Please note that the problems are in a no particular order of difficulty. If you find one difficult then just try another one that you think may be easier.

*The harder questions have an asterisk * – these may be skipped at first then attempted later.*

The Pumping Lemma says that *any "sufficiently long" string in a regular language L can be broken into three parts such that if we "pump" the middle part (repeat it zero or more times) then the result would still belong to L.*

Pumping Lemma: Let L be a regular language. Then there exists a constant p such that for every string w in L, with $|w| \geq p$, we can break w into three strings $w = xyz$ such that

1. $y \neq \varepsilon$ (i.e $|y| > 0$ or $|y| \neq 0$)
2. $|xy| \leq p$ (xy cannot be more than the first p symbols of w)
3. For all $k \geq 0$, the string $xy^k z$ is also in L (i.e. $xy^* z \in L$)

The Pumping Lemma when used to prove that a language L is **not regular** can be viewed as a "game" between a **Prover** and a **Falsifier** as follows:

❶ **Prover** claims L is regular and fixes the value of the pumping length p.

❷ **Falsifier** challenges **Prover** and picks a string $w \in L$ of length at least p symbols.
Often, we pick w to be "at the edge" of membership, i.e. as close as possible to failing to be a member.

❸ **Prover** writes $w = xyz$ such that $|xy| \leq p$ and $y \neq \varepsilon$.

❹ **Falsifier** wins by finding a value for k such that $xy^k z$ is **not** in L. If it cannot then it fails and **Prover** wins.

The language L is not regular if **Falsifier** can always win this game systematically.

Questions (1)–(6) are almost complete proofs using the Pumping Lemma (PL). Complete them by filling in the hidden details.

(1) Show that the language $L = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$ is not regular.

❶ **Prover** claims L is regular and fixes the value of the pumping length p.

❷ **Falsifier** challenges **Prover** and picks $w = \mathtt{a}^p\boxed{\mathtt{b}^p} \in L$ ($|w| = \boxed{2p} \geq p$).

❸ **Prover** tries to decompose w into three parts $w = \boxed{xyz}$ but sees that the condition $|xy| \leq \boxed{p}$ forces x and y to only contain the symbol $\boxed{\mathtt{a}}$. Furthermore, y cannot just be the empty string because of the condition $\boxed{y \neq \varepsilon}$.
Seeing this, the only option available is to have $xy = \mathtt{a}^m$ for some $m \geq 1$, and then we get $z = \mathtt{a}^{p-m}\mathtt{b}^p$.

❹ **Falsifier** now sees that $xy^0z, xy^2z, xy^3z, \ldots$ all do not belong to L because they either have less or more $\boxed{\mathtt{a}}$'s than there are $\boxed{\mathtt{b}}$'s.
So, any such string will be enough for **Falsifier** to win the game.

(2) $L = \{ww \mid w \in \{0,1\}^*\}$.

❶ **Prover** claims L is regular and fixes the value of the pumping length p.

❷ **Falsifier** challenges **Prover** and chooses $w = (0^p1)(0^p1) \in L$.
This has length

$$|w| = (p+1) + (\boxed{p+1}) = \boxed{2p+2} \geq p.$$

❸ **Prover** The PL now guarantees that w can be split into three substrings $w = xyz$ satisfying $|xy| \leq p$ and $y \neq \varepsilon$.

❹ **Falsifier** Since

$$w = (0^p1)(\boxed{0^p1}) = xyz$$

with $|xy| \leq p$ then we must have that y only contains the symbol $\boxed{0}$.
We can then pump y and produce $xy^2z = xyyz \notin L$ because the first half $\boxed{\text{no longer matches}}$ the second half.
So L is not regular.

(3) $L = \{a^ib^j \mid i > j\}$

❶ **Prover** claims L is regular and fixes the value of the pumping length p.

❷ **Falsifier** challenges **Prover** and chooses

$$w = a^{\boxed{p}+1}b^{\boxed{p}}$$

Here $|w| = \boxed{(p+1)+p} = 2p+1 \boxed{\geq} p$

❸ **Prover** writes

$$w = (xy)z = (a^{\boxed{p}})ab^{\boxed{p}}$$

i.e. xy is a string of $\boxed{a}$'s only

❹ **Falsifier** forms

$$xy^{\boxed{0}}z = xz = a^{p+1-\boxed{|y|}}b^p \notin L$$

because $|y| \geq 1$. (so $p+1-\boxed{|y|} \leq p$).

(4) $L = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid 0 \le i < j < k\}$

❶ **Prover** claims L is regular and fixes the value of the pumping length p.

