10 The Pumping Lemma for Regular Languages

36. Prove that $L_5 = \{w \in \{a,b\}^* \mid |w|_a < |w|_b\}$ is not regular (where $|w|_a$ is the number of 'a's occurring in w).

    (Solution)

    (The x as we used in proving L_{ab} non-regular works here as well.)

    Suppose, by way of contradiction, L_5 is regular.
    Let n be the constant of the pumping lemma.
    Let $x = a^n b^n$.
    Since $x \in L_5$ and $|x| \geq n$, by the pumping lemma there must be some u, v and w such that $x = uvw$, $|uv| \leq n$, $|v| \geq 1$ and $uv^i w \in L_{ab}$ for all i.
    Since $|uv| \leq n$, both u and v must fall within the first n symbols of x. Thus, u and v must consist only of 'a's. Furthermore, v must include at least one 'a', since $|v| \geq 1$. Thus, there must be some j, k and l such that

    $$u = a^j, \qquad v = a^k, \qquad w = a^l b^n, \qquad j + k + l = n, \text{ and } k \geq 1.$$

    But suppose $i = 0$. Then

    $$uv^0 w = uw = a^j a^l b^n$$

    where $j + l = n - k$ which is strictly less than n. Thus $uv^i w \notin L_5$ for $i = 0$ and the pumping lemma does not hold for L_5, contradicting our assumption that it was regular.

37. Prove that the following language L_6 is not regular.

    $$L_6 \stackrel{\text{def}}{=} \{w \in \{a,b\}^* \mid |w|_a < |w|_b \text{ if } |w|_a \text{ even}, \ |w|_a > |w|_b \text{ otherwise } \}$$

    (Solution)

    Here our choice of x will depend on whether n is even or odd; we must account for either possibility. One easy way to do this is to let $x = a^{2n} b^{2n+1}$. Then $|x|_a$ is even and less than $|x|_b$, and $x \in L_6$. As before u

and v must both fall within the block of 'a's. Thus, there must be some j, k and l such that

$$u = a^j, \qquad v = a^k, \qquad w = a^l b^{2n+1}, \qquad j + k + l = 2n, \text{ and } k \geq 1.$$

Our choice of counter-example for i must depend on whether k is even or odd. In the first case $|uv^i w|_a$ will be even for all i. In the second it will be odd whenever i is even and *v.v.*.
Suppose k was even. Then $|uv^2 w|_a = 2n + k$. Since k is even and greater than zero, $2n + k$ is even and greater than $2n + 1$. Thus $|uv^2 w|_a$ is even but $|uv^2 w|_a > |uv^2 w|_b$ and $uv^2 w \notin L_6$.
Suppose, on the other hand, k was odd. Then, since $k \geq 1$, $|uv^0 w|_a = 2n - k$ is odd and less than $|uv^0 w|_b = 2n$. Therefore, $uv^0 w \notin L_6$.
In both cases there is some i for which the pumping lemma fails. Thus L_6 is not regular.

11 The Myhill-Nerode Theorem

Lemma 1 *Let $\mathcal{A}' = \langle Q', \Sigma, q_0', \delta', F' \rangle$, where Q', q_0', δ' and F' are as described above. Then $L = L(\mathcal{A}')$.*

38. Prove that, for all $w \in \Sigma^*$, $\hat{\delta}'(q_0', w) = [w]_{R_L}$.

    (Solution)

    By induction on $|w|$.

    (Basis:)

    $\hat{\delta}'(q_0', \varepsilon) = q_0' = [\varepsilon]_{R_L}$.

    (Induction:)

    $\hat{\delta}'(q_0', w\sigma) = \delta'(\hat{\delta}'(q_0', w), \sigma) = \delta'([w]_{R_L}, \sigma) = [w\sigma]_{R_L}$.

39. Use this to prove the lemma.

    (Solution)

$$\begin{aligned} w \in L(\mathcal{A}') &\Leftrightarrow \hat{\delta}'(q_0', w) \in F' \\ &\Leftrightarrow \hat{\delta}'(q_0', w) \in \{[w]_{R_L} \mid w \in L\} \\ &\Leftrightarrow w \in L. \end{aligned}$$

40. Consider, again, the problem of specifying schedules for a machine tool (see Section 8.2). Suppose, in this instance, that there is just a single type of part which requires two operations A_1 and A_2 to complete. These can be done in any order, although we will assume that operations always complete partially completed parts if they can. Thus, at any given time the partially completed parts will all be waiting for the same operation (although which operation can change over time). Assume, further, that there is an unbounded amount of space to store partially completed parts; the only constraint on a schedule is that every part gets completed.

