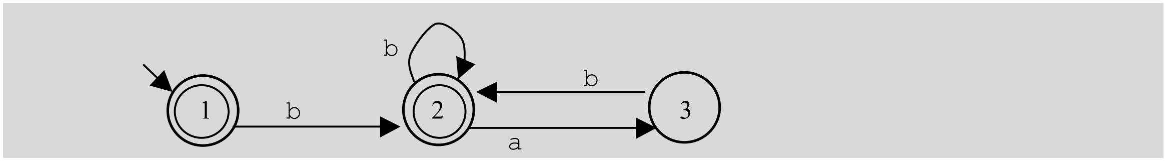
Regular Languages

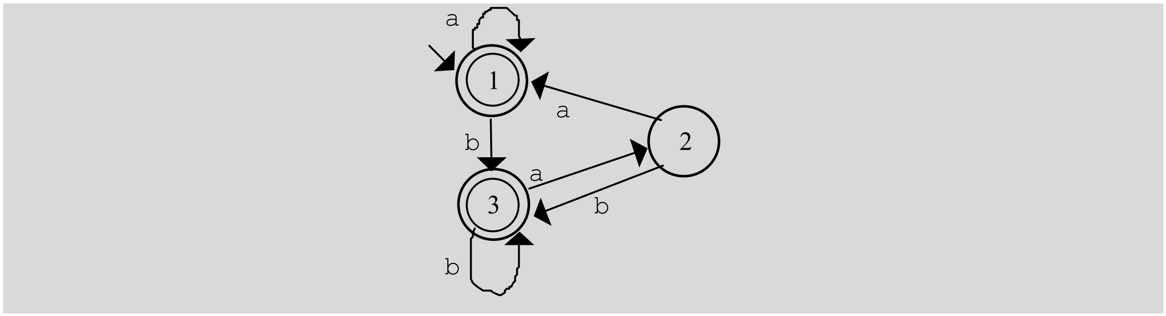
5 Finite State Machines

2 Build a deterministic FSM for each of the following languages:

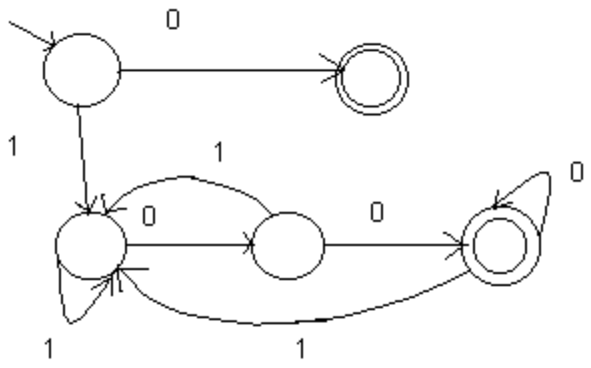
a) {w ∈ {`a`, `b`}* : every `a` in w is immediately preceded and followed by `b`}.



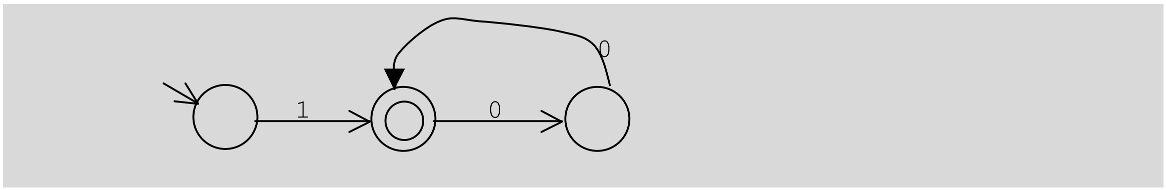
b) {w ∈ {`a`, `b`}* : w does not end in `ba`}.



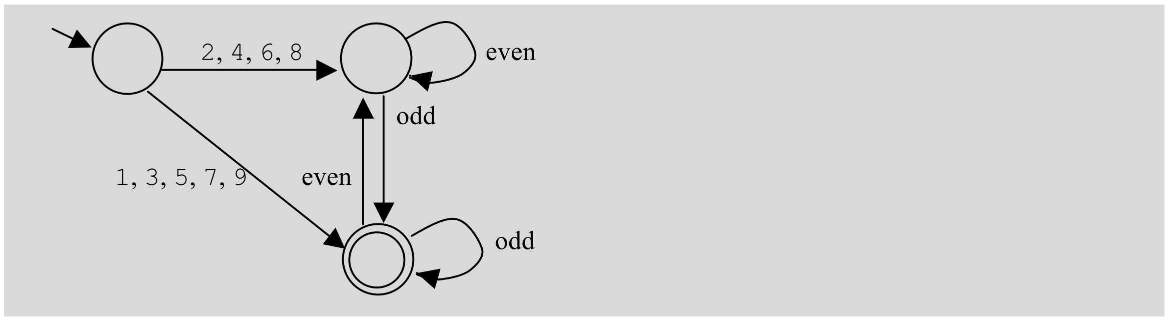
c) {w ∈ {`0`, `1`}* : w corresponds to the binary encoding, without leading 0's, of natural numbers that are evenly divisible by 4}.



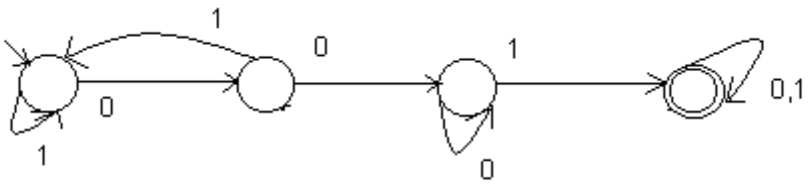
d) {w ∈ {`0`, `1`}* : w corresponds to the binary encoding, without leading 0's, of natural numbers that are powers of 4}.



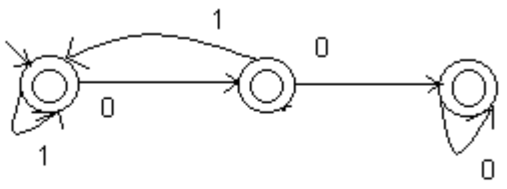
e) {w ∈ {0-9}* : w corresponds to the decimal encoding, without leading 0's, of an odd natural number}.



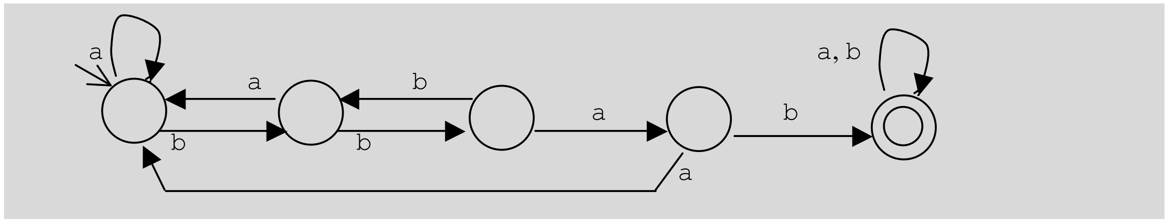
f) {w ∈ {0, 1}* : w has 001 as a substring}.



