs that work by directly converting automata or regexes for some languages into an automaton or regex for the resulting language. Second, there are proofs that work by applying other closure properties in various combinations. In this problem, we'd like you to prove another closure property of the regular languages. Remember that you have lots of different tools at your disposal!

Prove that the regular languages are closed under subtraction. That is, prove that if L_1 and L_2 are regular languages over some alphabet Σ, then the language $L_1 - L_2$ is also regular.

The Myhill-Nerode Theorem

The Myhill-Nerode theorem says the following:

> Let L be a language over Σ. If there is a set $S \subseteq \Sigma^*$ such that
>
> · S contains infinitely many strings, and
>
> · every pair of two distinct strings $x, y \in S$ are distinguishable relative to L (that is, $x \not\equiv_L y$),
>
> then L is not a regular language.

Below is a (slightly modified) version of the proof of the Myhill-Nerode theorem from lecture:

> ***Proof:*** Let L be an arbitrary language over Σ. Let $S \subseteq \Sigma^*$ be an infinite set of strings with the following property: if $x, y \in S$ and $x \neq y$, then $x \not\equiv_L y$. We will show that L is not regular.
>
> Suppose for the sake of contradiction that L is regular. This means that there must be some DFA D for L. Let k be the number of states in D. Since S is an infinite set, we can choose k+1 distinct strings from S and run each of those strings through D. Because there are only k states in D and we've chosen k+1 distinct strings from S, by the pigeonhole principle we know that at least two strings from S must end in the same state in D. Choose any two such strings and call them x and y.
>
> Since $x, y \in S$ and $x \neq y$, we know that $x \not\equiv_L y$. Consequently, by our earlier theorem, we know that x and y must end in different states when run through D. But this is impossible – we chose x and y specifically because they end in the same state when run through D. We have reached a contradiction, so our assumption must have been wrong. Thus L is not a regular language. ■

This question explores the theorem in a bit more detail.

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.

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.

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

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

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.

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

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.

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

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

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

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

Context-Free Grammars

Here's some practice problems to help you get comfortable designing CFGs. There are a number of patterns that come up over the course of these problems, and we hope that by the time you've finished working through them you have a deeper understanding of how CFGs work!

i. Let $\Sigma = \{a, b\}$ and let $L = \{ w \in \Sigma^* \mid w$ has no a's or has no b's $\}$. Write a CFG for L.

ii. Let $\Sigma = \{a, b\}$ and let $L = \{ w \in \Sigma^* \mid w$ has at least one a and at least one b $\}$. Write a CFG for L.

iii. Let $\Sigma = \{a, b\}$ and let $L = \{ a^n b a^n \mid n \in \mathbb{N} \}$. Write a CFG for L.

iv. Let $\Sigma = \{a, b\}$ and let $L = \{ a^n b^{2n} \mid n \in \mathbb{N} \}$. Write a CFG for L.

v. Let $\Sigma = \{a, b\}$ and let $L = \{ a^n b^m \mid n, m \in \mathbb{N}$ and $n \leq m \leq 5n \}$. Write a CFG for L.

vi. Let $\Sigma = \{a, b\}$ and let $L = \{ a^n b^m \mid n, m \in \mathbb{N}$ and $n \neq m \}$. Write a CFG for L.

vii. Let $\Sigma = \{a, b, c\}$ and let $L = \{a^n b^m c^p \mid n, m, p \in \mathbb{N}$ and $n = m$ or $n = p \}$. Write a CFG for L.

viii. Let $\Sigma = \{a, b\}$ and let $L = \{a^n b^n \mid n \in \mathbb{N}\}$. Write a CFG for L^*, the Kleene closure of L.

ix. Let $\Sigma = \{a, b\}$ and let $L = \{a^n b^m \mid n, m \in \mathbb{N}$ and either $n=2m$ or $m=2n \}$. Write a CFG for L.

x. Let $\Sigma = \{a, b\}$ and let $L = \{a^n b^n \mid n \in \mathbb{N}\}$. Write a CFG for $\overline{L}$, the complement of L.