6 Regular Expressions

1) Describe in English, as briefly as possible, the language defined by each of these regular expressions:

   a) (b ∪ ba) (b ∪ a)* (ab ∪ b).

      The set of strings over the alphabet {a, b} that start and end with b.

   b) (((a*b*)*ab) ∪ ((a*b*)*ba))(b ∪ a)*.

      The set of strings over the alphabet {a, b} that contain at least one occurrence of ab or ba.

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

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

      (b ∪ bab)*

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

      ε ∪ a ∪ (a ∪ b)* (b ∪ aa)

   c) {*w* ∈ {0, 1}* : ∃*y* ∈ {0, 1}* (|*xy*| is even)}.

      (0 ∪ 1)*

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

      1(00)*

   f) {*w* ∈ {0-9}* : *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* ∈ {0, 1}* : *w* has 001 as a substring}.

      (0 ∪ 1)* 001 (0 ∪ 1)*

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

      (1 ∪ 01)* 0*

   i) {*w* ∈ {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$\}$.

$(a \cup b)^* aa (a \cup b)^* bb (a \cup b)^* \cup (a \cup b)^* bb (a \cup b)^* aa (a \cup b)^*$

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

$(a \cup b)^* aa (a \cup b)^* aba (a \cup b)^* \quad \cup$
$(a \cup b)^* aba (a \cup b)^* aa (a \cup b)^* \quad \cup$
$(a \cup b)^* aaba (a \cup b)^* \quad \cup$
$(a \cup b)^* abaa (a \cup b)^*$

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

$(a \cup b)^* bb \cup (a \cup b)^* bbb (a \cup b)^*$

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

$(\varepsilon \cup 1)(01)^*(\varepsilon \cup 1)(01)^* (\varepsilon \cup (00 (\varepsilon \cup (1 (01)^* (\varepsilon \cup 0)))))$ /* 11 comes first.
$\cup$
$(\varepsilon \cup 0)(10)^* (\varepsilon \cup 0)(10)^* (\varepsilon \cup (11 (\varepsilon \cup (0 (10)^* (\varepsilon \cup 1)))))$ /* 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) $\{w \in \{a, b\}^* : \#_a(w) \leq 3\}$.

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

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

$(b \cup ab)^*aaa(b \cup ba)^* \cup (b \cup ab)^*aab(b \cup ab)^*aa(b \cup ba)^*$ (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 $aa\}$.

$(b \cup ab)^*(a \cup \varepsilon)$ /* 0 occurrences of the substring aa
$\cup$
$(b \cup ab)^*aa(b \cup ba)^*$ /* 1 occurrence of the substring aa
$\cup$
$(b \cup ab)^*aaa(b \cup ba)^* \cup (b \cup ab)^*aab(b \cup ab)^*aa(b \cup ba)^*$
/* 2 occurrences of the substring aa

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

$(a \cup ba)^* (\varepsilon \cup b \cup bbb^*) = (a \cup ba)^*b^*$

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

$((0 \cup 1)(0 \cup 1))^* \cup 11(0 \cup 1)^*$

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

$(\varepsilon \cup ((1\text{-}9)(0\text{-}9)^*))(1 \cup 3 \cup 5 \cup 7 \cup 9)$

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.)

w) The set of legal United States zipcodes 💻.

x) The set of strings that correspond to domestic telephone numbers in your country.

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

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

4) For each of the following expressions E, answer the following three questions and prove your answer:

(i) Is E a regular expression?
(ii) If E is a regular expression, give a simpler regular expression.
(iii) Does E describe a regular language?

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

E is a regular expression. A simpler one is $(a \cup b)^*$. The language is regular.

b) $(a^+ a^n b^n)$.

E is not a regular expression. The language is not regular. It is $\{a^m b^n : m > n\}$.

c) $((ab)^* \varnothing)$.

E is a regular expression. A simpler one is $\varnothing$. The language is regular.

d) $(((ab) \cup c)^* \cap (b \cup c^*))$.

E is not a regular expression because it contains $\cap$. But it does describe a regular language (c^*) because the regular languages are closed under intersection.

e) $(\varnothing^* \cup (bb^*))$.

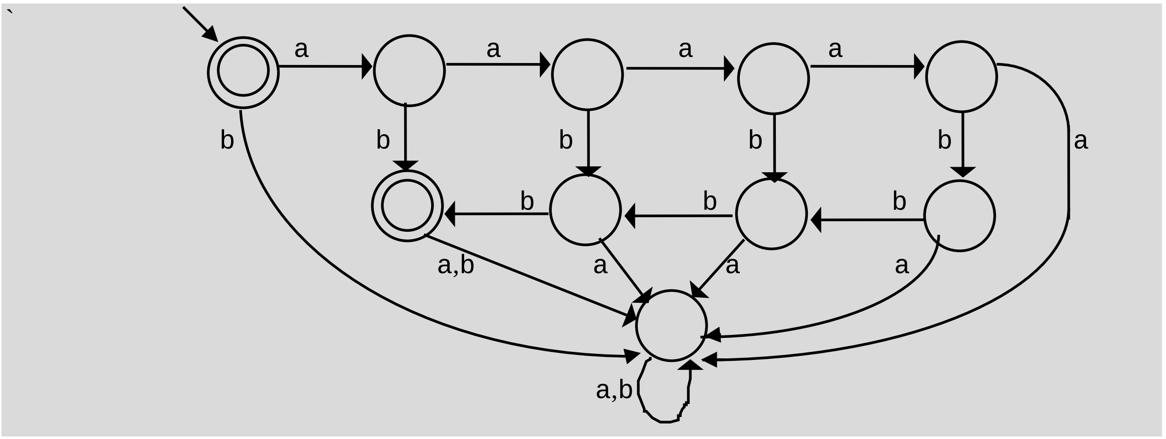
E is a regular expression. A (slightly) simpler one is $(\varepsilon \cup (bb^*))$. The language is regular.

5) Let $L = \{a^n b^n : 0 \leq n \leq 4\}$.

a) Show a regular expression for *L*.

$(\varepsilon \cup ab \cup aabb \cup aaabbb \cup aaaabbbb)$

b) Show an FSM that accepts *L*.

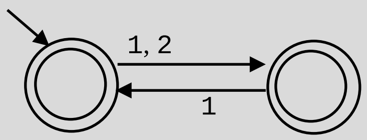


6) Let $L = \{w \in \{1, 2\}^* :$ for all prefixes *p* of *w*, if $|p| > 0$ and $|p|$ is even, then the last character of *p* is $1\}$.

a) Write a regular expression for *L*.

$((1 \cup 2)1)^* (1 \cup 2 \cup \varepsilon)$

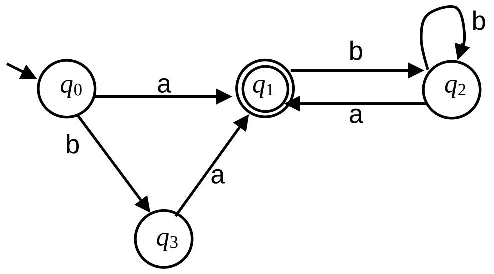
b) Show an FSM that accepts *L*.