    (a) Describe this language.

        (Solution)

        This is the language over $\{A_1, A_2\}^*$ in which the number of 'A_1's and the number of 'A_2's are equal.

(b) Use Myhill-Nerode to show that the set of feasible schedules, given these constraints, is not regular.

(Solution)

Let's call this language L_7. We can use roughly the same sequence of strings as we used in the example:

$$\langle A_1^0, A_1^2, \ldots, A_1^i, \ldots \mid i \geq 0 \rangle.$$

We claim that every string in this sequence is distinct wrt R_{L_7} from every other string. To see this, consider A_1^i and A_1^j for $i \neq j$. Then $A_1^i A_2^i \in L_{ab}$ while $A_1^j A_2^i \notin L_{ab}$. Thus, A_2^i witnesses that A_1^i and A_1^j are *not* related by R_{L_7}. Since i and j are arbitrary, every string in the sequence is distinct, wrt R_{L_7}, from every other string in the sequence; no two share the same equivalence class. It follows that there must be at least as many equivalence classes of R_{L_7} as there are strings in the sequence; R_{L_7} cannot have finite index.

41. Using the Myhill-Nerode Theorem, prove the language L_3 of Problem 25 is regular.

(Solution)

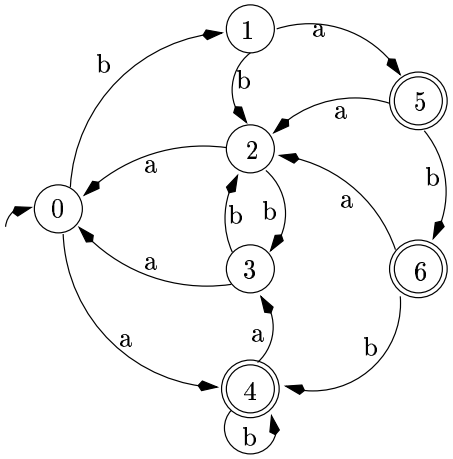
For all $w \in \{m_1, m_2, a_1, a_2, a_1 2\}^*$, let Pending($w$) be either the set of messages left unacknowledged at the end of the sequence w or FAIL if at some point during the sequence w the protocol is violated (either more than one message of a given type is outstanding or an acknowledge occurs without the corresponding message(s) being outstanding). Then Pending(w) $\in \{\emptyset, \{m_1\}, \{m_2\}, \{m_1, m_2\}, \text{FAIL}\}$.
Suppose Pending(w) $\neq$ Pending(u). Without loss of generality, suppose Pending(w) $\neq$ FAIL. Let v be either ε, 'a_1', 'a_2' or 'a_{12}', as appropriate to acknowledge exactly those messages in Pending(w). Then $wv \in L_3$ but, uv either leaves some message unacknowledged or acknowledges some non-outstanding message. On the other hand, if Pending(w) = Pending(u) then either there is no string that extends either w or u to become a string in L_3 (i.e., if Pending(w) and Pending(u) are both FAIL), or $wv \in L_3$ iff v acknowledges the messages in Pending(w) (along with any additional messages in v) without violating the protocol iff v acknowledges the messages in Pending(u) (along with any additional

messages in v) without violating the protocol iff $uv \in L_3$. Thus, for all $w \in \{m_1, m_2, a_1, a_2, a_1 2\}^*$,

$$[w]_{R_{L_3}} = \{u \mid \text{Pending}(u) = \text{Pending}(w)\}.$$

Since Pending(w) can take only five values, there are no more than five such equivalence classes. Thus, by the Myhill-Nerode Theorem, L_3 is regular.