❷ **Falsifier** challenges **Prover** and chooses

$$w = \mathtt{a}^{\boxed{p}}\mathtt{b}^{\boxed{p}+1}\mathtt{c}^{\boxed{p}+2}.$$

Here $|w| = p + \boxed{(p+1)+(p+2)}\ \boxed{\ge}\ p$

❸ **Prover** writes

$$w = (xy)z = (\mathtt{a}^p)\mathtt{b}^{p+1}\mathtt{c}^{p+2}$$

where xy is a string of $\boxed{\mathtt{a}}$'s only

❹ **Falsifier** forms

$$xy^2z = \mathtt{a}^{p+\boxed{|y|}}\mathtt{b}^{p+1}\mathtt{c}^{p+2} \notin L$$

because $|y| \ge 1$.

(5) $L = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i > j > k \ge 0\}$

❶ **Prover** claims L is regular and fixes the value of the pumping length p.

❷ **Falsifier** challenges **Prover** and chooses

$$w = \mathtt{a}^{\boxed{p+2}}\mathtt{b}^{p+1}\mathtt{c}^{0}.$$

Here $|w| = \boxed{p+2} + (p+1) + 0\ \boxed{\ge}\ \boxed{p}$.

❸ **Prover** writes

$$w = (xy)z = (\mathtt{a}^{\boxed{p+2}})\mathtt{b}^{p+1}\mathtt{c}^{0}$$

where xy is a string of $\boxed{\mathtt{a}}$'s only

❹ **Falsifier** forms

$$xy^0z = xz = \mathtt{a}^{\boxed{p+2}-|y|}\mathtt{b}^{p+1}\mathtt{c}^{0} \notin L$$

because $|y| \ge 1$.

(6)* $L = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid j \neq i \text{ or } j \neq k\}$

❶ **Prover** claims L is regular and fixes the value of the pumping length p.

❷ **Falsifier** challenges **Prover** and chooses

$$w = \mathtt{a}^p\mathtt{b}^{\boxed{p!+p}}\mathtt{c}^{\boxed{p!+p}}$$

Here $|w| = p + 2(\boxed{p!+p}) \geq p$.

❸ **Prover** writes

$$w = (xy)z = (\mathtt{a}^p)\mathtt{b}^{\boxed{p!+p}}\mathtt{c}^{\boxed{p!+p}}$$

where xy is a string of a's only

❹ **Falsifier** forms

$$xy^kz = a^{p+(k-1)|y|}\mathtt{b}^{\boxed{p!+p}}\mathtt{c}^{\boxed{p!+p}}$$

where $k = 1 + \boxed{p!}/\boxed{|y|}$. This gives $a^{p!+p}\mathtt{b}^{\boxed{p!+p}}\mathtt{c}^{\boxed{p!+p}}$ which is not in the language.

(7) Let $\Sigma = \{\mathtt{0}, \mathtt{1}, \mathtt{+}, \mathtt{=}\}$, and ADD be the language given by

$\{u\mathtt{=}v\mathtt{+}w \mid u, v, w$ are binary integers, and u is the sum of v and w in the usual sense$\}$

Show that ADD is not regular.

Solution

❶ **Prover** claims L is regular and fixes the value of the pumping length p.

❷ **Falsifier** challenges **Prover** and chooses w to be $\mathtt{1}^p = \mathtt{0}^p + \mathtt{1}^p$

❸ **Prover** can only have 1's in y, so $y = \mathtt{1}^d$ for some $d \geq 1$

❹ **Falsifier** constructs $xyyz$ and finds it to be

$$\mathtt{1}^{p+d} = \mathtt{0}^p + \mathtt{1}^p$$

which is not correct.

(8) Let $L = \{1^{2^n} \mid n \geq 0\}$. Show that L cannot be regular.

Solution

❶ **Prover** claims L is regular and fixes the value of the pumping length p.

❷ **Falsifier** challenges **Prover** and $w = 1^{p^2}$

❸ **Prover** writes

$$x = 1^a, y = 1^b, z = 1^{2^p-a-b}$$

where $1 \leq b \leq p$.

❹ **Falsifier**

$$xyyz = 1^{2^p+b}$$

The next string after 1^{p^2} in terms of length is $1^{2^{p+1}} = 1^{2^p+2^p}$ but

$$2^p < 2^p + b < 2^p + 2^p.$$

because

$$1 \leq b \leq p < 2^p$$

So $xyyz \notin L$.

(9) Go through the JFLAP tutorial on: http://www.jflap.org/tutorial/pumpinglemma/regular/ and then try all the "games." You may find it useful to work on this in pairs (one as **Prover** and another as **Falsifier**).

Note that *some of the languages below are actually regular* – in this case, you will need to devise a strategy for **Prover** to always win no matter what **Falsifier** chooses as a challenge string.