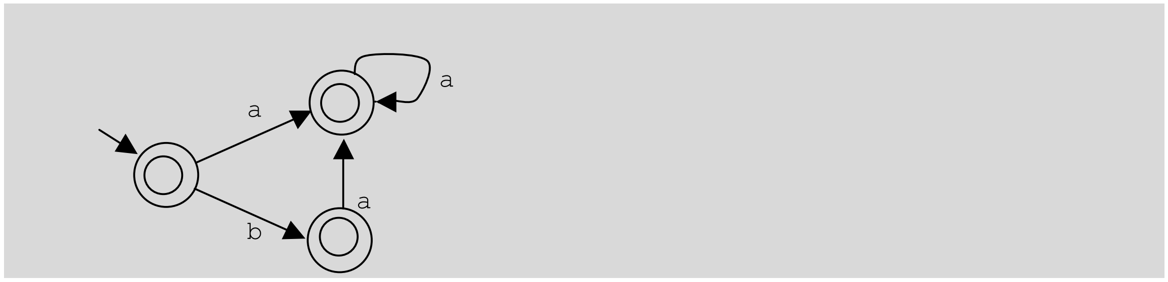
g) {w ∈ {0, 1}* : w does not have 001 as a substring}.



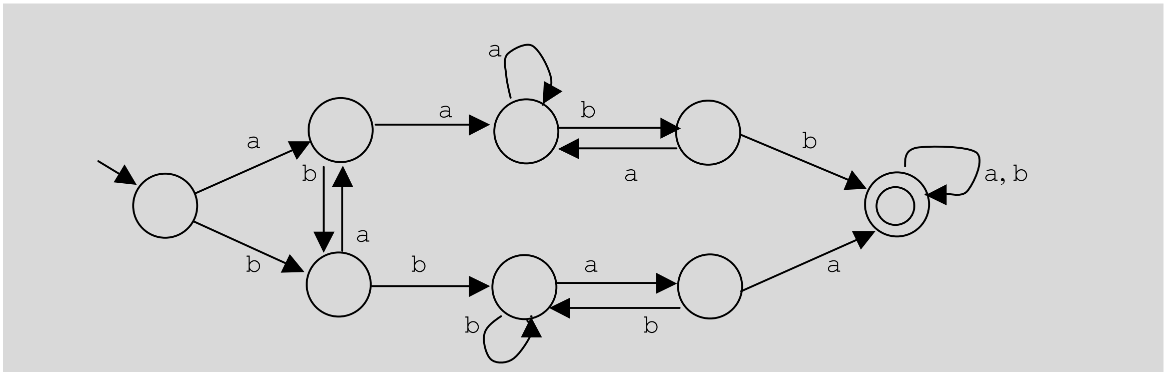
h) {w ∈ {a, b}* : w has bbab as a substring}.



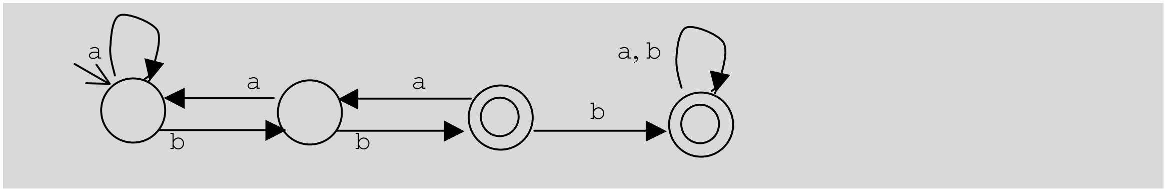
i) {w ∈ {a, b}* : w has neither ab nor bb as a substring}.



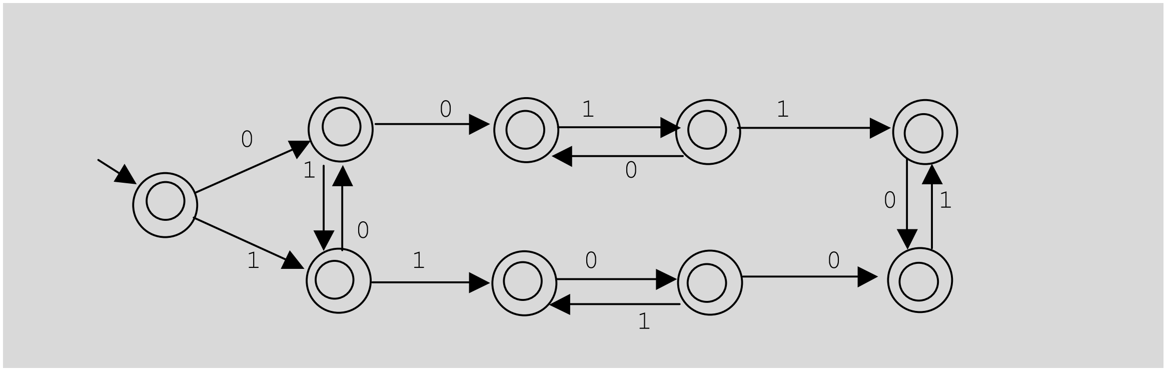
j) {*w* ∈ {a, b}* : *w* has both aa and bb as a substrings}.



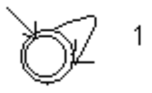
k) {*w* ∈ {a, b}* : *w* contains at least two b's that are not immediately followed by a's}.



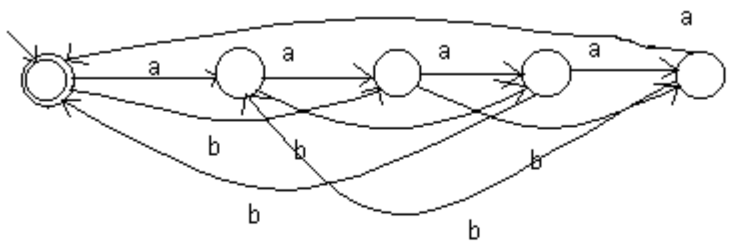
l) The set of binary strings with at most one pair of consecutive 0's and at most one pair of consecutive 1's.



m) {*w* ∈ {0, 1}* : none of the prefixes of *w* ends in 0}.

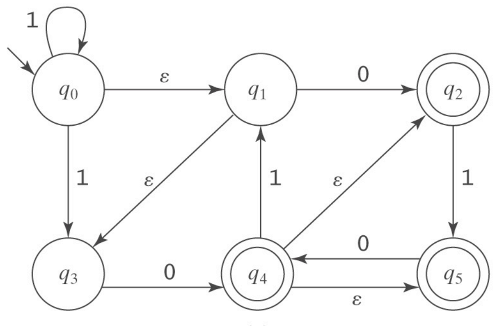


n) $\{w \in \{a, b\}^*: (\#_a(w) + 2 \cdot \#_b(w)) \equiv_5 0\}$. ($\#_a w$ is the number of a's in w).



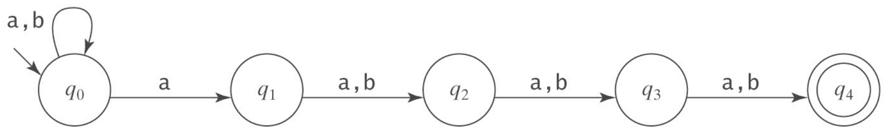
9 For each of the following NDFSMs, use *ndfsmtodfsm* to construct an equivalent DFSM. Begin by showing the value of *eps*(q) for each state q:

a)



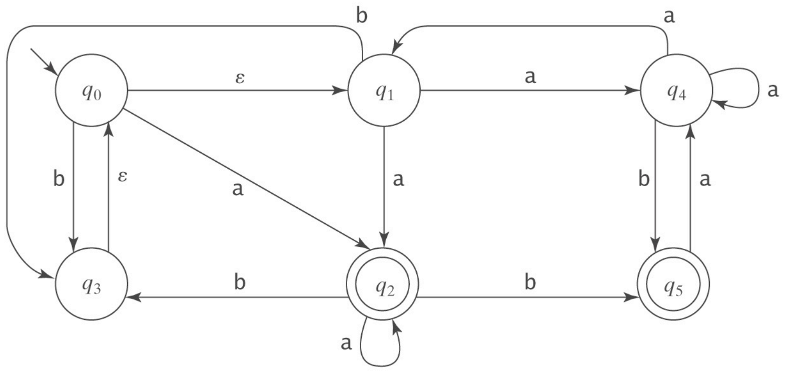
(a)

b)