7) Use the algorithm presented in the proof of Kleene's theorem to construct an FSM to accept the languages generated by the following regular expressions:

a) (b(b ∪ ε)b)*.
b) bab ∪ a*.

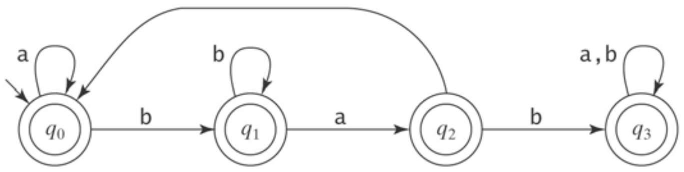
8) Let *L* be the language accepted by the following finite state machine:



Indicate, for each of the following regular expressions, whether it correctly describes *L*:
a) (a ∪ ba)bb*a.
b) (ε ∪ b)a(bb*a)*.
c) ba∪ ab*a.
d) (a ∪ ba)(bb*a)*.

a) no; b) yes; c) no; d) yes.

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* ∪ a*bb* ∪ a*bb*a ∪ a*bb*ab (a ∪ b)*
Correct version: (a ∪ bb*aa)* (ε ∪ bb*(a ∪ ε)).

b) Describe *L*(*M*) in English.

Printed version: No obvious concise description.
Correct version: All strings in {a, b}* that contain no occurrence of bab.

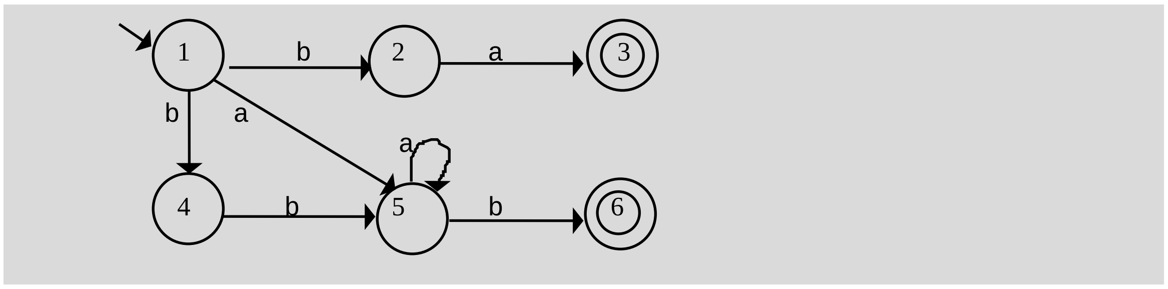
10) Consider the FSM *M* of Example 5.3. Use *fsmtoregexheuristic* to construct a regular expression that describes *L*(*M*).

11) Consider the FSM *M* of Example 6.9. Apply *fsmtoregex* to *M* and show the regular expression that results.

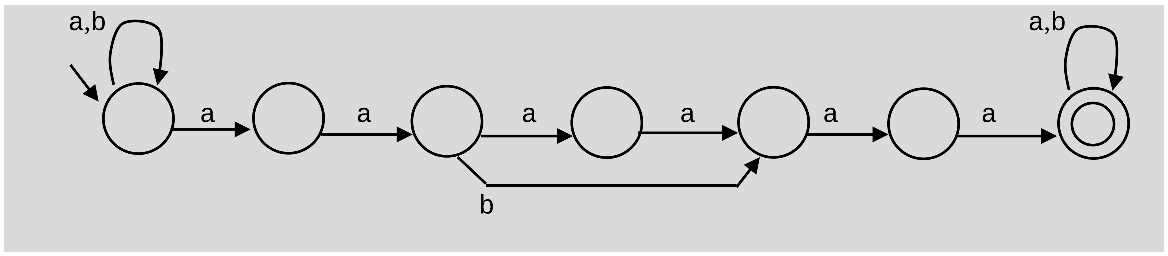
12) Consider the FSM *M* of Example 6.8. Apply *fsmtoregex* to *M* and show the regular expression that results. (Hint: this one is exceedingly tedious, but it can be done.)

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

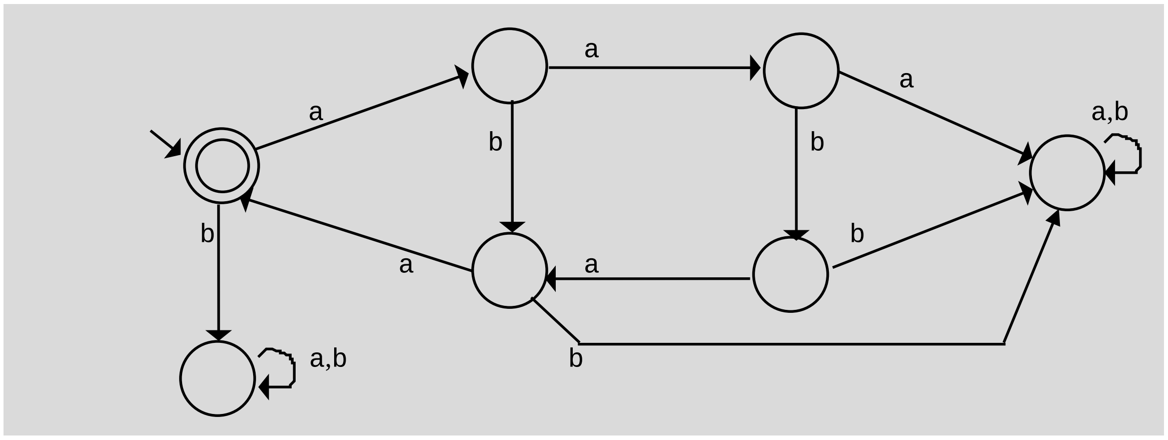
a) (((a ∪ ba) b ∪ aa)*.
b) (b ∪ ε)(ab)*(a ∪ ε).
c) (babb* ∪ a)*.
d) (ba ∪ ((a ∪ bb) a*b)).



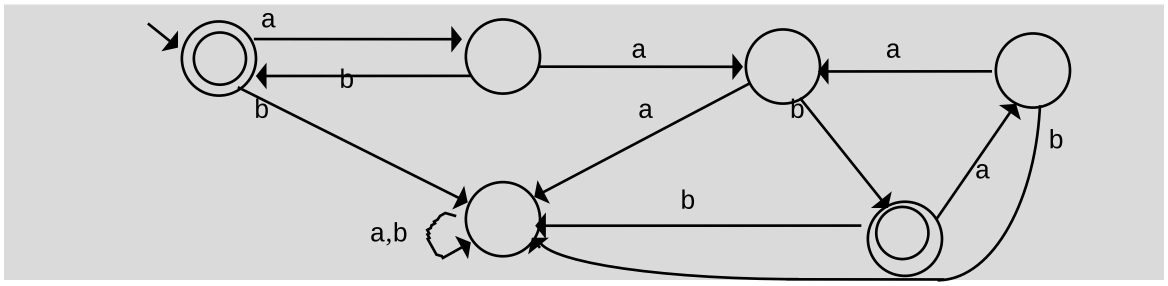
e) (a ∪ b)* aa (b ∪ aa) bb (a ∪ b)*.