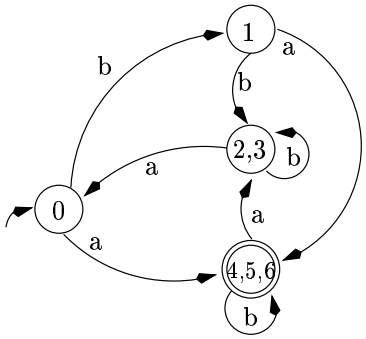
42. Minimize the following DFA.



(Solution)

1	ba					
2	a	a				
3	a	a				
4	ε	ε	ε	ε		
5	ε	ε	ε	ε		
6	ε	ε	ε	ε		
	0	1	2	3	4	5

States 2 and 3 are equivalent as are states 4, 5 and 6.


1
a
b
b
2,3
b
a
0
a
a
4,5,6
b

12 Closure Properties of the Class of Regular Languages

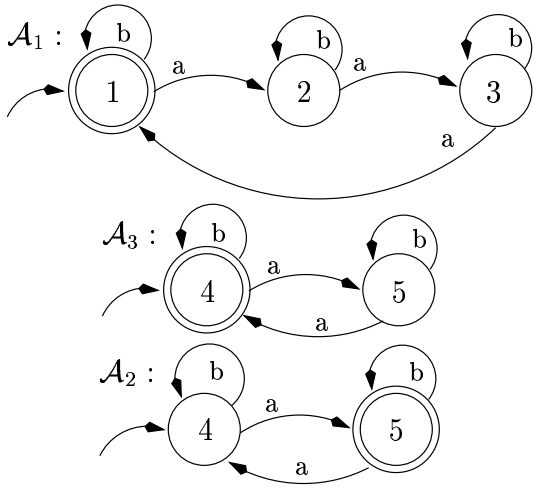
43. Using this approach, show that the set of strings over $\{a, b\}$ in which the number of 'a's is divisible by three but not divisible by two is regular.

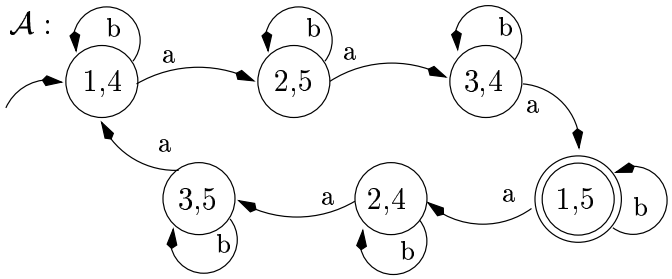
    (Solution)

    Let L denote this language. Let L_1 denote the language in which the number of 'a's is divisible by three and L_2 the set in which the number is not divisible by two. Then $L = L_1 \cap L_2$, $L_1 = L((b^*ab^*ab^*ab^*)^*)$, and $L2 = \overline{L_3}$ where $L_3 = L((b^*ab^*ab^*)^*)$. Thus, L is the intersection of a regular language and the complement of a regular language. Since the class of regular languages is closed under complement and intersection, L is regular.

44. Starting with simple automata for your "obviously" regular languages and using the constructions of the proofs of the closure properties, build a DFA for this language.

    (Solution)





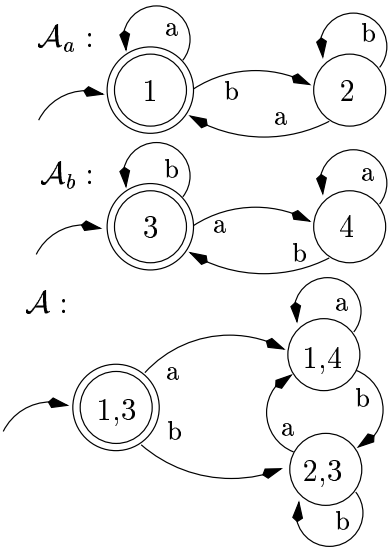
45. Consider the two languages:

    L_a**:** The set of strings over $\{a, b\}$ in which the last symbol is not 'b'.

    L_b**:** The set of strings over $\{a, b\}$ in which the last symbol is not 'a'.

    Using the approach of the previous problem, construct a DFA accepting the language of strings that satisfy both of these descriptions.

    (Solution)



46. What is that language? Explain why it is not empty.

    (Solution)

    The language is $\{\varepsilon\}$. It is not empty because the empty string has no last symbol and, thus, does not end either with 'a' or 'b'.

