# 1. Nonregular Languages Review

a. What problems do nonregular languages correspond to?

b. Intuitively, why is $E = \{a^n b^n \mid n \in \mathbb{N}\}$ **not regular**? Meanwhile, intuitively, why is the language $L = \{a^n b^n \mid n \in \mathbb{N} \text{ and } n \leq 103\}$ **regular**?

c. For some language $L$ over $\Sigma$ and strings $x$ and $y$, the formal definition of the statement "$x$ and $y$ are distinguishable relative to $L$", denoted by $x \not\equiv_L y$, is $\exists w \in \Sigma^*. \ (xw \in L \leftrightarrow yw \notin L)$. Explain this definition in plain English.

d. Explain the definition of a distinguishing set for $L$: $\forall x \in S. \ \forall y \in S. \ (x \neq y \to x \not\equiv_L y)$

   Given an arbitrary language, what is the smallest distinguishing set for it?

e. For the language $L = \{a^n b^n \mid n \in \mathbb{N}\}$, give an example of two strings $x$ and $y$ where $x \not\equiv_L y$ is **true**. Give an example of two strings $x$ and $y$ where $x \not\equiv_L y$ is **false**.

# 2. Proving Languages are Not Regular

The Myhill-Nerode theorem says the following:

> Let $L$ be a language over $\Sigma$. If there is a set $S \in \Sigma^*$ such that
> - $S$ contains infinitely many strings, and
> - every pair of 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.

a. Explain intuitively why $S$ has to be an infinite set for this theorem to work.

b. Does $S$ have to be a subset of $L$? Why or why not?

c. Give an example of a distinguishing set for the language $L = \{a^n b^n \mid n \in \mathbb{N}\}$.

d. Let's practice using the theorem. Let $\Sigma = \{a, b\}$ and let $L = \{b^n a^m \mid n, m \in \mathbb{N} \text{ and } n \neq m\}$.

   (1) Explain why $L$ is not the complement of the language $\{a^n b^n \mid n \in \mathbb{N}\}$.

   (2) Give an intuitive justification for why $L$ isn't regular – what would we need to "remember" that would not fit in a finite amount of memory?

   (3) 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 $L$. Finding this set is the difficult part of any nonregular language proof. Think of some category of 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.

# 3. Writing Regular Expressions

Here are some tips for writing regular expressions:

- Think about ways to simplify the problem. Is there a choice between multiple options, which you could represent with $\cup$? Is there some way to split strings in this language into multiple parts or sections, which you could concatenate?
- Try writing out example strings in the language. A regex can only generate arbitrarily long strings using the $^*$ operator. Look out for a repeating pattern that you can star.

To practice with regular expressions, write a regular expression for each of these languages.

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

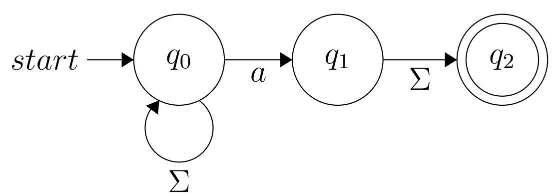
b. Let $\Sigma = \{a, b\}$ and let $L = \{w \in \Sigma^* \mid w \neq \varepsilon$ and the first and last character of $w$ are the same$\}$.

c. Let $\Sigma = \{a, b\}$ and let $L = \{w \in \Sigma^* \mid$ some substring of $w$ consists of two $b$s separated by five characters $\}$.

d. Let $\Sigma = \{a, b\}$ and let $L = \{w \in \Sigma^* \mid w$ does not contain two consecutive $a$s or $b$s$\}$. (Hint: Write out some strings in this language. What do you notice?)

# 4. The State Elimination Algorithm

Let's practice the state elimination algorithm, which converts an NFA into a regular expression. Consider this NFA:




a. Prepare the NFA for the state elimination algorithm by adding two new states, $q_{start}$ and $q_{end}$, adding an $\varepsilon$ transition from $q_{start}$ to the old start state, adding an $\varepsilon$ transition from all of the accept states to $q_{end}$, marking all of the accept states as no longer-accepting, and marking the new end state as accepting.

To eliminate a state $q$, 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 $q_{out}$ may be the same state.

b. Eliminate state $q_2$ from the NFA.

c. Eliminate state $q_1$ from the NFA.

d. Eliminate state $q_0$ from the NFA. What is the final regex?