The notation $n_{\mathtt{a}}(w)$ means: the number of occurrence of the symbol a in the string w. e.g. $n_{\mathtt{a}}(\mathtt{aba}) = 2$ and $n_{\mathtt{b}}(\mathtt{aba}) = 1$.

Hint — The list of languages is as follows: (Assume $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ unless otherwise specified)

1. $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$ — Hint: $\mathtt{a}^p\mathtt{b}^p$
2. $\{w \in \Sigma^* \mid n_{\mathtt{a}}(w) < n_{\mathtt{b}}(w)\}$ — Hint: $\mathtt{a}^p\mathtt{b}^{p+1}$
   i.e. language of strings which have less a's than there are b's.
3. $\{ww^R \mid w \in \Sigma^*\}$ — Hint: $\mathtt{a}^p\mathtt{b}^{2p}\mathtt{a}^p$
   w^R denotes the reverse string of w, e.g. $\mathtt{abb}^R = \mathtt{bba}$.
4. $\{(\mathtt{ab})^n\mathtt{a}^m \mid n > m \geq 0\}$ — Hint: $(\mathtt{ab})^{p+1}\mathtt{a}^p$
5. $\{\mathtt{a}^n\mathtt{b}^m\mathtt{c}^{n+m} \mid n \geq 0, m \geq 0\}$
6. $\{\mathtt{a}^n\mathtt{b}^\ell\mathtt{a}^k \mid n > 5, \ell > 3, \ell \geq k\}$
7. $\{\mathtt{a}^n \mid n \text{ is even}\}$
8. $\{\mathtt{a}^n\mathtt{b}^m \mid n \text{ is odd } \textbf{or } m \text{ is even}\}$
9. $\{\mathtt{bba}(\mathtt{ba})^n\mathtt{a}^{n-1} \mid n \geq 1\}$

10. $\{\mathtt{b}^5 w \mid w \in \Sigma^* \text{ and } 2n_{\mathtt{a}}(w) = 3n_{\mathtt{b}}(w)\}$
11. $\{\mathtt{b}^5 w \mid w \in \Sigma^* \text{ and } n_{\mathtt{a}}(w) + n_{\mathtt{b}}(w) \equiv 0 \pmod 3\}$
12. $\{\mathtt{b}^m(\mathtt{ab})^n(\mathtt{ba})^n \mid m \geq 4, n \geq 1\}$
13. $\{(\mathtt{ab})^{2n} \mid n \geq 1\}$

(10) **(Minimum pumping length)** *The purpose of the following problem is for you to pay close attention to the exact formulation of the Pumping Lemma (PL).*

The PL says that every RL has a pumping length p, such that every string in the language can be pumped if it has length p or more.

Note that if p is a pumping length for a language L then so is any other length $\geq p$. We define the *minimum pumping length* for L to be the smallest such p.

For example, if $L = \mathtt{ab}^*$ then the minimum pumping length is 2. This is because the string $w = \mathtt{a}$ is in L and has length 1, yet w cannot be pumped; but any string in L of length 2 or more contains a b and hence can be pumped by dividing it so that $x = \mathtt{a}$, $y = \mathtt{b}$ and z is the rest of the string.

For each of the following languages, give the minimum pumping length and justify your answer.

a) $\mathtt{aab}^*$
b) $\mathtt{a}^*\mathtt{b}^*$
c) $\mathtt{aab} + \mathtt{a}^*\mathtt{b}^*$
d) $\mathtt{a}^*\mathtt{b}^+\mathtt{a}^+\mathtt{b}^* + \mathtt{ba}^*$
The notation $\mathtt{a}^+$ is equivalent to $\mathtt{aa}^*$, i.e. 1 or more a's (as opposed to $\mathtt{a}^*$ which means zero or more a's).
e) $(\mathtt{01})^*$
f) ε
g) $\mathtt{b}^*\mathtt{ab}^*\mathtt{ab}^*$
h) $\mathtt{10}(\mathtt{11}^*\mathtt{0})^*\mathtt{0}$
i) 1011
j) Σ^*

Solution

a) $\mathtt{aab}^* = \{\mathtt{aa}, \mathtt{aab}, \mathtt{aab}^2, \mathtt{aab}^3, \mathtt{aab}^4, \mathtt{aab}^5, \mathtt{aab}^6, \ldots\}$, sorted in ascending order with respect to string length.