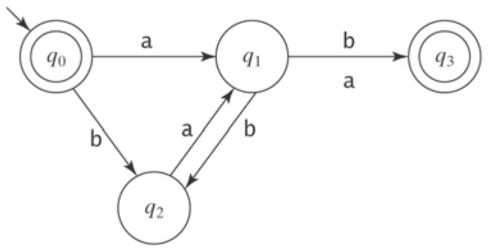
14) Show a DFSM to accept the language defined by each of the following regular expressions:
a) (aba ∪ aabaa)*



b) (ab)*(aab)*



15) Consider the following FSM *M*:



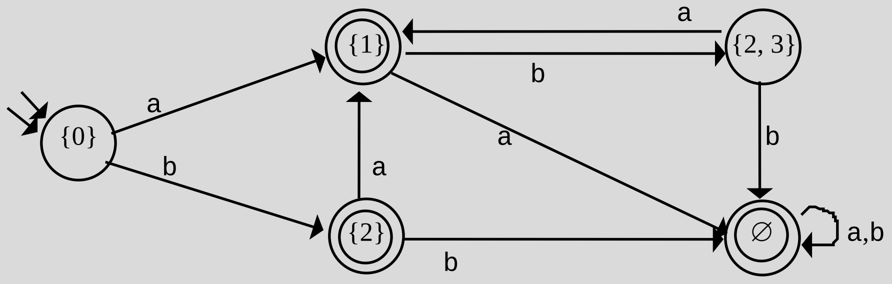
The first printing of the book has a mistake in this figure: There should not be an a labeling the transition from q_1 to q_3.

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

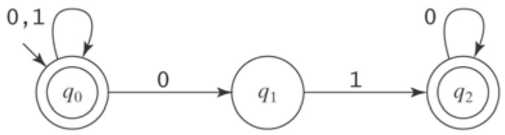
Printed version: $\varepsilon \cup$ ((a $\cup$ ba)(ba)*b $\cup$ a)
Correct version: $\varepsilon \cup$ ((a $\cup$ ba)(ba)*b).

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

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



16) Given the following DFSM M, write a regular expression that describes $\neg L(M)$:



Give a regular expression for $\neg L(M)$.

We first construct a deterministic FSM M^* equivalent to M. M^* =
$(\{\{q_0\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_0, q_1, q_2\}\}, \{0, 1\}, \delta^*, \{q_0\}, \{\{q_0, q_2\}, \{q_0, q_1, q_2\}\})$, where δ^* =
$\{(\{q_0\}, 0, \{q_0, q_1\}),$
$(\{q_0\}, 1, \{q_0\}),$
$(\{q_0, q_1\}, 0, \{q_0, q_1\}),$
$(\{q_0, q_1\}, 1, \{q_0, q_2\}),$
$(\{q_0, q_2\}, 0, \{q_0, q_1, q_2\}),$
$(\{q_0, q_2\}, 1, \{q_0\}),$
$(\{q_0, q_1, q_2\}, 0, \{q_0, q_1, q_2\}),$
$(\{q_0, q_1, q_2\}, 1, \{q_0, q_2\})\}$

From M^*, we flip accepting and nonaccepting states to build M^{**}, which accepts $\neg L(M)$. $M^{**} = M^*$ except that $A^{**} = \{\}$.

So a regular expression that accepts $L(M^{**})$ is $\varnothing$.

17) Add the keyword `able` to the set in Example 6.13 and show the FSM that will be built by *buildkeywordFSM* from the expanded keyword set.

18) Let $\Sigma = \{a, b\}$. Let $L = \{\varepsilon, a, b\}$. Let R be a relation defined on Σ^* as follows: $\forall xy$ (xRy iff $y = x$b). Let R' be the reflexive, transitive closure of R. Let $L' = \{x : \exists y \in L\ (yR'x)\}$. Write a regular expression for L'.

$R' = \{(\varepsilon, \varepsilon), (\varepsilon, b), (\varepsilon, bb), (\varepsilon, bbb), \ldots (a, a), (a, ab), (a, abb), \ldots (b, b), (b, bb), (b, bbb), \ldots\}$.

So a regular expression for L' is: $(\varepsilon \cup a \cup b)b^*$.

19) In Appendix O, we summarize the main features of the regular expression language in Perl. What feature of that regular expression language makes it possible to write regular expressions that describe languages that aren't regular?

The ability to store strings of arbitrary length in variables and then require that those variables match later in the string.

20) For each of the following statements, state whether it is *True* or *False*. Prove your answer.

a) $(ab)^*a = a(ba)^*$.

True.

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

True.

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

False.

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

True.

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

True.

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

True.

g) If α and β are any two regular expressions, then $(\alpha \cup \beta)^* = \alpha(\beta\alpha \cup \alpha)$.

False.

h) If α and β are any two regular expressions, then $(\alpha\beta)^*\alpha = \alpha(\beta\alpha)^*$.

True.

7 Regular Gramamars

1) Show a regular grammar for each of the following languages:

a) $\{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ contains an odd number of a's and an odd number of b's$\}$.

$G = (\{EE, EO, OE, OO, \mathtt{a}, \mathtt{b}\}, \{\mathtt{a}, \mathtt{b}\}, EE, R)$, where $R =$

$EE \rightarrow \mathtt{a}\ OE$
$EE \rightarrow \mathtt{b}\ EO$
$OE \rightarrow \mathtt{b}\ OO$
$OE \rightarrow \mathtt{a}\ EE$
$EO \rightarrow \varepsilon$
$EO \rightarrow \mathtt{a}\ OO$
$EO \rightarrow \mathtt{b}\ EE$
$OO \rightarrow \mathtt{a}\ EO$
$OO \rightarrow \mathtt{b}\ OE$

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

$S \rightarrow \mathtt{a}A \mid \mathtt{b}B \mid \varepsilon$
$A \rightarrow \mathtt{a}C \mid \mathtt{b}B \mid \varepsilon$
$B \rightarrow \mathtt{a}A \mid \mathtt{b}B \mid \varepsilon$
$C \rightarrow \mathtt{a}C \mid \mathtt{b}B$

c) $\{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ contains the substring abb$\}$.

d) $\{w \in \{\mathtt{a}, \mathtt{b}\}^* :$ if w contains the substring aa then $|w|$ is odd$\}$.

