1. Let \( L \) be a regular language. Let \( L' \subseteq L \). Is \( L' \) necessarily regular? Why?

2. A language \( L \) is called finite if it contains finitely many strings. Prove that every finite language is regular.

3. Find regular expressions for the following languages.
   (a) \( \{ w : w \text{ contains the substring 0101} \} \)
   (b) \( \{ w : w \text{ has length at least 3 and its third symbol is a 0} \} \)
   (c) \( \{ w : w \text{ starts with a 0 and has odd length, or starts with a 1 and has even length} \} \)
   (d) \( \{ w : w \text{ is any string except 11 and 111} \} \)

4. Can the following sets of strings be accepted by finite automata? Justify your answers!
   (a) \( \{1^n : n \text{ is a prime number}\} \)
   (b) \( \{0^{2n}1^{2m} : n \text{ and } m \text{ are integers}\} \)
   (c) \( \{x : x \text{ is a binary power of two}\} \)
   (d) \( \{x : \text{the center symbol of } x \text{ is a 1}\} \)

5. (A Worked Problem). Let \( S \) be a language. Show that \( S^* = (S^*)^* \).
   **Solution:** We need to show that \( S^* \subseteq (S^*)^* \) and \( (S^*)^* \subseteq S^* \). To show that \( S^* \subseteq (S^*)^* \), recall that any word in a set is also in the set’s Kleene closure. For the other inclusion, suppose that \( w \in (S^*)^* \). Then
   \[
   w = w_1 \cdots w_n
   \]
   for some \( n \geq 0 \), where \( w_i \in S^* \) for all \( i \in \{1, \ldots, n\} \). Since \( w_i \in S^* \) for every \( i \), it follows that
   \[
   w_i = w_{i,1} \cdots w_{i,k_i}
   \]
   for some \( k_i \geq 0 \), where \( w_{i,j} \in S \) for all \( j \in \{1, \ldots, k_i\} \). Then
   \[
   w = w_{1,1} \cdots w_{1,k_1} w_{2,1} \cdots w_{2,k_2} \cdots w_n \cdots w_{n,k_n},
   \]
   where \( w_{i,j} \in S \). That is, \( w \) is the concatenation of a finite number of words \( (k_1 + \cdots + k_n \text{ words}) \) in \( S \), so by definition of the star operation, \( w \in S^* \).

6. Show that the regular sets are not closed under infinite union by producing an infinite family of regular sets whose union is not regular.

7. An epsilon move takes place when a finite automaton reads and changes state but does not move its tape head. Does this new operation add power to finite automata? Justify your answer.
8. Let $\Sigma = \{0, 1, +, =\}$ and

$$\text{ADD} = \{ x = y + z : x, y, z \text{ are binary integers, and } x \text{ is the sum of } y \text{ and } z \}.$$

Show that $\text{ADD}$ is not regular.

9. We can use closure properties to help prove certain languages are not regular. Start with

the fact that the language

$L_{01n} = \{ 0^n1^n : n \geq 0 \}$

is not regular. Prove the following languages not to be regular by transforming them, using

operations known to preserve regularity, to $L_{01n}$:

(a) $\{0^i1^j : i \neq j\}$;
(b) $\{0^n1^m2^{n-m} : n \geq m \geq 0\}$.

10. Design an algorithm to determine whether a finite automaton accepts an infinite set. Prove

that your algorithm is correct.

11. Exhibit an algorithm that detects whether one finite automaton accepts a subset of the set

accepted by another machine. Show that this procedure works.

12. Two states of a finite automaton are said not to be equivalent if there is a string which takes

one into an accepting state and the other into a rejecting state. How many strings must be

checked in order to determine whether two states are equivalent? Develop an algorithm for

this.

13. If $L$ is a language, and $a$ is a symbol, then $L/a$, the quotient of $L$ and $a$, is the set of strings

$w$ such that $wa \in L$. For example, if $L = \{a, aab, baa\}$, then $L/a = \{\varepsilon, ba\}$. Prove that if

$L$ is regular, so is $L/a$. That is, the set of regular languages are closed under the quotient

operation.

**Hint:** Start with a DFA for $L$ and consider the set of accepting states.

14. If $L$ is a language, and $a$ is a symbol, then $a\setminus L$ is the set of strings $w$ such that $aw \in L$. For

example, if $L = \{a, aab, baa\}$, then $a\setminus L = \{\varepsilon, ab\}$. Convince yourself, but not the grader,

that if $L$ is regular, so is $a\setminus L$. This operation is sometimes viewed as a “derivative,” and

$a\setminus L$ is written $\frac{dL}{da}$. These derivative apply to regular expressions in a manner similar to the

way ordinary derivatives apply to arithmetic expressions. Thus, if $R$ is a regular expression,

we shall use $\frac{dR}{da}$ to mean the same as $\frac{dL}{da}$, if $L = L(R)$.

(a) Show that $\frac{d(R+S)}{da} = \frac{dR}{da} + \frac{dS}{da}$;

(b) Give the rule for the “derivative” of $RS$. **Hint:** You need to consider two cases: if

$L(R)$ does or does not contain $\varepsilon$. This rule is not quite the same as the “product

rule” for ordinary derivatives, but is similar;

(c) Give the rule for the derivative of a closure; i.e., $\frac{d(R^*)}{da}$;

(d) Use the rules above to find the “derivatives” of regular expression $(0+1)^*011$ with

respect to 0 and 1;

(e) Characterize those languages $L$ for which $\frac{dL}{da} = \emptyset$;

(f) Characterize those languages $L$ for which $\frac{dL}{da} = L$. 

2
15. Give a family of languages $E_n$, where each $E_n$ can be recognized by an $n$-state NFA but requires at least $c^n$ states on a DFA for some constant $c > 1$. Prove that your languages have this property.

16. Show that the set of all binary integers that are the sum of exactly four (no more, no less!) positive squares is a regular set. **HINT:** They are all found by substituting for $m$ and $n$ in the formula $4^n(8m + 7)$.

17. (Challenging) If $A$ is a set of natural numbers and $k$ is a natural number greater than 1, let $B_k(A) = \{ w : w \text{ is the representation in base } k \text{ of some number in } A \}$.

Here, we do not allow leading 0s in the representation of a number. For example, $B_2(\{3, 5\}) = \{11, 101\}$ and $B_3(\{3, 5\}) = \{10, 12\}$. Give an example of a set $A$ for which $B_2(A)$ is regular but $B_3(A)$ is not regular. Prove that your example works.

18. (Challenging) Give an algorithm that takes a DFA $A$ and computes the number of strings of length $n$ (for some given $n$, not related to the number of states of $A$) accepted by $A$. Your algorithm should be polynomial in both $n$ and the number of states of $A$. 