(b)

c)

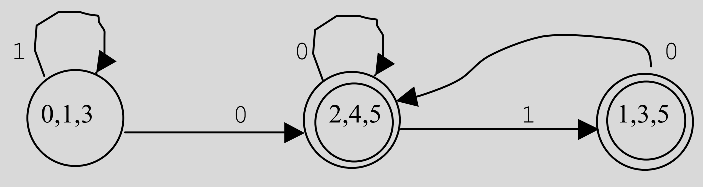


(c)

a)

s	*eps*(*s*)
q_0	$\{q_0, q_1, q_3\}$
q_1	$\{q_1, q_3\}$
q_2	$\{q_2\}$
q_3	$\{q_3\}$
q_4	$\{q_2, q_4, q_5\}$
q_5	$\{q_5\}$

$\{q_0, q_1, q_3\}$	0	$\{q_2, q_4, q_5\}$
	1	$\{q_0, q_1, q_3\}$
$\{q_2, q_4, q_5\}$	0	$\{q_2, q_4, q_5\}$
	1	$\{q_1, q_3, q_5\}$
$\{q_1, q_3, q_5\}$	0	$\{q_2, q_4, q_5\}$
	1	{ }



b)

c)

s	***eps*(*s*)**
q_0	$\{q_0, q_1\}$
q_1	$\{q_1\}$
q_2	$\{q_2\}$
q_3	$\{q_3, q_0, q_1\}$
q_4	$\{q_4\}$
q_5	$\{q_5\}$

$\{q_0, q_1\}$	a	$\{q_2, q_4\}$
	b	$\{q_0, q_1, q_3\}$
$\{q_2, q_4\}$	a	$\{q_1, q_2, q_4\}$
	b	$\{q_0, q_1, q_3, q_5\}$
$\{q_0, q_1, q_3\}$	a	$\{q_2, q_4,\}$
	b	$\{q_0, q_1, q_3\}$
$\{q_1, q_2, q_4\}$	a	$\{q_1, q_2, q_4\}$
	b	$\{q_0, q_1, q_3, q_5\}$
$\{q_0, q_1, q_3, q_5\}$	a	$\{q_2, q_4\}$
	b	$\{q_0, q_1, q_3\}$

Accepting state is $\{q_0, q_1, q_3, q_5\}$. 1

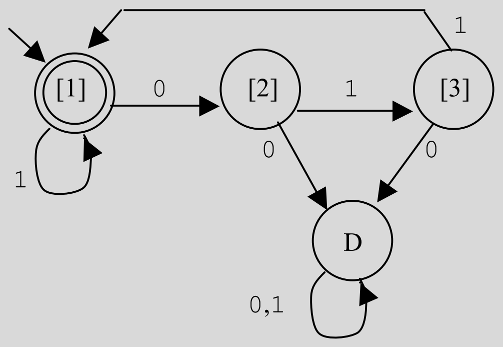
11 For each of the following languages L:

(i) Describe the equivalence classes of $\approx_L$.

(ii) If the number of equivalence classes of $\approx_L$ is finite, construct the minimal DFSM that accepts L.

a. $\{w \in \{0, 1\}^* :$ every 0 in w is immediately followed by the string 11$\}$.

[1] {in L}
[2] {otherwise in L except ends in 0}
[3] {otherwise in L except ends in 01}
[D] {corresponds to the Dead state: string contains at least one instance of 00 or 010}



b. $\{w \in \{0, 1\}^* : w$ has either an odd number of 1's and an odd number of 0's or it has an even number of 1's and an even number of 0's$\}$.

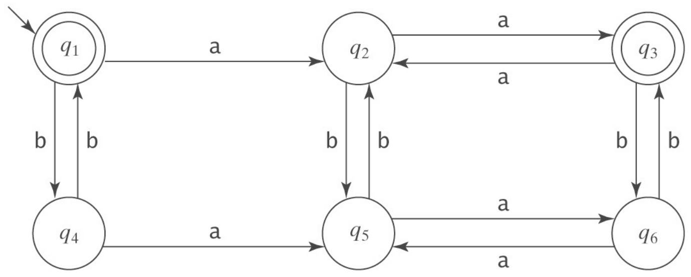
c. $\{w \in \{a, b\}^* : w$ contains at least one occurrence of the string aababa$\}$.

d. $\{ww^R : w \in \{a, b\}^*\}$.

[1] {ε}	in *L*
[2] {a}	
[3] {b}	
[4] {aa}	in *L*
[5] {ab}	

And so forth. Every string is in a different equivalence class because each could become in *L* if followed by the reverse of itself but not if followed by most other strings. This language is not regular.

12 Let *M* be the following DFSM. Use *minDFSM* to minimize *M*.



Initially, *classes* = {[1, 3], [2, 4, 5, 6]}.

At step 1:

((1, a), [2, 4, 5, 6])	((3, a), [2, 4, 5, 6])		No splitting required here.
((1, b), [2, 4, 5, 6])	((3, b), [2, 4, 5, 6])		
((2, a), [1, 3])	((4, a), [2, 4, 5, 6])	((5, a), [2, 4, 5, 6])	((6, a), [2, 4, 5, 6])
((2, b), [2, 4, 5, 6])	((4, b), [1, 3])	((5, b), [2, 4, 5, 6])	((6, b), [1, 3])

These split into three groups: [2], [4, 6], and [5]. So classes is now {[1, 3], [2], [4, 6], [5]}.

At step 2, we must consider [4, 6]:

((4, a), [5])	((6, a), [5])
((4, b), [1])	((6, b), [1])

No further splitting is required. The minimal machine has the states: {[1, 3], [2], [4, 6], [5]}, with transitions as shown above.

6 Regular Expressions

2 Write a regular expression to describe each of the following languages:

a) $\{w \in \{a, b\}^* :$ every a in w is immediately preceded and followed by b$\}$.

(b ∪ bab)*

b) $\{w \in \{a, b\}^* : w$ does not end in ba$\}$.

ε ∪ a ∪ (a ∪ b)* (b ∪ aa)

c) $\{w \in \{0, 1\}^* : \exists y \in \{0, 1\}^* (|xy|$ is even$)\}$.

(0 ∪ 1)*

d) $\{w \in \{0, 1\}^* : w$ corresponds to the binary encoding, without leading 0's, of natural numbers that are evenly divisible by 4$\}$.

(1(0 ∪ 1)* 00) ∪ 0

e) $\{w \in \{0, 1\}^* : w$ corresponds to the binary encoding, without leading 0's, of natural numbers that are powers of 4$\}$.

1(00)*

f) $\{w \in \{0\text{-}9\}^* : w$ corresponds to the decimal encoding, without leading 0's, of an odd natural number$\}$.

(ε ∪ ((1-9)(0-9)*))(1 ∪ 3 ∪ 5 ∪ 7 ∪ 9)

g) $\{w \in \{0, 1\}^* : w$ has 001 as a substring$\}$.

(0 ∪ 1)* 001 (0 ∪ 1)*

h) $\{w \in \{0, 1\}^* : w$ does not have 001 as a substring$\}$.

(1 ∪ 01)* 0*

i) $\{w \in \{a, b\}^* : w$ has bba as a substring$\}$.

(a ∪ b)* bba (a ∪ b)*

j) $\{w \in \{a, b\}^* : w$ has both aa and bb as substrings$\}$.

(a ∪ b)* aa (a ∪ b)* bb (a ∪ b)* ∪ (a ∪ b)* bb (a ∪ b)* aa (a ∪ b)*

k) $\{w \in \{a, b\}^* : w$ has both aa and aba as substrings$\}$.