It helps to begin by rewriting this as:

$\{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ does not contain the substring aa or $|w|$ is odd$\}$

The easiest way to design this grammar is to build an FSM first. The FSM has seven states:

- S, the start state, is never reentered after the first character is read.
- T_1: No aa yet; even length; last character was a.
- T_2: No aa yet; odd length; last character was a.
- T_3: No aa yet; even length; last character was b.
- T_4: No aa yet; odd length; last character was b.
- T_5: aa seen; even length.
- T_6: aa seen; odd length.

Now we build a regular grammar whose nonterminals correspond to those states. So we have:

$S \rightarrow \mathtt{a}T_2 \mid \mathtt{b}T_4$
$T_1 \rightarrow \mathtt{a}T_6 \mid \mathtt{b}T_4 \mid \varepsilon$
$T_2 \rightarrow \mathtt{a}T_5 \mid \mathtt{b}T_3 \mid \varepsilon$
$T_3 \rightarrow \mathtt{a}T_2 \mid \mathtt{b}T_4 \mid \varepsilon$
$T_4 \rightarrow \mathtt{a}T_1 \mid \mathtt{b}T_3 \mid \varepsilon$
$T_5 \rightarrow \mathtt{a}T_6 \mid \mathtt{b}T_6$
$T_6 \rightarrow \mathtt{a}T_5 \mid \mathtt{b}T_5 \mid \varepsilon$

e) $\{w \in \{\text{a, b}\}^* : w$ does not contain the substring aabb$\}$.

$S \rightarrow \text{a}A$	$A \rightarrow \text{a}B$	$B \rightarrow \text{a}B$	$C \rightarrow \text{a}A$
$S \rightarrow \text{b}S$	$A \rightarrow \text{b}S$	$B \rightarrow \text{b}C$	$C \rightarrow \varepsilon$
$S \rightarrow \varepsilon$	$A \rightarrow \varepsilon$	$B \rightarrow \varepsilon$	

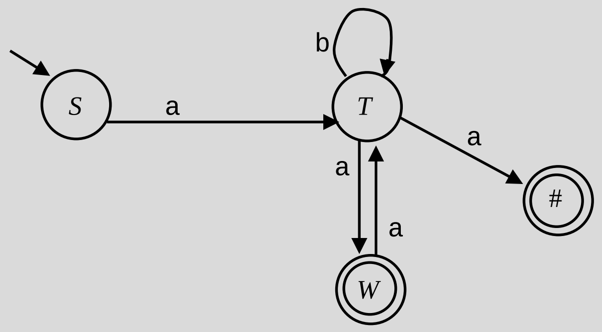
2) Consider the following regular grammar G:

$S \rightarrow \text{a}T$
$T \rightarrow \text{b}T$
$T \rightarrow \text{a}$
$T \rightarrow \text{a}W$
$W \rightarrow \varepsilon$
$W \rightarrow \text{a}T$

a) Write a regular expression that generates $L(G)$.

a (b ∪ aa) a

b) Use *grammartofsm* to generate an FSM M that accepts $L(G)$.



3) Consider again the FSM M shown in Exercise 5.1. Show a regular grammar that generates $L(M)$.

4) Show by construction that, for every FSM M there exists a regular grammar G such that $L(G) = L(M)$.

1. Make M deterministic (to get rid of ε-transitions).
2. Create a nonterminal for each state in the new M.
3. The start state becomes the starting nonterminal
4. For each transition $\delta(T, a) = U$, make a rule of the form $T \rightarrow aU$.
5. For each accepting state T, make a rule of the form $T \rightarrow \varepsilon$.

5) Let $L = \{w \in \{\text{a, b}\}^* :$ every a in w is immediately followed by at least one b$\}$.

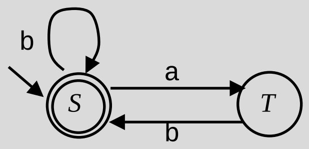
a) Write a regular expression that describes L.

(ab ∪ b)*

b) Write a regular grammar that generates L.

$S \rightarrow \text{b}S$
$S \rightarrow \text{a}T$
$S \rightarrow \varepsilon$
$T \rightarrow \text{b}S$

c) Construct an FSM that accepts *L*.



8 Regular and Nonregular Languages

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

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

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

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

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

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

   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) $\{w \in \{\mathtt{0}, \mathtt{1}, \#\}^* : w = x\#y, \text{ where } x, y \in \{\mathtt{0}, \mathtt{1}\}^* \text{ and } |x|\cdot|y| \equiv_5 0\}$. (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:
   $((\mathtt{0} \cup \mathtt{1})^5)^*\#(\mathtt{0} \cup \mathtt{1})^* \;\cup\; (\mathtt{0} \cup \mathtt{1})^*\#((\mathtt{0} \cup \mathtt{1})^5)^*$, where $(\mathtt{0} \cup \mathtt{1})^5$ is simply a shorthand for writing $(\mathtt{0} \cup \mathtt{1})$ five times in a row.

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

   Regular. Finite.

   f) $\{w \in \{\mathtt{Y}, \mathtt{N}\}^* : w$ contains at least two Y's and at most two N'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) $\{w = xy : x, y \in \{\mathtt{a}, \mathtt{b}\}^* \text{ and } |x| = |y| \text{ and } \#_{\mathtt{a}}(x) \geq \#_{\mathtt{a}}(y)\}$.

   Not regular, which we'll show by pumping. Let $w = \mathtt{a}^k\mathtt{bba}^k$. y must occur in the first a region and be equal to $\mathtt{a}^p$ 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, $\mathtt{a}^{k-p}\mathtt{bba}^k$, is not in L.

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

   Regular. Note that $L = (\mathtt{a} \cup \mathtt{b})^*$. Why? Take any string s in $(\mathtt{a} \cup \mathtt{b})^*$. 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) $\{w = xyzy : x, y, z \in \{\mathtt{0}, \mathtt{1}\}^+\}$.

Regular. Any string w in $\{\mathtt{0}, \mathtt{1}\}^+$ 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:

$$L = ((\mathtt{0} \cup \mathtt{1})^+ \mathtt{0} (\mathtt{0} \cup \mathtt{1})^+ \mathtt{0}) \cup ((\mathtt{0} \cup \mathtt{1})^+ \mathtt{1} (\mathtt{0} \cup \mathtt{1})^+ \mathtt{1}).$$

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

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.

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

It is easy to show, using pumping, that L' is not regular: Let $w = \mathtt{0}^k\mathtt{1}^k$. y must occur in the initial string of 0's, since $|xy| \leq k$. So $y = \mathtt{0}^i$ 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, $L'' = L' \cap \mathtt{0}^*\mathtt{1}^*$ would also have to be regular. But L'' is $\mathtt{0}^n\mathtt{1}^n$, which we already know is not regular.