47. Let $L_1 = L(a^*ba^*)$ and $L_2 = L(b^*a)$. What is L_1/L_2?

    (Solution)

    If $w \in L_1/L_2$ then there is some $v \in L(b^*a)$ such that $wv \in L(a^*ba^*)$. Thus $wv = a^iba^j$ for some i and j and $v = b^ka$ for some k. Consequently, both b and j must equal 1. Therefore $wv = a^iba$ for arbitrary i and $w = a^i$. Thus, $L_1/L_2 = L(a^*)$

48. Let $L_3 = L(ba^*b)$ and L_1 remain the same. What is L_1/L_3?

    (Solution)

    Here, $w \in L_1/L_3$ implies that there is some $v \in L(ba^*b)$ such that $wv \in L(a^*ba^*)$. But, then, wv would contain a single 'b' while v, a substring of w, contains two—a contradiction! Hence there is no such w and $L_1/L_3 = \emptyset$.

49. Show that $w \in L(\mathcal{M}') \Leftrightarrow w \in L_1/L_2$.

    (Solution)

    To show that $w \in L(\mathcal{M}') \Rightarrow w \in L_1/L_2$, suppose $w \in L(\mathcal{M}')$. Then $\hat{\delta}(q_0, w) = q$ for some q such that $(\exists v \in L_2)[\hat{\delta}(q, v) \in F]$. Thus there is some $v \in L_2$ such that $\hat{\delta}(\hat{\delta}(q_0, w), v) \in F$, which is to say, $\hat{\delta}(q_0, wv) \in F$. Thus, there is some $v \in L_2$ such that $wv \in L_1$, in other words, $w \in L_1/L_2$.
    For the converse, $w \in L_1/L_2$ implies there is some $v \in L_2$ such that $wv \in L_1$. Thus $\hat{\delta}(q_0, wv) \in F$ and v is a string in L_2 such that $\hat{\delta}(\hat{\delta}(q_0, w), v) \in F$. Consequently, $\hat{\delta}(q_0, w) \in F'$, in other words, $w \in L(\mathcal{M}')$.

50. Suppose L is any nonempty language. What is Σ^*/L?

    (Solution)

    $\Sigma^*/L = \{w \mid (\exists v \in L)[wv \in \Sigma^*]\}$. Since L is non-empty, there is such a v for any $w \in \Sigma^*$ (in fact, any string in L will do for all w). Thus $\Sigma^*/L = \Sigma^*$.

51. What is $L/\emptyset$?

    (Solution)

    $L/\emptyset = \{w \mid (\exists v \in \emptyset)[wv \in L]\}$. But, as there are no strings at all in $\emptyset$ there are no such v for any w. Thus $L/\emptyset = \emptyset$.

52. What is $L/\{\varepsilon\}$?

    (Solution)

    $L/\{\varepsilon\} = \{w \mid (\exists v \in \{\varepsilon\})[wv \in L]\}$. If there is any $w \in L$ then $w \cdot \varepsilon \in L$ and $L/\{\varepsilon\} = L$. On the other hand, if there is no $w \in L$ then there is no such v and $L/\{\varepsilon\} = \emptyset$. But so does L. Therefore $L/\{\varepsilon\} = L$.

53. Suppose L_1, L_2 and L_3 are arbitrary languages. Show that

$$L_1/(L_2 \cup L_3) = (L_1/L_2) \cup (L_1/L_3)$$

    and that

$$L_1/(L_2 \cap L_3) = (L_1/L_2) \cap (L_1/L_3).$$

    (Solution)

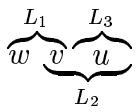
$$\begin{aligned}
L_1/(L_2 \cup L_3) &= \{w \mid (\exists v \in L_2 \cup L_3)[wv \in L_1]\} \\
&= \{w \mid (\exists v \in L_2)[wv \in L_1]\} \cup \{w \mid (\exists v \in L_3)[wv \in L_1]\} \\
&= (L_1/L_2) \cup (L_1/L_3). \\
L_1/(L_2 \cap L_3) &= \{w \mid (\exists v \in L_2 \cap L_3)[wv \in L_1]\} \\
&= \{w \mid (\exists v \in L_2)[wv \in L_1]\} \cap \{w \mid (\exists v \in L_3)[wv \in L_1]\} \\
&= (L_1/L_2) \cap (L_1/L_3).
\end{aligned}$$