(a ∪ b)* aa (a ∪ b)* aba (a ∪ b)* ∪
(a ∪ b)* aba (a ∪ b)* aa (a ∪ b)* ∪
(a ∪ b)* aaba (a ∪ b)* ∪
(a ∪ b)* abaa (a ∪ b)*

l) $\{w \in \{a, b\}^* : w$ contains at least two b's that are not followed by an a$\}$.

(a ∪ b)* bb ∪ (a ∪ b)* bbb (a ∪ b)*

m) $\{w \in \{0, 1\}^* : w$ has at most one pair of consecutive 0's and at most one pair of consecutive 1's$\}$.

(ε ∪ 1)(01)*(ε ∪ 1) (01)* (ε ∪ (00 (ε ∪ (1 (01)* (ε ∪ 0))))) /* 11 comes first.
∪
(ε ∪ 0) (10)* (ε ∪ 0) (10)* (ε ∪ (11 (ε ∪ (0 (10)* (ε ∪ 1))))) /* 00 comes first.

n) $\{w \in \{0, 1\}^* :$ none of the prefixes of w ends in 0$\}$.

1*

o) $\{w \in \{a, b\}^* : \#_a(w) \equiv_3 0\}$.

(b*ab*ab*a)*b*

p) $\{w \in \{a, b\}^* : \#_a(w) \leq 3\}$.

b* (a ∪ ε) b* (a ∪ ε) b* (a ∪ ε) b*

q) $\{w \in \{a, b\}^* : w$ contains exactly two occurrences of the substring aa$\}$.

(b ∪ ab)*aaa(b ∪ ba)* ∪ (b ∪ ab)*aab(b ∪ ab)*aa(b ∪ ba)* (Either the two occurrences are contiguous, producing aaa, or they're not.)

r) $\{w \in \{a, b\}^* : w$ contains no more than two occurrences of the substring aa$\}$.

(b ∪ ab)*(a ∪ ε) /* 0 occurrences of the substring aa
∪
(b ∪ ab)*aa(b ∪ ba)* /* 1 occurrence of the substring aa
∪
(b ∪ ab)*aaa(b ∪ ba)* ∪ (b ∪ ab)*aab(b ∪ ab)*aa(b ∪ ba)*
/* 2 occurrences of the substring aa

s) $\{w \in \{a, b\}^* - L\}$, where $L = \{w \in \{a, b\}^* : w$ contains bba as a substring$\}$.

(a ∪ ba)* (ε ∪ b ∪ bbb*) = (a ∪ ba)*b*

t) $\{w \in \{0, 1\}^* :$ every odd length string in L begins with 11$\}$.

((0 ∪ 1)(0 ∪ 1))* ∪ 11(0 ∪ 1)*

u) $\{w \in \{0\text{-}9\}^* : w$ represents the decimal encoding of an odd natural number without leading 0's.

(ε ∪ ((1-9)(0-9)*))(1 ∪ 3 ∪ 5 ∪ 7 ∪ 9)

v) $L_1 - L_2$, where L_1 = a*b*c* and L_2 = c*b*a*.

$a^+b^+c^* \cup a^+b^*c^+ \cup a^*b^+c^+$ (The only strings that are in both L_1 and L_2 are strings composed of no more than one different letter. So those are the strings that we need to remove from L_1 to form the difference. What we're left with is strings that contain two or more distinct letters.)

3 Simplify each of the following regular expressions:

a. $(a \cup b)^* (a \cup \varepsilon) b^*$.

$(a \cup b)^*$.

b. $(\varnothing^* \cup b) b^*$.

b^*.

c. $(a \cup b)^* a^* \cup b$.

$(a \cup b)^*$.

d. $((a \cup b)^*)^*$.

$(a \cup b)^*$.

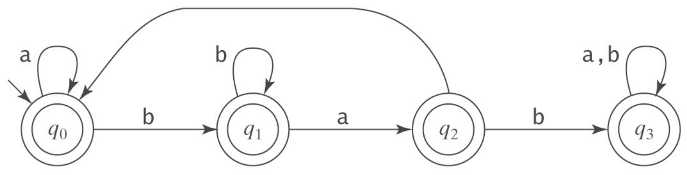
e. $((a \cup b)^+)^*$.

$(a \cup b)^*$.

f. $a ((a \cup b)(b \cup a))^* \cup a ((a \cup b) a)^* \cup a ((b \cup a) b)^*$.

$a ((a \cup b)(b \cup a))^*$.

9 Consider the following FSM M:



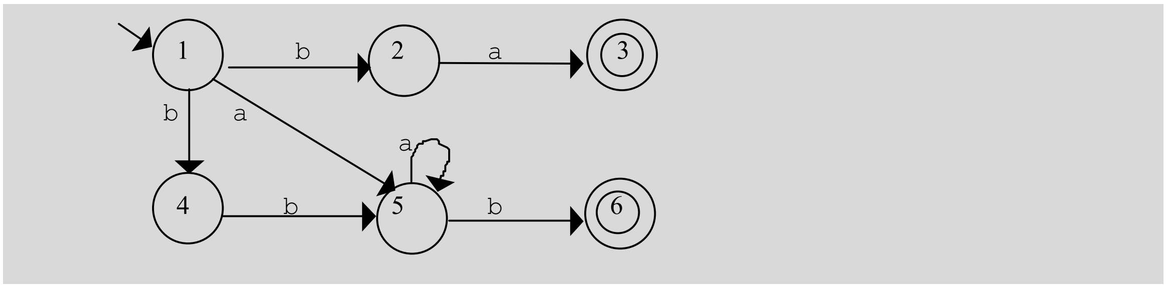
The first printing of the book has two mistakes in this figure: The transition from q_2 back to q_0 should be labeled a, and state q_3 should not be accepting. We'll give answers for the printed version (in which we'll simply ignore the unlabeled transition from q_2 to q_0) and the correct version.

a. Show a regular expression for $L(M)$.

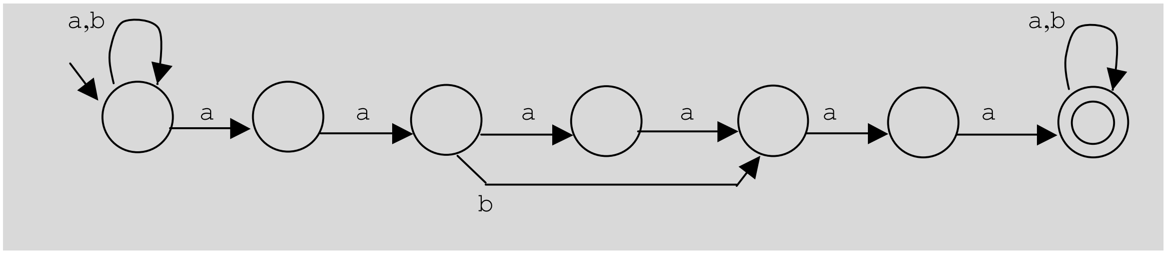
Printed version: $a^* \cup a^*bb^* \cup a^*bb^*a \cup a^*bb^*ab\,(a \cup b)^*$
Correct version: $(a \cup bb^*aa)^* (\varepsilon \cup bb^*(a \cup \varepsilon))$.