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

Not regular, which we show by pumping. Let $w = \mathtt{a}^k\mathtt{b}^k\mathtt{b}^k\mathtt{a}^k$. So y must be $\mathtt{a}^p$ 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) $\{w \in \{\mathtt{a}, \mathtt{b}\}^* : \exists x \in \{\mathtt{a}, \mathtt{b}\}^+ (w = x\, x^R\, x)\}$.

Not regular, which we show by pumping: Let $w = \mathtt{a}^k\mathtt{bba}^k\mathtt{a}^k\mathtt{b}$. y must occur in the initial string of a's, since $|xy| \leq k$. So $y = \mathtt{a}^i$ 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 $\mathtt{a}^{k+i}\mathtt{bba}^k\mathtt{a}^k\mathtt{b}$. 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) $\{w \in \{\mathtt{a}, \mathtt{b}\}^*$: 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. $L = \mathtt{a}^* \cup \mathtt{a}^+\mathtt{b}^+\mathtt{a}^+(\mathtt{b}^+\mathtt{a}^+)^* \cup \mathtt{b}^* \cup \mathtt{b}^+\mathtt{a}^+\mathtt{b}^+(\mathtt{a}^+\mathtt{b}^+)^*$.

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

Not regular, which we'll show by pumping. Let $w = \mathtt{a}^k\mathtt{b}^{k+2}$. y must equal $\mathtt{a}^p$ 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) $\{w \in \{\mathtt{a}, \mathtt{b}\}^* : w = xyz, |x| = |y| = |z|$, 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 $w = \mathtt{a}^k\mathtt{a}^k\mathtt{b}^k$. This string is in L since $x = \mathtt{a}^k$, $y = \mathtt{a}^k$, and $z = \mathtt{b}^k$. y (from the pumping theorem) $= \mathtt{a}^p$ 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) $\{w$: $w \in \{\mathtt{a} – \mathtt{z}\}^*$ 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) $\{w$: $w \in \{\mathtt{a} - \mathtt{z}\}^*$ 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 $\{\mathtt{0} - \mathtt{9}\}^*$ 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 $L_0 = \{\mathtt{b}\mathtt{a}^i\mathtt{b}^j\mathtt{a}^k, j \geq 0, 0 \leq i \leq k\}$.

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

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

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

Not regular, which we can prove by pumping. Let $w = \mathtt{1}^t$, where t is the smallest integer that is a power of ten and is greater than k. y must be $\mathtt{1}^p$ 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) $\{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* :$ in each prefix x of w, $\#_{\mathtt{a}}(x) = \#_{\mathtt{b}}(x) = \#_{\mathtt{c}}(x))\}$.

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

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

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

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

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

3) Define the following two languages:

$L_{\mathtt{a}} = \{w \in \{\mathtt{a}, \mathtt{b}\}^* :$ in each prefix x of w, $\#_{\mathtt{a}}(x) \geq \#_{\mathtt{b}}(x)\}$.

$L_{\mathtt{b}} = \{w \in \{\mathtt{a}, \mathtt{b}\}^* :$ in each prefix x of w, $\#_{\mathtt{b}}(x) \geq \#_{\mathtt{a}}(x)\}$.

a) Let $L_1 = L_{\mathtt{a}} \cap L_{\mathtt{b}}$. Is L_1 regular? Prove your answer.

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

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

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

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

4) For each of the following languages L, state whether L is regular or not and prove your answer:

a) $\{uww^Rv : u, v, w \in \{\mathtt{a}, \mathtt{b}\}^+\}$.

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 $L = (\mathtt{a} \cup \mathtt{b})^+ (\mathtt{aa} \cup \mathtt{bb}) (\mathtt{a} \cup \mathtt{b})^+$.

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.

5) Use the Pumping Theorem to complete the proof, given in L.3.1, that English isn't regular.

6) Prove *by construction* that the regular languages are closed under:
a) intersection.
b) set difference.

7) Prove that the regular languages are closed under each of the following operations:
a) $pref(L) = \{w: \exists x \in \Sigma^* (wx \in L)\}$.

By construction. Let $M = (K, \Sigma, \delta, s, A)$ be any FSM that accepts L:
Construct $M' = (K', \Sigma', \delta', s', A')$ to accept $pref(L)$ from M:

```
1) Initially, let $M'$ be $M$.
2) Determine the set $X$ of states in $M'$ from which there exists at least one path to some accepting state:
    a) Let $n$ be the number of states in $M'$.
    b) Initialize $X$ to {}.
    c) For each state $q$ in $M'$ do:
        For each string $w$ in $\Sigma^*$ where $0 \le w \le n-1$ do:
            Starting in state $q$, read the characters of $w$ one at a time and make the appropriate
                transition in $M'$.
            If this process ever lands in an accepting state of $M'$, add $q$ to $X$ and quit processing $w$.
3) $A_{M'} = X$.
```

Comments on this algorithm: The only trick here is to find all the states from which there exists some continuation string that could eventually lead to an accepting state. To accept $pref(L)$, that continuation does not have to be present. But we need to know when there could exist one. This can be done by trying continuation strings. But there is an infinite number of them. We can't try them all. True, but we don't have to. If there is a path to an accepting state, then there must be a path that does not require going through a loop. The maximum length of such a path is n-1. So we simply try all strings of length up to n-1.

b) $suff(L) = \{w: \exists x \in \Sigma^* (xw \in L)\}$.

By construction. Let $M = (K, \Sigma, \delta, s, A)$ be any FSM that accepts L. Construct $M' = (K', \Sigma', \delta', s', A')$ to accept $suf(L)$ from M:

```
1) Initially, let $M'$ be $M$.
2) Determine the set $X$ of states that are reachable from $s$:
    a) Let $n$ be the number of states in $M'$.
    b) Initialize $X$ to $\{s\}$. .
    c) For $i = 1$ to $n - 1$ do:
        i. For every state $q$ in $X$ do:
            a. For every character $c$ in $\Sigma \cup \{\varepsilon\}$ do:
                i. For every transition $(q, c, q')$ in $M'$ do:
                    a. $X = X \cup \{q'\}$
3) Add to $M'$ a new start state $s'$. Create an $\varepsilon$-transition from $s'$ to $s$.
4) Add to $M'$ an $\varepsilon$-transition from $s'$ to every element of $X$.
```

Comments on this algorithm: The basic idea is that we want to accept the end of any string in L. Another way of thinking of this is that M'' must act as though it had seen the beginning of the string without having actually seen it. So we want to skip from the start state to anyplace that the beginning of any string could have taken us. But we can't necessarily skip to all states of M', since there may be some that aren't reachable from s by any input string. So we must first find the reachable strings.

To find the reachable strings, we must follow all possible paths from s. We need only try strings of length up to n-1 where n is the number of states of M, since any longer strings must simply follow loops and thus cannot get anywhere that some shorter string could not have reached.

We must also be careful that we only skip the beginning of a string. We do not want to be able to skip pieces out of the middle. If the original start state had any transitions into it, we might do that if we allow ε-jumps from it. So we create a new start state s' that functions solely as the start state. We then create the ε-transitions from it to all the states that were marked as reachable.

c) $reverse(L) = \{x \in \Sigma^* : x = w^R \text{ for some } w \in L\}$.