54. Suppose, again, that L_1, L_2 and L_3 are arbitrary languages. What is $L_1/(L_2/L_3)$?

    (Solution)

    $L_2/L_3 = \{v \mid (\exists u \in L_3)[vu \in L_2]\}$.
    $L_1/(L_2/L_3) = \{w \mid (\exists v \in L_2/L_3)[wv \in L_1]\}$.
    $L_1/(L_2/L_3) = \{w \mid (\exists v, u)[u \in L_3 \text{ and } vu \in L_2 \text{ and } wv \in L_1]\}$.

    Which is to say, the strings of $L_1/(L_2/L_3)$ are those prefixes of strings in L_1 which are completed by prefixes of strings in L_2 which are, themselves, completed by strings in L_3.



55. What is $(L_1/L_2)/L_3$?

    (Solution)

    $L_1/L_2 = \{x \mid (\exists v \in L_2)[xv \in L_1]\}$.
    $(L_1/L_2)/L_3 = \{w \mid (\exists u \in L_3)[wu \in L_1/L_2]\}$.
    $(L_1/L_2)/L_3 = \{w \mid (\exists u \in L_3)[(\exists v \in L_2)[wuv \in L_1]]\}$ $\quad (x = wu)$.
    $(L_1/L_2)/L_3 = \{w \mid (\exists u \in L_3, v \in L_2)[wuv \in L_1]\}$.

    Which is to say, $(L_1/L_2)/L_3 = L_1/(L_3 \cdot L_2)$.

$$\overbrace{w\underbrace{u}_{L_3}\underbrace{v}_{L_2}}^{L_1}$$

56. Let $L = L(((a + ba)^*ba)^*)$ and $f = \{a \mapsto L(ab^*a), b \mapsto L(b^*ab^*)\}$. Give a regular expression for the language $f(L)$.

    (Solution)

    $((ab^*a + b^*ab^*ab^*a)^*b^*ab^*ab^*a)^*$.

57. Why is $\mathcal{A}'$ an NFA rather than another DFA.

    (Solution)

    Since δ is not one-to-one, it may be the case that $\delta(q_1, \sigma) = \delta(q_2, \sigma) = p$ for $q_1 \neq q_2$. Then $\{q_1, q_2\} \subseteq \delta'(p, \sigma)$.

58. Complete the proof by showing that

$$w \in L(\mathcal{A}) \Leftrightarrow w^{\mathrm{R}} \in L(\mathcal{A}').$$

    (Solution)

    First we prove the lemma: for all $q, p \in Q$, $\hat{\delta}(q, w) = p \Leftrightarrow q \in \hat{\delta}'(p, w^{\mathrm{R}})$. To show that $\hat{\delta}(q, w) = p \Rightarrow q \in \hat{\delta}'(p, w^{\mathrm{R}})$ (induction on the structure of w):

    (Basis:)

    $\hat{\delta}(q, \varepsilon) = q$ and $q \in \hat{\delta}'(q, \varepsilon^{\mathrm{R}})$.

    (IH:)

    Suppose that $w = v\sigma$ and that the claim is true of v.

(Ind:)

Suppose $\hat{\delta}(q, v \cdot \sigma) = p$ and $\hat{\delta}(q, v) = q_1$. Then $\delta(q_1, \sigma) = p$.
By the IH, $q \in \hat{\delta}'(q_1, v^{\mathrm{R}})$.
By the definition of δ', $q_1 \in \delta'(p, \sigma)$.
By an analog of Exercise 24, $\hat{\delta}(q, wv) = \bigcup_{q' \in \hat{\delta}'(q,w)} [\hat{\delta}(q', v)]$.
Thus, $\hat{\delta}'(p, \sigma v^{\mathrm{R}}) = \bigcup_{q' \in \hat{\delta}'(p,\sigma)} [\hat{\delta}(q', v^{\mathrm{R}})]$.
Since $q_1 \in \delta'(p, \sigma) = \hat{\delta}'(p, \sigma)$ and $q \in \hat{\delta}'(q_1, v^{\mathrm{R}})$, it follows that $q \in \hat{\delta}'(p, \sigma v^{\mathrm{R}}) = \hat{\delta}'(p, w^{\mathrm{R}})$.