13 Show a possibly nondeterministic FSM to accept the language defined by each of the following regular expressions:

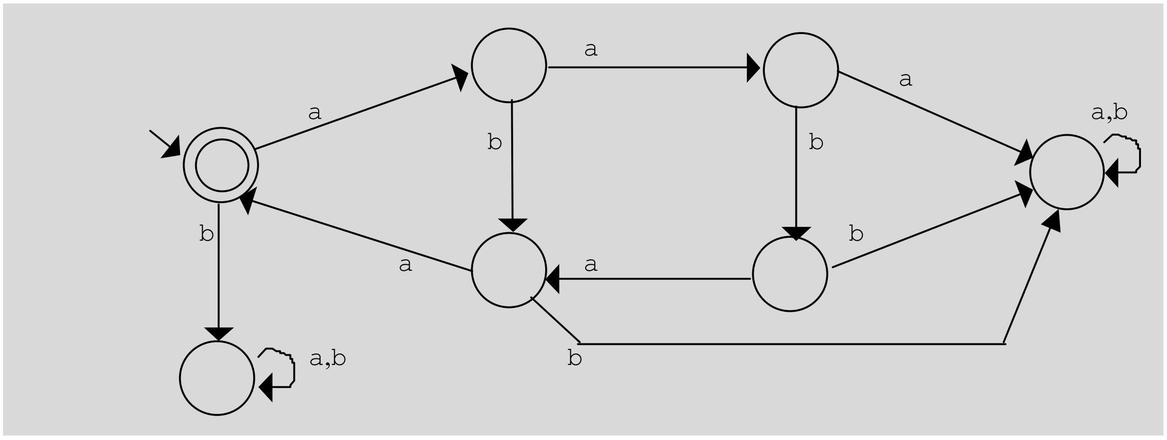
a) (((a ∪ ba) b ∪ aa)*.
b) (b ∪ ε)(ab)*(a ∪ ε).
c) (babb* ∪ a) *.
d) (ba ∪ ((a ∪ bb) a*b)).



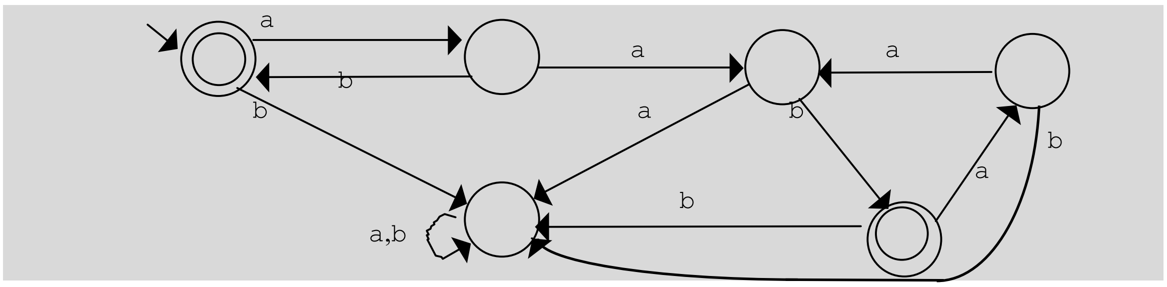
e) (a ∪ b)* aa (b ∪ aa) bb (a ∪ b)*.



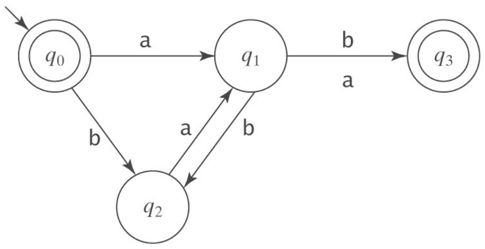
14 Show a DFSM to accept the language defined by each of the following regular expressions:
a. (aba ∪ aabaa)*



b. (ab)*(aab)*



15 Consider the following FSM M:



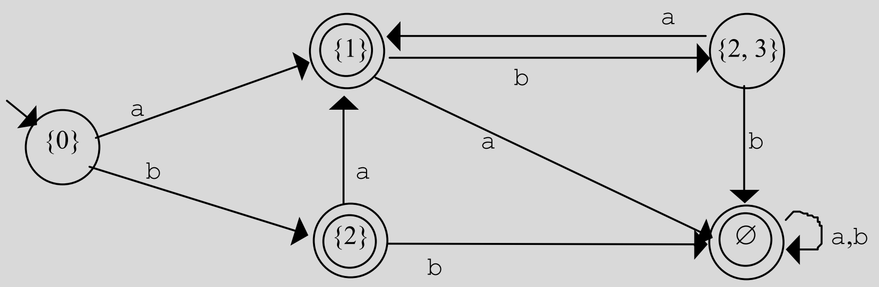
The first printing of the book has a mistake in this figure: There should not be an a labeling the transition from q_1 to q_3.

a. Write a regular expression that describes $L(M)$.

Printed (in book) version: $\varepsilon \cup ((\texttt{a} \cup \texttt{ba})(\texttt{ba})^*\texttt{b} \cup \texttt{a})$
Correct version: $\varepsilon \cup ((\texttt{a} \cup \texttt{ba})(\texttt{ba})^*\texttt{b})$.

b. Show a DFSM that accepts $\neg L(M)$.

This is for the correct version (without the extra label a):



8 Regular and Nonregular Languages

1) For each of the following languages L, state whether or not L is regular. Prove your answer:

a) $\{\texttt{a}^i\texttt{b}^j : i, j \geq 0 \text{ and } i + j = 5\}$.

Regular. A simple FSM with five states just counts the total number of characters.

b) $\{\texttt{a}^i\texttt{b}^j : i, j \geq 0 \text{ and } i - j = 5\}$.

Not regular. L consists of all strings of the form `a*b*` where the number of `a`'s is five more than the number of `b`'s. We can show that L is not regular by pumping. Let $w = \texttt{a}^{k+5}\texttt{b}^k$. Since $|xy| \leq k$, y must equal $\texttt{a}^p$ for some $p > 0$. We can pump y out once, which will generate the string $\texttt{a}^{k+5-p}\texttt{b}^k$, which is not in L because the number of `a`'s is is less than 5 more than the number of `b`'s.

c) $\{\texttt{a}^i\texttt{b}^j : i, j \geq 0 \text{ and } |i - j| \equiv_5 0\}$.

Regular. Note that $i - j \equiv_5 0$ iff $i \equiv_5 j$. L can be accepted by a straightforward FSM that counts `a`'s (mod 5). Then it counts `b`'s (mod 5) and accepts iff the two counts are equal.

d) $\{w \in \{\texttt{0}, \texttt{1}, \#\}^* : w = x\#y, \text{ where } x, y \in \{\texttt{0}, \texttt{1}\}^* \text{ and } |x|\cdot|y| \equiv_5 0\}$. (Let $\cdot$ mean integer multiplication).

Regular. For $|x| \cdot |y|$ to be divisible by 5, either $|x|$ or $|y|$ (or both) must be divisible by 5. So L is defined by the following regular expression:
$((\texttt{0} \cup \texttt{1})^5)^*\#(\texttt{0} \cup \texttt{1})^* \ \cup \ (\texttt{0} \cup \texttt{1})^*\#((\texttt{0} \cup \texttt{1})^5)^*$, where $(\texttt{0} \cup \texttt{1})^5$ is simply a shorthand for writing $(\texttt{0} \cup \texttt{1})$ five times in a row.

e) $\{\texttt{a}^i\texttt{b}^j : 0 \leq i < j < 2000\}$.

Regular. Finite.

f) $\{w \in \{\texttt{Y}, \texttt{N}\}^* : w \text{ contains at least two } \texttt{Y}\text{'s and at most two } \texttt{N}\text{'s}\}$.

Regular. L can be accepted by an FSM that keeps track of the count (up to 2) of `Y`'s and `N`'s.

g) $\{w = xy : x, y \in \{\texttt{a}, \texttt{b}\}^* \text{ and } |x| = |y| \text{ and } \#_{\texttt{a}}(x) \geq \#_{\texttt{a}}(y)\}$.