By construction. Let $M = (K, \Sigma, \delta, s, A)$ be any FSM that accepts L. M must be written out completely, without an implied dead state. Then construct $M' = (K', \Sigma', \delta', s', A')$ to accept $reverse(L)$ from M:

1) Initially, let M' be M.
2) Reverse the direction of every transition in M'.
3) Construct a new state q. Make it the start state of M'. Create an ε-transition from q to every state that was an accepting state in M.
4) M' has a single accepting state, the start state of M.

d) letter substitution (as defined in Section 8.3).

Let *sub* be any function from Σ_1 to Σ_2*. Let *letsub* be a letter substitution function from L_1 to L_2. So *letsub*(L_1) = $\{w \in \Sigma_2^* : \exists y \in L_1$ and $w = y$ except that every character c of y has been replaced by $sub(c)\}$. There are two ways to prove by construction that the regular languages are closed under letter substitution:

First, we can do it with FSMs: if L is regular, then it is accepted by some DFSM $M = (K, \Sigma, \delta, s, A)$. From M we construct a machine $M' = (K', \Sigma', \delta', s', A')$ to accept *letsub*(L). Initially, let $M' = M$. Now change M' as follows. For each arc in M' labelled x, for any x, change the label to $sub(x)$. M' will accept exactly the strings accepted by M except that, if M accepted some character x, M' will accept $s(x)$.

Or we can do it with regular expressions: If L is a regular language, then it is generated by some regular expression α. We generate a new regular expression α' that generates *letsub*(L). $\alpha' = \alpha$, except that, for every character c in Σ_1, we replace every occurrence of c in α by $sub(c)$.

8) Using the defintions of *maxstring* and *mix* given in Section 8.6, give a precise definition of each of the following languages:

a) *maxstring*(A^nB^n).

maxstring(A^nB^n) = $\{a^nb^n : n>0\}$. (Note: $\varepsilon \notin$ *maxstring*(A^nB^n) because each element of A^nB^n can be concatenated to ε to generate a string in A^nB^n. But, given any other string in A^nB^n (e.g., aabb), there is nothing except ε that can be added to make a string in A^nB^n.)

b) *maxstring*($a^ib^jc^k$, $1 \le k \le j \le i$).

maxstring($a^ib^jc^k$, $1 \le k \le j \le i$) = $\{a^ib^jc^j, 1 \le j \le i\}$.

c) *maxstring*(L_1L_2), where $L_1 = \{w \in \{a, b\}^* : w$ contains exactly one a$\}$ and $L_2 = \{a\}$.

L_1L_2 = b*ab*a. So *maxstring*(L_1L_2) = b*ab*a.

d) *mix*((aba)*).

mix((aba)*) = (abaaba)*.

e) *mix*(a*b*).

This one is tricky. To come up with the answer, consider the following elements of a*b* and ask what elements they generate in *mix*(a*b*):

- aaa: nothing, since only even length strings can contribute.
- aaaa: aaaa. Every even length string of all a's just contributes itself.
- bbb: nothing, since only even length strings can contribute.
- bbbbbb: bbbbbb. Every even length string of all b's just contributes itself.
- aaabbb: aaabbb. If the original string has even length and the number a's is the same as the number of b's, the string contributes itself.
- aab: nothing, since only even length strings can contribute.
- aabbbb: aabbbb. If the original string has even length and the number a's is less than the number of b's, the string contributes itself since the second half is all b's and is thus unchanged by being reversed.
- aaaabb: aaabba. If the original string has even length and the number a's is greater than the number of b's, then the second half starts with a's. Thus reversing the second half creates a substring that starts with b's and ends with a's.

This analysis tells us that $mix(\mathtt{a}^*\mathtt{b}^*) = (\mathtt{aa})^* \cup (\mathtt{bb})^* \cup \{\mathtt{a}^i\mathtt{b}^j, i \leq j \text{ and } i + j \text{ is even}\} \cup \{w : |w| = n, n \text{ is even}, w = \mathtt{a}^i\mathtt{b}^j\mathtt{a}^k, i = n/2\}$. But this can be simplified, since the case of all a's is a special case of more a's than b's and the case of all b's is a special case of more b's than a's. So we have:

$mix(\mathtt{a}^*\mathtt{b}^*) = \{\mathtt{a}^i\mathtt{b}^j, i \leq j \text{ and } i + j \text{ is even}\} \cup \{w : |w| = n, n \text{ is even}, w = \mathtt{a}^i\mathtt{b}^j\mathtt{a}^k, i = n/2\}$.

9) Prove that the regular languages are not closed under *mix*.

Let $L = (\mathtt{ab})^*$. Then $mix((\mathtt{ab})^*) = L' = \{(\mathtt{ab})^{2n+1}, n \geq 0\} \cup \{(\mathtt{ab})^n(\mathtt{ba})^n, n \geq 0\}$. L' is not regular. If it were, then $L'' = L' \cap (\mathtt{ab})^+(\mathtt{ba})^+ = \{(\mathtt{ab})^n(\mathtt{ba})^n, n \geq 1\}$ would also be regular. But it isn't, which we can show by pumping. Let $w = (\mathtt{ab})^N(\mathtt{ba})^N$. y must occur in the $(\mathtt{ab})^N$ region. We consider each possibility:

- $|y|$ is odd. Let $q = 2$. The resulting string has odd length. But all strings in L have even length, so it is not in L.
- $|y|$ is even and $y = (\mathtt{ab})^p$ for some p. Let $q = 2$. The resulting string has more (ab) pairs than (ba) pairs, and so is not in L.
- $|y|$ is even and $y = (\mathtt{ba})^p$ for some p. Let $q = 2$. The resulting string has more (ba) pairs than (ab) pairs, and so is not in L.

10) Recall that $maxstring(L) = \{w: w \in L \text{ and } \forall z \in \Sigma^* (z \neq \varepsilon \rightarrow wz \notin L)\}$.

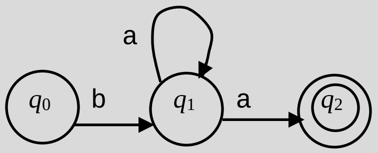
a) Prove that the regular languages are closed under *maxstring*.