To show that $q \in \hat{\delta}'(p, w^{\mathrm{R}}) \Rightarrow \hat{\delta}(q, w) = p$ (induction on the structure of w):

(Basis:)

Suppose $w = \varepsilon$. Since $p, q \in Q$ and there are no ε-transitions in the portion of $\mathcal{A}'$ restricted to Q, $\hat{\delta}'(q, \varepsilon) = \{q\}$. Therefore, $q \in \hat{\delta}'(p, \varepsilon^{\mathrm{R}}) \Rightarrow q = p$ and $\hat{\delta}(q, \varepsilon) = q = p$.

(IH:)

Suppose $w = v\sigma$ and the claim is true of v.

(Ind:)

To show that $q \in \hat{\delta}'(p, (v\sigma)^{\mathrm{R}}) \Rightarrow \hat{\delta}(a, v\sigma) = p$:
$q \in \hat{\delta}'(p, (v\sigma)^{\mathrm{R}}) \Rightarrow q \in \hat{\delta}'(p, \sigma v^{\mathrm{R}}) \Rightarrow q \in \hat{\delta}'(q_1, v^{\mathrm{R}})$ for some $q_1 \in \delta'(p, \sigma)$.
By the definition of δ', $q_1 \in \delta'(p, \sigma)$ implies $\hat{\delta}(q_1, \sigma) = p$.
By the IH, $q \in \hat{\delta}'(q_1, v^{\mathrm{R}}) \Rightarrow \hat{\delta}(q, v) = q_1$.
Then, $\hat{\delta}(q, w) = \hat{\delta}(q, v\sigma) = \delta(\hat{\delta}(q, v), \sigma) = p$.

To prove the theorem:

$$\begin{aligned} w \in L(\mathcal{A}) &\Leftrightarrow \hat{\delta}(q_0, w) = p, \text{ for some } p \in F \\ &\Leftrightarrow p \in \delta'(q_0', \varepsilon) \text{ and } q_0 \in \hat{\delta}'(p, w^{\mathrm{R}}) \\ &\Leftrightarrow q_0 \in \hat{\delta}'(q_0', w^{\mathrm{R}}) \\ &\Leftrightarrow w^{\mathrm{R}} \in L(\mathcal{A}'). \end{aligned}$$

59. Prove that the class of regular languages is closed under Suffix.

(Solution)

$$\begin{aligned}\text{Suffix}(L) &= \{w \mid (\exists v \in \Sigma^*)[vw \in L]\} \\ &= \{w \mid (\exists v' \in \Sigma^*)[w^{\mathrm{R}}v' \in L^{\mathrm{R}}]\} \\ &= \{w^{\mathrm{R}} \mid w \in \text{Prefix}(L^{\mathrm{R}})\} \\ &= (\text{Prefix}(L^{\mathrm{R}}))^{\mathrm{R}}\end{aligned}$$

60. Consider again a system of two processes (A and B) exchanging messages as in Exercise 25. Again A sends either 'm_1' or 'm_2' and B acknowledges with 'a_1', 'a_2' or 'a_{12}', where 'a_1' acknowledges 'm_1', 'a_2' acknowledges 'm_2' and 'a_{12}' acknowledges both. In contrast to Exercise 25, we will now allow any number of 'm_1's or 'm_2's to be outstanding. We require only that every message is eventually acknowledged and that no acknowledgment is sent unless there is some outstanding message(s) of the corresponding type. Show that the set of finite sequences of messages that satisfy this protocol is *not* regular.

(Solution)

Call this language L_{60}.
Let $L_1 = L_{60} \cap L(m_1^* a_1^*)$. Then $L_1 = \{m_1^i a_1^j \mid 0 \leq i = j\}$.
Let $h = \{m_1 \mapsto a, a_1 \mapsto b\}$. Then $h(L_1) = \{a^i b^j \mid 0 \leq i = j\}$, our canonical non-regular language.
Since the class of regular languages is closed under intersection and homomorphism, if L_{60} were regular then $h(L_1)$ would be also. Thus L_{60} cannot be regular.