Not regular, which we'll show by pumping. Let $w = \texttt{a}^k\texttt{bba}^k$. y must occur in the first `a` region and be equal to $\texttt{a}^p$ for some nonzero p. Let $q = 0$. If p is odd, then the resulting string is not in L because all strings in L have even length. If p is even it is at least 2. So both `b`'s are now in the first half of the string. That means that the number of `a`'s in the second half is greater than the number in the first half. So resulting string, $\texttt{a}^{k-p}\texttt{bba}^k$, is not in L.

h) $\{w = xyzy^{\text{R}}x : x, y, z \in \{\texttt{a}, \texttt{b}\}^*\}$.

Regular. Note that $L = (\texttt{a} \cup \texttt{b})^*$. Why? Take any string s in $(\texttt{a} \cup \texttt{b})^*$. Let x and y be ε. Then $s = z$. So the string can be written in the required form. Moral: Don't jump too fast when you see the nonregular "triggers", like ww or ww^{R}. The entire context matters.

i) $\{w = xyzy : x, y, z \in \{0, 1\}^+\}$.

Regular. The key to why this is so is to observe that we can let y be just a single character. Then the rest of w can generated by x and z. So any string w in $\{0, 1\}^+$ is in L iff:

- the last letter of w occurs in at least one other place in the string,
- that place is not the next to the last character,
- nor is it the first character, and
- w contains least 4 letters.

Either the last character is 0 or 1. So:

$$L = ((0 \cup 1)^+ \, 0 \, (0 \cup 1)^+ \, 0) \cup ((0 \cup 1)^+ \, 1 \, (0 \cup 1)^+ \, 1).$$

j) $\{w \in \{0, 1\}^* : \#_0(w) \neq \#_1(w)\}$.

Not regular. This one is quite hard to prove by pumping. Since so many strings are in L, it's hard to show how to pump and get a string that is guaranteed not to be in L. Generally, with problems like this, you want to turn them into problems involving more restrictive languages to which it is easier to apply pumping. So: if L were regular, then the complement of L, L' would also be regular.

$$L' = \{w \in \{0, 1\}^* : \#_0(w) = \#_1(w)\}.$$

It is easy to show, using pumping, that L' is not regular: Let $w = 0^k1^k$. y must occur in the initial string of 0's, since $|xy| \leq k$. So $y = 0^i$ for some $i \geq 1$. Let q of the pumping theorem equal 2 (i.e., we will pump in one extra copy of y). We now have a string that has more 0's than 1's and is thus not in L'. Thus L' is not regular. So neither is L. Another way to prove that L' isn't regular is to observe that, if it were, $L'' = L' \cap 0^*1^*$ would also have to be regular. But L'' is 0^n1^n, which we already know is not regular.

k) $\{w \in \{a, b\}^* : w = w^R\}$.

Not regular, which we show by pumping. Let $w = a^kb^kb^ka^k$. So y must be a^p for some nonzero p. Pump in once. Reading w forward there are more a's before any b's than there are when w is read in reverse. So the resulting string is not in L.

l) $\{w \in \{a, b\}^* : \exists x \in \{a, b\}^+ \, (w = x \, x^R \, x)\}$.

Not regular, which we show by pumping: Let $w = a^kbba^ka^kb$. y must occur in the initial string of a's, since $|xy| \leq k$. So $y = a^i$ for some $i \geq 1$. Let q of the pumping theorem equal 2 (i.e., we will pump in one extra copy of y). That generates the string $a^{k+i}bba^ka^kb$. If this string is in L, then we must be able to divide it into thirds so that it is of the form $x \, x^R \, x$. Since its total length is $3k + 3 + i$, one third of that (which must be the length of x) is $k + 1 + i/3$. If i is not a multiple of 3, then we cannot carve it up into three equal parts. If i is a multiple of 3, we can carve it up. But then the right boundary of x will shift two characters to the left for every three a's in y. So, if i is just 3, the boundary will shift so that x no longer contains any b's If i is more than 3, the boundary will shift even farther away from the first b. But there are b's in the string. Thus the resulting string cannot be in L. Thus L is not regular.

m) $\{w \in \{a, b\}^*$: the number of occurrences of the substring ab equals the number of occurrences of the substring ba$\}$.

Regular. The idea is that it's never possible for the two counts to be off by more than 1. For example, as soon as there's an ab, there can be nothing but b's without producing the first ba. Then the two counts are equal and will stay equal until the next b. Then they're off by 1 until the next a, when they're equal again. $L = a^* \cup a^+b^+a^+(b^+a^+)^* \cup b^* \cup b^+a^+b^+(a^+b^+)^*$.

n) $\{w \in$ {a, b}* : w contains exactly two more b's than a's}.

Not regular, which we'll show by pumping. Let w = a^{k}b^{k+2}. y must equal a^p for some $p > 0$. Set q to 0 (i.e., pump out once). The number of a's changes, but the number of b's does not. So there are no longer exactly 2 more b's than a's.

o) $\{w \in$ {a, b}* : $w = xyz$, $|x| = |y| = |z|$, and $z = x$ with every a replaced by b and every b replaced by a}. Example: abbbabbaa $\in L$, with x = abb, y = bab, and z = baa.

Not regular, which we'll show by pumping. Let w = a^{k}a^{k}b^k. This string is in L since x = a^k, y = a^k, and z = b^k. y (from the pumping theorem) = a^p for some nonzero p. Let $q = 2$ (i.e., we pump in once). If p is not divisible by 3, then the resulting string is not in L because it cannot be divided into three equal length segments. If $p = 3i$ for integer i, then, when we divide the resulting string into three segments of equal length, each segment gets longer by i characters. The first segment is still all a's, so the last segment must remain all b's. But it doesn't. It grows by absorbing a's from the second segment. Thus z no longer = x with every a replaced by b and every b replaced by a. So the resulting string is not in L.

p) $\{w$: $w \in$ {a – z}* and the letters of w appear in reverse alphabetical order}. For example, spoonfeed $\in L$.

Regular. L can be recognized by a straightforward 26-state FSM.

q) $\{w$: $w \in$ {a - z}* every letter in w appears at least twice}. For example, unprosperousness$\in L$.

Regular. L can be recognized by an FSM with 26^3 states.

r) $\{w$: w is the decimal encoding of a natural number in which the digits appear in a non-decreasing order without leading zeros}.

Regular. L can be recognized by an FSM with 10 states that checks that the digits appear in the correct order. Or it can be described by the regular expression: 0*1*2*3*4*5*6*7*8*9*.

s) $\{w$ of the form: <*integer*$_1$>+<*integer*$_2$>=<*integer*$_3$>, where each of the substrings <*integer*$_1$>, <*integer*$_2$>, and <*integer*$_3$> is an element of {0 - 9}* and *integer*$_3$ is the sum of *integer*$_1$ and *integer*$_2$}. For example, 124+5=129 $\in L$.

Not regular.

t) L_0*, where L_0 = {ba^{i}b^{j}a^k, $j \geq 0$, $0 \leq i \leq k$}.

Regular. Both i and j can be 0. So L = (b$^+$a*)*.

u) $\{w$: w is the encoding (in the scheme we describe next) of a date that occurs in a year that is a prime number}. A date will be encoded as a string of the form *mm*/*dd*/*yyyy*, where each m, d, and y is drawn from {0-9}.

Regular. Finite.

v) $\{w \in$ {1}* : w is, for some $n \geq 1$, the unary encoding of 10^n}. (So L = {1111111111, 1^{100}, 1^{1000}, …}.)

Not regular, which we can prove by pumping. Let w = 1^t, where t is the smallest integer that is a power of ten and is greater than k. y must be 1^p for some nonzero p. Clearly, p can be at most t. Let $q = 2$ (i.e., pump in once). The length of the resulting string s is at most $2t$. But the next power of 10 is $10t$. Thus s cannot be in L.