The proof is by construction. If L is regular, then it is accepted by some DFSA $M = (K, \Sigma, \Delta, s, A)$. We construct a new DFSM $M^* = (K^*, \Sigma^*, \Delta^*, s^*, A^*)$, such that $L(M^*) = maxstring(L)$. The idea is that M^* will operate exactly as M would have except that A^* will include only states that are accepting states in M and from which there exists no path of at least one character to any accepting state (back to itself or to any other). So an algorithm to construct M^* is:

```
1.   Initially, let $M^* = M$.
/* Check each accepting state in $M$ to see whether there are paths from it to some accepting state.
2.   For each state $q$ in $A$ do:
     2.1.  Follow all paths out of $q$ for $|K|$ steps or until the path reaches an element of $A$ or some
           state it has already visited.
     2.2.  If the path reached an element of $A$, then $q$ is not an element of $A^*$.
     2.3.  If the path ended without reaching an element of $A$, then $q$ is an element of $A^*$.
```

Comments on this algorithm:

1. Why do we need to start with a deterministic machine? Suppose L is ba*a. $maxstring(L) = \{\}$. But suppose that M were:



If we executed our algorithm with this machine, we would accept ba*a rather than {}.

2. Your initial thought may be that the accepting states of M^* should be simply the accepting states of M that have no transitions out of them. But there could be transitions that themselves lead no where, in which case they don't affect the computation of *maxstring*. So we must start by finding exactly those accepting states of M such that there is no continuation (other than ε) that leads again to an accepting state.

b) If *maxstring*(L) is regular, must L also be regular? Prove your answer.

No. Consider Prime$_a$ = {a^n : n is prime}. Prime$_a$ is not regular. But *maxstring*(Prime$_a$) = ∅, which is regular.

11) Let *midchar* be a function on languages. Define:

midchar(L) = {c : $\exists w \in L$ ($w = ycz$, $c \in \Sigma$, $y \in \Sigma^*$, $z \in \Sigma^*$, $|y| = |z|$}

a) Are the regular languages closed under *midchar*? Prove your answer.

Yes. For any language L, *midchar*(L) must be finite and thus regular.

b) Are the nonregular languages closed under *midchar*? Prove your answer.

No. A^nB^n is not regular, but *midchar*(A^nB^n) = ∅, which is regular.

12) Define the function *twice*(L) = {w : $\exists x \in L$ (x can be written as $c_1c_2 \ldots c_n$, for some $n \geq 1$, where each $c_i \in \Sigma_L$, and $w = c_1c_1c_2c_2 \ldots c_nc_n$)}.

a) Let L = (1 ∪ 0)*1. Write a regular expression for *twice*(L).

(11 ∪ 00)*11.

b) Are the regular languages closed under *twice*? Prove your answer.

Yes, by construction. If L is regular, then there is some DFSM M that accepts it. We build an FSM M' that accepts *twice*(L):

1. Initially, let M' be M.
2. Modify M' as follows: For every transition in M from some state p to some state q with label c, do:
   2.1. Remove the transition from M'.
   2.2. Create a new state p'.
   2.3. Add to M' two transitions: ((p, c), p') and (p', c), q).
3. Make the start state of M' be the same as the start state of M.
4. Make every accepting state in M also be an accepting state in M'.

13) Define the function *shuffle*(L) = {w : $\exists x \in L$ (w is some permutation of x)}. For example, if L = {ab, abc}, then *shuffle*(L) = {ab, abc, ba, acb, bac, bca, cab, cba}. Are the regular languages closed under *shuffle*? Prove your answer.

No. Let L = (ab)*. Then *shuffle*(L) = {$w \in$ {a, b}* : $\#_a(w) = \#_b(w)$}, which is not regular.

14) Define the function *copyandreverse*(L) = {w : $\exists x \in L$ ($w = xx^R$)}. Are the regular languages closed under *copyandreverse*? Prove your answer.

No. Let L = (a ∪ b)*. Then *copyandreverse*(L) = WW^R, which is not regular.

15) Let L_1 and L_2 be regular languages. Let L be the language consisting of strings that are contained in exactly one of L_1 and L_2. Prove that L is regular.

We prove this by construction of an FSM that accepts L. Let M_1 be a DFSM that accepts L_1 and let M_2 be a DFSM that accepts L_2. The idea is that we'll build a machine M' that simulates the parallel execution of M_1 and M_2. The states of M' will correspond to ordered pairs of the states of M_1 and M_2. The accepting states of M' will be the ones that correspond to ordered pairs of states that are accepting in their original machines.

16) Define two integers i and j to be ***twin primes*** 💻 iff both i and j are prime and $|j - i| = 2$.

a) Let $L = \{w \in \{\texttt{1}\}^* : w$ is the unary notation for a natural number n such that there exists a pair p and q of twin primes, both $> n.\}$ Is L regular?

Regular. We don't know how to build an FSM for L. We can, however, prove that it is regular by considering the following two possibilities:

(1) There is an infinite number of twin primes. In this case, for every n, there exists a pair of twin primes greater than n. Thus $L = \texttt{1}^*$, which is clearly regular.

(2) There is not an infinite number of twin primes. In this case, there is some largest pair. There is thus also a largest n that has a pair greater than it. Thus the set of such n's is finite and so is L (since it contains exactly the unary encodings of those values of n). Since L is finite, it is regular.

It is unknown which of these is true.

b) Let $L = \{x, y : x$ is the decimal encoding of a positive integer i, y is the decimal encoding of a positive integer j, and i and j are twin primes$\}$. Is L regular?

It is unknown whether the number of pairs of twin primes is finite. If it is, then L is regular. If it is not, then L is probably not regular.

17) Consider any function $f(L_1) = L_2$, where L_1 and L_2 are both languages over the alphabet $\Sigma = \{0,1\}$. We say that function f is ***nice*** iff L_2 is regular iff L_1 is regular. For each of the following functions, state whether or not it is nice and prove your answer:

a) $f(L) = L^R$.

Nice. The regular languages are closed under reverse.

b) $f(L) = \{w$: w is formed by taking a string in L and replacing all `1`'s with `0`'s and leaving the `0`'s unchanged$\}$.

Not nice. Let L be $\{0^n1^n: n \geq 0\}$, which is not regular. Then $f(L) = (\texttt{00})^*$, which is regular.

c) $f(L) = L \cup 0^*$

Not nice. Let L be $\{0^p: p$ is prime$\}$, which is not regular. Then $f(L) = \texttt{0}^*$, which is regular.

d) $f(L) = \{w$: w is formed by taking a string in L and replacing all `1`'s with `0`'s and all `0`'s with `1`'s (simultaneously)$\}$.

Nice. The regular languages are closed under letter substitution.

e) $f(L) = \{w: \exists x \in L\ (w = x\texttt{00})\}$.

Nice. If L is regular, then $f(L)$ is the concatenation of two regular languages and so is regular. If $f(L)$ is regular, then there is some FSM M that accepts it. From M, we construct a new FSM M' that accepts L', defined to be the set of strings with the property that if `00` is concatenated onto them they are in $f(L)$. To build M', we begin by letting it be M. Then we start at each accepting state of M and trace backwards along arcs labeled `0`. The set of states that can be reached by following exactly two steps in that way become the accepting states of M'. Since there is an FSM that accepts L', it is regular. Finally, we note that $L' = L$.

f) $f(L) = \{w$: w is formed by taking a string in L and removing the last character$\}$.