13 Some Decision Problems for the Class of Regular Languages

61. Why don't the instances of these problems just include the languages themselves rather than representations of the languages?

    (Solution)

    The languages are, in general, infinite but we require our instances to be finite (otherwise even the process of reading them would not terminate).

62. Which of these properties are algorithmically decidable for the class of finite languages?

    (Solution)

    All of them. Since these languages are finite we can assume they are explicitly included in the instance. Finiteness is trivially true. The rest are implemented by simply examining the language.

13.1 Membership

63. Which is to say, the length of the computation in transitions (steps) of the DFA is $|w|$. What is the length of the computation in terms of its representation as a sequence of IDs; how many IDs are in the sequence?

    (Solution)

    $|w| + 1$—including one initial ID and one successor for each transition.

64. Can we establish such a bound on the computations of NFAs without ε-transitions? With them?

    (Solution)

    Without ε-transitions the same argument applies; each step consumes exactly one symbol. Moreover, because of the way we defined directly computes for NFAs with ε-transitions, each step of the computation of such an NFA also consumes exactly one symbol of the input. So, again, the same argument applies regardles of the species of automaton we employ.

This is not the case for acceptance in terms of paths in the transition graph. The problem is, with ε-transitions paths may contain loops of arbitrary length that do not consume any input. However, no such loop is essential in accepting the string. Thus, if there is any path including such loops there is one without them. We can, therefore, limit ourselves to paths in which no such loops occur. How long can these be? As an upper bound, we might allow for each ε-transition to be traversed once between each σ-transition. This bounds the number of edges to $|w| + 1$ times the number of ε-transitions plus 1. So, with a little more effort, we can get a finite bound on the length of the paths we need to check.

13.2 Finiteness

65. Give an algorithm for deciding finiteness that is based on known algorithms for deciding problems for graphs.

    (Solution)

    The language will be infinite iff there is some path from q_0 to a final state which includes a cycle. We can detect this by finding all the cycles in the graph and all the acyclic paths from q_0 to a final state. The language is infinite iff these have some state in common.

13.3 Equivalence

66. Sketch an algorithm to decide isomorphism of edge-labeled graphs.

    (Solution)

    Suppose the instance is $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ and $\mathcal{A}' = \langle Q', \Sigma', \delta', q_0', F'\rangle$. First of all, these are not isomorphic if $\Sigma \neq \Sigma'$. We will assume, then, that the alphabets are the same. We will also assume that they are both *connected*: every state in Q (respectively, Q') is reachable by some path from q_0 (respectively, q_0').
    To show that these are isomorphic—that they are identical except, possibly, for the names of the nodes—we need to exhibit a mapping h of the nodes in Q to those in Q' that shows how to relabel the nodes in Q to transform $\mathcal{A}$ into $\mathcal{A}'$. This mapping must be functional: it must not relabel any node in Q with more than one node in Q'; it must be total: it must relabel every node in Q; it must be one-to-one: it must not relabel

more than one node in Q with the same node in Q'; and it must be onto: the result of relabeling Q must be all of Q'. (All of which is to say that h is a *bijection* from Q to Q'.) Moreover, it must preserve the edges: if $h(q) = q'$ then, for all σ it must be the case that $h(\delta(q, \sigma)) = \delta'(q', \sigma)$. Finally, it must map final states to final states: $\{h(q) \mid q \in F\} = F'$. (We will call this set $h(F)$.)

We can build h inductively:

- Let $h(q_0) = q_0'$.
- If $h(q) = q'$, for each $\sigma \in \Sigma$, let $h(\delta(q, \sigma)) = \delta'(q', \sigma)$.
- Nothing else.

Implementing this as an algorithm we repeat the inductive step until there is no change in h. To see that this terminates, note that since Q and Q' are finite there are only finitely many pairs $\langle q, h(q)\rangle$. Thus, we can add only finitely many values to h before it becomes all of $Q \times Q'$ at which point it cannot get any larger. (In practice, it will, presumably, usually converge sooner.)

It is clear that this map preserves the edge relation and, since $\mathcal{A}$ is connected, it will be total. It is a simple matter to check if it is one-to-one, if it is onto, and if $h(F) = F'$. The automata are isomorphic iff the answer to all of these questions is 'YES'.