2) For each of the following languages L, state whether L is regular or not and prove your answer:
   a) $\{w \in \{\texttt{a}, \texttt{b}, \texttt{c}\}^* : \text{in each prefix } x \text{ of } w, \#_{\texttt{a}}(x) = \#_{\texttt{b}}(x) = \#_{\texttt{c}}(x))\}$.

   Regular. $L = \{\varepsilon\}$.

   b) $\{w \in \{\texttt{a}, \texttt{b}, \texttt{c}\}^* : \exists \text{ some prefix } x \text{ of } w\ (\#_{\texttt{a}}(x) = \#_{\texttt{b}}(x) = \#_{\texttt{c}}(x))\}$.

   Regular. $L = \Sigma^*$, since every string has ε as a prefix.

   c) $\{w \in \{\texttt{a}, \texttt{b}, \texttt{c}\}^* : \exists \text{ some prefix } x \text{ of } w\ (x \neq \varepsilon \text{ and } \#_{\texttt{a}}(x) = \#_{\texttt{b}}(x) = \#_{\texttt{c}}(x))\}$.

   Not regular, which we prove by pumping. Let $w = \texttt{a}^{2k}\texttt{b}\texttt{a}^{2k}$.

3) Define the following two languages:
   $L_{\texttt{a}} = \{w \in \{\texttt{a}, \texttt{b}\}^* : \text{in each prefix } x \text{ of } w, \#_{\texttt{a}}(x) \geq \#_{\texttt{b}}(x)\}$.
   $L_{\texttt{b}} = \{w \in \{\texttt{a}, \texttt{b}\}^* : \text{in each prefix } x \text{ of } w, \#_{\texttt{b}}(x) \geq \#_{\texttt{a}}(x)\}$.
   a) Let $L_1 = L_{\texttt{a}} \cap L_{\texttt{b}}$. Is L_1 regular? Prove your answer.

   Regular. $L_1 = \{\varepsilon\}$.

   b) Let $L_2 = L_{\texttt{a}} \cup L_{\texttt{b}}$. Is L_2 regular? Prove your answer.

   Not regular. First, we observe that $L_2 = \{\varepsilon\} \cup \{w\text{: the first character of } w \text{ is an } \texttt{a} \text{ and } w \in L_{\texttt{a}}\}$
   $\cup \{w\text{: the first character of } w \text{ is a } \texttt{b} \text{ and } w \in L_{\texttt{b}}\}$

   We can show that L_2 is not regular by pumping. Let $w = \texttt{a}^{2k}\texttt{b}^{2k}$. y must be $\texttt{a}^p$ for some $0 < p \leq k$. Pump out. The resulting string $w' = \texttt{a}^{2k-p}\texttt{b}^{2k}$. Note that w' is a prefix of itself. But it is not in L_2 because it is not ε, nor is it in $L_{\texttt{a}}$ (because it has more b's than a's) or in $L_{\texttt{b}}$ (because it has a as a prefix).

4) For each of the following languages L, state whether L is regular or not and prove your answer:
   a) $\{uww^{R}v : u, v, w \in \{\texttt{a}, \texttt{b}\}^{+}\}$.

   Regular. Every string in L has at least 4 characters. Let w have length 1. Then ww^{R} is simply two identical characters next to each other. So L consists of exactly those strings of at least four characters such that there's a repeated character that is not either the first or last. Any such string can be rewritten as u (all the characters up to the first repeated character) w (the first repeated character) w^{R} (the second repeated character) v (all the rest of the characters). So $L = (\texttt{a} \cup \texttt{b})^{+}\ (\texttt{aa} \cup \texttt{bb})\ (\texttt{a} \cup \texttt{b})^{+}$.

b) $\{xyzy^R x : x, y, z \in \{\texttt{a}, \texttt{b}\}^+\}$.

Not regular, which we show by pumping. Let $w = \texttt{a}^k\texttt{babaa}^k\texttt{b}$. Note that w is in L because, using the letters from the language definition, $x = \texttt{a}^k\texttt{b}$, $y = \texttt{a}$, and $z = \texttt{b}$. Then y (from the Pumping Theorem) must occur in the first a region. It is $\texttt{a}^p$ for some nonzero p. Set q to 2 (i.e., pump in once). The resulting string is $\texttt{a}^{k+p}\texttt{babaa}^k\texttt{b}$. This string cannot be in L. Since its initial x (from the language definition) region starts with a, there must be a final x region that starts with a. Since the final x region ends with a b, the initial x region must also end with a b. So, thinking about the beginning of the string, the shortest x region is $\texttt{a}^{k+p}\texttt{b}$. But there is no such region at the end of the string unless p is 1. But even in that case, we can't call the final $\texttt{aa}^k\texttt{b}$ string x because that would leave only the middle substring ab to be carved up into yzy^R. But since both y and z must be nonempty, yzy^R must have at least three characters. So the resulting string cannot be carved up into $xyzy^Rx$ and so is not in L.

21 For each of the following claims, state whether it is *True* or *False*. Prove your answer.:

c) There are uncountably many non-regular languages over $\Sigma = \{\texttt{a}, \texttt{b}\}$.

True. There are uncountably many languages over $\Sigma = \{\texttt{a}, \texttt{b}\}$. Only a countably infinite number of those are regular.

d) The union of an infinite number of regular languages must be regular.

False. Let $L = \cup (\{\varepsilon\}, \{\texttt{ab}\}, \{\texttt{aabb}\}, \{\texttt{aaabbb}\}, \ldots)$ Each of these languages is finite and thus regular. But the infinite union of them is $\{\texttt{a}^n\texttt{b}^n, n \geq 0\}$, which is not regular.

e) The union of an infinite number of regular languages is never regular.

Nothing says the languages that are being unioned have to be different. So, Let $L = \cup (\texttt{a}^*, \texttt{a}^*, \texttt{a}^*, \ldots)$, which is $\texttt{a}^*$, which is regular.

f) If L_1 and L_2 are not regular languages, then $L_1 \cup L_2$ is not regular.

False. Let $L_1 = \{\texttt{a}^p : p \text{ is prime}\}$. Let $L_2 = \{\texttt{a}^p : p \text{ is greater than 0 and not prime}\}$. Neither L_1 nor L_2 is regular. But $L_1 \cup L_2 = \texttt{a}^+$, which is regular.

g) If L_1 and L_2 are regular languages, then $L_1 \otimes L_2 = \{w : w \in (L_1 - L_2) \text{ or } w \in (L_2 - L_1)\}$ is regular.

True. The regular languages are closed under union and set difference.

h) If L_1 and L_2 are regular languages and $L_1 \subseteq L \subseteq L_2$, then L must be regular.

False. Let $L_1 = \varnothing$. Let $L_2 = \{\texttt{a} \cup \texttt{b})^*$. Let $L = \{\texttt{a}^n\texttt{b}^n : n \geq 0\}$, which is not regular.

i) The intersection of a regular language and a nonregular language must be regular.

False. Let $L_1 = (\texttt{a} \cup \texttt{b})^*$, which is regular. Let $L_2 = \{\texttt{a}^n\texttt{b}^n, n \geq 0\}$, which is not regular. Then $L_1 \cap L_2 = \{\texttt{a}^n\texttt{b}^n, n \geq 0\}$, which is not regular. A simpler example is: Let $L_1 = \texttt{a}^*$, which is regular. Let $L_2 = \{\texttt{a}^p : p \text{ is prime}\}$, which is not regular. $L_1 \cap L_2 = L_2$.

j) The intersection of a regular language and a nonregular language must not be regular.