Not nice. Let L be $\{\texttt{a}^p\texttt{b}: p$ is prime$\} \cup \{\texttt{a}^p\texttt{c}: p$ is a positive integer and is not prime$\}$, which is not regular. Then $f(L) = \texttt{a}^+$, which is regular.

18) We'll say that a language L over an alphabet Σ is ***splitable*** iff the following property holds: Let w be any string in L that can be written as $c_1c_2 \ldots c_{2n}$, for some $n \geq 1$, where each $c_i \in \Sigma$. Then $x = c_1c_3 \ldots c_{2n-1}$ is also in L.

   a) Give an example of a splitable regular language.

   Two simple examples are $(\mathtt{a} \cup \mathtt{b})^*$ and $\mathtt{a}^*$.

   b) Is every regular language splitable?

   No. $L = (\mathtt{ab})^*$ is regular. Let $w = \mathtt{abab}$. Then $w \in L$. The split version of w is $\mathtt{aa}$, which is not in L. So L isn't splitable. An even simpler example is $L = \{\mathtt{ab}\}$.

   c) Does there exist a nonregular language that is splitable?

   Yes. One example is $L = \mathrm{A^nB^nC^n} = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n : n \geq 0\}$. Only even length strings in L have an impact on whether or not L is splitable. So consider any string w in $\{\mathtt{a}^{2n}\mathtt{b}^{2n}\mathtt{c}^{2n} : n \geq 0\}$. The split version of any such string is $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$, which is also in L.

   Another example is $L = \{\mathtt{a}^n : n > 2 \text{ and } n \text{ is prime}\}$. Since L contains no even length strings, it is trivially splitable.

19) Define the class IR to be the class of languages that are both infinite and regular. Is the class IR closed under:

   a) union.

   Yes. Let $L_3 = L_1 \cup L_2$. If L_1 and L_2 are in IR, then L_3 must be. First we observe that L_3 must be infinite because every element of L_1 is in L_3. Next we observe that L_3 must be regular because the regular languages are closed under union.

   b) intersection.

   No. Let $L_1 = \mathtt{a}^*$. Let $L_2 = \mathtt{b}^*$. Both L_1 and L_2 are infinite and regular. But $L_3 = L_1 \cap L_2 = \{\varepsilon\}$, which is finite.

   c) Kleene star.

   Yes. Let $L_2 = L_1^*$. If L_1 is in IR, then L_2 must be. First we observe that L_2 must be infinite because every element of L_1 is in L_2. Next we observe that L_2 must be regular because the regular languages are closed under Kleene star.

20) Consider the language $L = \{x\mathtt{0}^ny\mathtt{1}^nz : n \geq 0, x \in P, y \in Q, z \in R\}$, where P, Q, and R are nonempty sets over the alphabet $\{\mathtt{0}, \mathtt{1}\}$. Can you find regular languages P, Q, and R such that L is not regular? Can you find regular languages P, Q, and R such that L is regular?

   Let P, Q, $R = \{\varepsilon\}$. Then $L = \mathtt{0}^n\mathtt{1}^n$, and so is not regular. On the other hand, let $P = \mathtt{0}^*$, $Q = \{\varepsilon\}$ and let $R = \mathtt{1}^*$. Now $L = \mathtt{0}^*\mathtt{1}^*$, which is regular.

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

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

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

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

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

c) 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 $L = \cup$ (a*, a*, a*, …), which is a*, which is regular.

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

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

e) 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.

f) 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 = {a $\cup$ b)*. Let L = {$a^n b^n$: $n \geq 0$}, which is not regular.

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

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

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

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

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

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

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

k) The intersection of an infinite number of regular languages must be regular.

False. Let x_1, x_2, x_3, …be the sequence 0, 1, 4, 6, 8, 9, … of nonprime, nonnegative integers. Let a^{xi} be a string of x_i a's. Let L_i be the language a* - {a^{xi}}.

Now consider L = the infinite intersection of the sequence of languages L_1, L_2, … Note that L = {a^p, where p is prime}. We have proved that L is not regular.

l) 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, $j \geq 0$ 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.

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

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

n) Every nonregular language can be described as the intersection of an infinite number of regular languages.

True. Note first that every nonregular language is countably infinite. (Because every finite language is regular and no language contains more strings than Σ*.)

Let L be a non-regular language, and consider the following infinite set of languages:

S = {L_i | L_i is the language Σ* - w_i}

where w_i is the i-th string in Σ* in lexicographical order. For example, suppose Σ= {a,b} and the words are ordered as follows: ε, a, b, aa, ab, ba, bb, aaa, ...Then, L_1=Σ*-{ε}, L_2 = Σ*-{a}, L_3=Σ*-{b}, L_4=Σ*-{aa}, etc. Obviously, each L_i is a regular language (since it's the complement of the regular language that contains only one string, namely the i-th string m, which is regular).

Now, consider the following set of languages: T = {L_i | the i-th string in Σ* is not in L }, i.e., T is the set of all L_i's where the i-th string (the one that defines L_i) is not in L. For example, suppose L = {$\mathtt{a}^n\mathtt{b}^n$: $n \geq 0$}. Since Σ = {a,b}, the L_i's are as mentioned above. T = {Σ*- a, Σ*- b, Σ*- aa, Σ*- ba, Σ*- bb, Σ*- aaa, …}.

Return now to our original, nonregular language L. We observe that it is simply the intersection of all of the languages in T. What we are effectively doing is subtracting out, one at a time, all the strings that should be excluded from L.

Note that T is infinite (i.e., it contains an infinite number of languages). Why? Recall that L = Σ* - one string for each element of T. If T were finite, that would mean that L were equal to Σ* - L', where L' contains only some finite number of strings. But then L' would be regular, since it would be finite. If L' were regular, then its complement, Σ* - L', would also be regular. But L is precisely the complement of L' and, by hypothesis, L is not regular.

This result may seem a bit counterintuitive. After all, we have proven that the regular languages are closed under intersection. But what we proved is that the regular languages are closed under *finite* intersection. It is true that if you intersect any finite number of regular languages, the result must be regular. We can prove that by induction on the number of times we apply the intersection operator, starting with the base case, that we proved by construction, in which we intersect any two regular languages and must get a result that is regular. But the induction proof technique only applies to finite values of n. To see this, consider a more straightforward case. We can prove, by induction, that the sum of the integers from 1 to N is $N*(N+1)/2$. So, for any value of N, we can determine the sum. But what is the sum of all the integers? In other words, what is the answer if we sum an infinite number of integers. The answer is that it is not an integer. There exists no value N such that the answer is $N*(N+1)/2$.

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

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

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

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

q) The nonregular languages are closed under intersection.

False.
Let $L_1 = \{a^n b^n$, where n is even$\}$. L_1 is not regular.
Let $L_2 = \{a^n b^n$, 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 $L = a^*$, which is regular.
Let $L' = a^p$, 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.