67. Suppose $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. What is $L(\mathcal{A}_1) \setminus L(\mathcal{A}_2)$?

    (Solution)

    $\emptyset$.

68. Suppose $L(\mathcal{A}_2) \subseteq L(\mathcal{A}_1)$. What is $L(\mathcal{A}_2) \setminus L(\mathcal{A}_1)$?

    (Solution)

    $\emptyset$.

69. Show that there is an effective construction that, given DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, builds a DFA accepting $L(\mathcal{A}_1) \setminus L(\mathcal{A}_2) \cup L(\mathcal{A}_2) \setminus L(\mathcal{A}_1)$.

    (Solution)

    Construct automata for $L(\mathcal{A}_1) \setminus L(\mathcal{A}_2)$, for $L(\mathcal{A}_2) \setminus L(\mathcal{A}_1)$ and, finaly, for thier union.

70. Use this result, along with decidability of emptiness, to show that equivalence of DFAs is decidable.

    (Solution)

$$\begin{aligned} L(\mathcal{A}_1) = L(\mathcal{A}_2) &\Leftrightarrow L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2) \text{ and } L(\mathcal{A}_2) \subseteq L(\mathcal{A}_1) \\ &\Leftrightarrow L(\mathcal{A}_1) \setminus L(\mathcal{A}_2) = L(\mathcal{A}_2) \setminus L(\mathcal{A}_1) = \emptyset \\ &\Leftrightarrow L(\mathcal{A}_1) \setminus L(\mathcal{A}_2) \cup L(\mathcal{A}_2) \setminus L(\mathcal{A}_1) = \emptyset. \end{aligned}$$

    In this way we can combine the constructions for difference and union and the decision procedure for emptiness to make a decision procedure for equivalence.

71. Prove that the question of whether a given regular language is closed under reversal is decidable.

    (Solution)

    L is closed under reversal iff $L^{\mathrm{R}} \subseteq L$. Note that $A \subseteq B \Rightarrow A^{\mathrm{R}} \subseteq B^{\mathrm{R}}$. Hence, $L^{\mathrm{R}} \subseteq L \Rightarrow (L^{\mathrm{R}})^{\mathrm{R}} \subseteq L^{\mathrm{R}}$, which is to say, $L \subseteq L^{\mathrm{R}}$.
    Thus, L is closed under reversal iff $L = L^{\mathrm{R}}$.
    To decide this, construct the machine for the reversal of the given DFA and use the equivalence algorithm to decide if the two machines accept the same language.

72. Show that decidability of both emptiness and membership is a consequence of decidability of equivalence.

    (Solution)

    Note that $\emptyset$ is accepted by the one-state DFA with no final states. (There is actually one of these for each alphabet.) To determine if a given DFA accepts the empty language use the equivalence algorithm to decide if it accepts the same language as the DFA for the empty language over the same alphabet.

    Deciding membership is only slightly more complicated. Given a DFA $\mathcal{A}$ and a string w, construct a DFA $\mathcal{A}_w$ which accepts the singleton language

$\{w\}$. This can be done as follows:

$$
\begin{aligned}
&Q = \{v \mid v \text{ is a prefix of w}\} \cup \{\text{fail}\} \\
&q_0 = \varepsilon \\
&F = \{w\} \\
&\delta(q,\sigma) = \begin{cases} v\sigma & \text{if } v\sigma \text{ is a prefix of w,} \\ \text{fail} & \text{otherwise .} \end{cases}
\end{aligned}
$$

Then $\hat{\delta}(\varepsilon, v) = \begin{cases} v & \text{if } v \text{ is a prefix of w,} \\ \text{fail} & \text{otherwise .} \end{cases}$

and $L(\mathcal{A}_w) = \{v \mid \hat{\delta}(\varepsilon, v) = w\} = \{w\}$.

Next, build the DFA that accepts $\mathcal{A} \cap \mathcal{A}_w$ and pass this to the algorithm for deciding emptiness. $w \in L(\mathcal{A})$ iff $\mathcal{A} \cap \mathcal{A}_w \neq \emptyset$. Return 'YES' iff the emptiness algorithm returns 'NO'.