False. Let $L_1 = \{\texttt{ab}\}$ (regular). Let $L_2 = \{\texttt{a}^n\texttt{b}^n, n \geq 0\}$ (not regular). $L_1 \cap L_2 = \{\texttt{ab}\}$, which is regular.

k) The intersection of two nonregular languages must not be regular.

False. Let $L_1 = \{\mathtt{a}^p: p \text{ is prime}\}$, which is not regular. Let $L_2 = \{\mathtt{b}^p: p \text{ is prime}\}$, which is also not regular. $L_1 \cap L_2 = \varnothing$, which is regular.

l) The intersection of a finite number of nonregular languages must not be regular.

False. Since two is a finite number, we can used the same counterexample that we used above in part i. Let $L_1 = \{\mathtt{a}^p: p \text{ is prime}\}$, which is not regular. Let $L_2 = \{\mathtt{b}^p: p \text{ is prime}\}$, which is also not regular. $L_1 \cap L_2 = \varnothing$, which is regular.

m) It is possible that the concatenation of two nonregular languages is regular.

True. To prove this, we need one example: Let $L_1 = \{\mathtt{a}^i\mathtt{b}^j,\ i, j \geq 0 \text{ and } i \neq j\}$. Let $L_2 = \{\mathtt{b}^n\mathtt{a}^m,\ n, m \geq 0$ and $n \neq m$. $L_1L_2 = \{\mathtt{a}^i\mathtt{b}^j\mathtt{a}^k$ such that:

- If i and k are both 0 then $j \geq 2$.
- If $i = 0$ and $k \neq 0$ then $j \geq 1$.
- If $i \neq 0$ and $k = 0$ then $j \geq 1$.
- If i and k are both 1 then $j \neq 1$.
- Otherwise, $j \geq 0\}$.

In other words, L_1L_2 is almost `a*b*a*`, with a small number of exceptions that can be checked for by a finite state machine.

n) It is possible that the union of a regular language and a nonregular language is regular.

True. Let $L_1 = \mathtt{a}^*$, which is regular. Let $L_2 = \{\mathtt{a}^p: 2 \leq p \text{ and } p \text{ is prime}\}$, which is not regular. $L_1 \cup L_2 = \mathtt{a}^*$, which is regular.

o) If L is a language that is not regular, then L^* is not regular.

False.
Let $L = \text{Prime}_\mathtt{a} = \{\mathtt{a}^n : n \text{ is prime}\}$. L is not regular.
$L^* = \{\varepsilon\} \cup \{\mathtt{a}^n : 1 < n\}$. L^* is regular.

p) If L^* is regular, then L is regular.

False.
Let $L = \text{Prime}_\mathtt{a} = \{\mathtt{a}^n : n \text{ is prime}\}$. L is not regular.
$L^* = \{\varepsilon\} \cup \{\mathtt{a}^n : 1 < n\}$. L^* is regular.

q) The nonregular languages are closed under intersection.

False.
Let $L_1 = \{\mathtt{a}^n\mathtt{b}^n$, where n is even$\}$. L_1 is not regular.
Let $L_2 = \{\mathtt{a}^n\mathtt{b}^n$, where n is odd$\}$. L_2 is not regular.
$L = L_1 \cap L_2 = \varnothing$, which is regular.

r) Every subset of a regular language is regular.

False.
Let $L = \mathtt{a}^*$, which is regular.
Let $L' = \mathtt{a}^p$, where p is prime. L' is not regular, but it is a subset of L.

s) Let $L_4 = L_1L_2L_3$. If L_1 and L_2 are regular and L_3 is not regular, it is possible that L_4 is regular.

True. Example:

Let $L_1 = \{\varepsilon\}$.	L_1 is regular.
Let $L_2 = a^*$.	L_2 is regular.
Let $L_3 = a^p$, where p is prime.	L_3 is not regular.
$L_4 = a^k$, where $k \geq 2$	L_4 is regular, because it is defined by aaa*.

t) If L is regular, then so is $\{xy : x \in L \text{ and } y \notin L\}$.

True. $\{xy : x \in L \text{ and } y \notin L\}$ can also be described as the concatenation of L and $\neg L$. Since the regular languages are closed under complement, this means that it is the concatenation of two regular languages. The regular languages are closed under complement.

u) Every infinite regular language properly contains another infinite regular language.

True. Let L be an infinite regular language and let w be a string in L. Then $L - \{w\}$ is also both infinite and regular.

9 Decision Procedures for Regular Languages

1) Define a decision procedure for each of the following questions. Argue that each of your decision procedures gives the correct answer and terminates.

a) Given two DFSMs M_1 and M_2, is $L(M_1) = L(M_2)^R$?

1. From M_1, construct M_R, which accepts $L(M_1)^R$, using the algorithm presented in the solution to Exercise 8.7 (c).
2. Determine whether M_1 and M_2 accept the same language, using the algorithm *equalFSMs*, presented in Section 9.1.4.

b) Given two DFSMs M_1 and M_2 is $|L(M_1)| < |L(M_2)|$?

1. If $L(M_1)$ is infinite, return *False*.
2. If $L(M_2)$ is infinite, return *True*.

/* Now we know that both languages are finite. We need to count the number of elements in each.

3. Run every string in Σ_{M1}* of length less than $|K_{M1}|$ through M_1, counting the number that are accepted. Call that number C_1.
4. Run every string in Σ_{M2}* of length less than $|K_{M2}|$ through M_2, counting the number that are accepted. Call that number C_2.
5. If $C_1 < C_2$ then return *True*. Else return *False*.

c) Given a regular grammar G and a regular expression α, is $L(G) = L(\alpha)$?

1. From G, create a NDFSM M_G that accepts $L(G)$.
2. From α, create an FSM M_α that accepts $L(\alpha)$.
3. Determine whether M_G and M_α are equivalent.

d) Given two regular expressions, α and β, do there exist any even length strings that are in $L(\alpha)$ but not $L(\beta)$?

1. From α, build FSM M_α, such that $L(M_\alpha) = L(\alpha)$.
2. From β, build FSM M_β, such that $L(M_\beta) = L(\beta)$.
3. Build M_{even}, such that $L(M_{\text{even}})$ = the set of all even length strings over the alphabet Σ_α.
4. Build M'_α, such that $L(M'_\alpha) = L(M_\alpha) \cap L(M_{\text{even}})$.
5. Build $M'_{\neg\beta}$ to accept the complement of $L(M_\beta)$.
6. Build M_F, such that $L(M_F) = L(M'_\alpha) \cap L(M'_{\neg\beta})$.
7. If $L(M_F)$ is not empty, return *True*. Else return *False*.

e) Let Σ = {a, b} and let α be a regular expression. Does the language generated by α contains all the even length strings in Σ*.

1. From α, build FSM M_α, such that $L(M_\alpha) = L(\alpha)$.
2. Build M_1 to accept all even length strings over Σ.
4. Build M_2 to accept $L(M_1)$ - $L(M_\alpha)$.
5. See if $L(M_2)$ is empty. If it is, return *True*. Else return *False*.

f) Given an FSM M and a regular expression α, is it true that $L(M)$ and $L(\alpha)$ are both finite and M accepts exactly two more strings than α generates?

1. From α, build FSM P, such that $L(P) = L(\alpha)$.
2. If $L(M)$ is infinite, return *False*.
3. If $L(P)$ is infinite, return *False*.

/* Now we know that both languages are finite. Thus neither machine accepts any strings of length equal to or greater than the number of states it contains. So we simply count the number of such "short" strings that each machine accepts.

4. Run every string in Σ_M* of length less than $|K_M|$ through M, counting the number that are accepted. Call that number C_M.
5. Run every string in Σ_P* of length less than $|K_P|$ through P, counting the number that are accepted. Call that number C_P.
6. If $C_M = C_P+2$, return *True*; else return *False*.