We notice that aa cannot be pumped (e.g. if we repeat a once then we get aaa which is not in the language), but starting from aab we can pump b to get $\mathtt{aab}^k$ for $k = 0, 1, 2, \ldots$ which are all in the language, so the pumping length for this language is $\boxed{p = 3}$ (the length of aab, the shortest string that can be pumped).

b) $\mathtt{a}^*\mathtt{b}^* = \{\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{a}^2, \mathtt{b}^2, \mathtt{ab}, \mathtt{a}^3, \mathtt{b}^3, \mathtt{aab}, \mathtt{abb}, \mathtt{a}^4, \mathtt{b}^4, \ldots\}$

ε is not pump-able, but a or b are, so $\boxed{p = 1}$.

c) $\mathtt{aab} + \mathtt{a}^*\mathtt{b}^* = \{\mathtt{aab}\} \cup \{\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{a}^2, \mathtt{b}^2, \mathtt{ab}, \mathtt{a}^3, \mathtt{b}^3, \mathtt{aab}, \mathtt{abb}, \mathtt{a}^4, \mathtt{b}^4, \ldots\}$ which is just $\{\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{a}^2, \mathtt{b}^2, \mathtt{ab}, \mathtt{a}^3, \mathtt{b}^3, \mathtt{aab}, \mathtt{abb}, \mathtt{a}^4, \mathtt{b}^4, \ldots\} = \mathtt{a}^*\mathtt{b}^*$ again, so $\boxed{p = 1}$.

d) $\mathtt{a^*b^+a^+b^*} + \mathtt{ba^*} = \{\mathtt{ba}, \mathtt{aba}, \mathtt{bab}, \mathtt{bba}, \mathtt{aab}, \} \cup \{\mathtt{b}, \mathtt{ba}, \mathtt{ba}^2, \mathtt{ba}^3, \dots\}$

This is a union of two languages:

- The RegEx $\mathtt{a^*b^+a^+b^*}$ gives $p = 2$ as we can loop a or b from its shortest string $\mathtt{a^0b^1a^1a^0} = \mathtt{ba}$.
- The RegEx $\mathtt{ba^*}$ also gives $p = 2$ as we can loop a from its second shortest string ba.

So, we conclude that the given language has $\boxed{p = 2}$ (the shortest of the two lengths, which happen to be the same in this example).

e) $(\mathtt{01})^* = \{\varepsilon, \mathtt{01}, \mathtt{0101}, \mathtt{010101}, (\mathtt{01})^4, (\mathtt{01})^5, \dots\}$

So starting from 01 we can set $x = z = \varepsilon$ and $y = \mathtt{01}$ in the Pumping Lemma. Hence, $\boxed{p = 2}$, the length of 01.

f) ε

This RegEx represents the language that only contains the empty string: $\{\varepsilon\}$. There is no way of writing $\varepsilon = xyz$ with $y \neq \varepsilon$, so it suffices to let $\boxed{p = 1}$. The language is finite (and therefore regular), and there are no pump-able strings!

Note: ε is the only possible string of length zero over any alphabet, and it is not pump-able in any language, so p is always ≥ 1, unless the language is the empty language $\emptyset$.

g) $\mathtt{b^*ab^*ab^*} = \{\mathtt{aa}, \mathtt{baa}, \mathtt{aba}, \mathtt{aab}, \mathtt{b^2aa}, \mathtt{ab^2a}, \mathtt{aab^2}, \mathtt{baba}, \mathtt{baab}, \mathtt{abab}, \dots\}$

Here, $\mathtt{b^0ab^0ab^0} = \mathtt{aa}$ is not pump-able (if pumped then it would produce $\mathtt{aa^+}$ which is not of the required form $\mathtt{b^*ab^*ab^*}$).

However, all the strings of length 3 are pump-able producing e.g. $\mathtt{b^*aa}$ from baa. So $\boxed{p = 3}$.

h) $\mathtt{10(11^*0)^*0} = \{\mathtt{100}, \mathtt{10100}, \mathtt{101100}, \mathtt{1011100}, \mathtt{1010100}, \dots\}$

$\mathtt{100} = \mathtt{10(11^*0)^00}$ is not pump-able, but $\mathtt{10100} = \mathtt{10(11^00)^10}$ is pump-able producing $\mathtt{10100} = \mathtt{10(10)^*0}$, so $\boxed{p = 5}$.

i) 1011

This RegEx represents the language that only contains one string: $\{\mathtt{1011}\}$. If we pump any symbol then the length of the resulting string will be at least 5, so it cannot be a member of this language. Hence, it suffices to let $\boxed{p = 5}$. The language is finite (and therefore regular), and there are no pump-able strings!

j) $\Sigma^* = \{\varepsilon, \dots\}$ is the language of all possible strings over the alphabet Σ.

In particular, if a is a symbol then $\mathtt{a}^*$ is also in Σ^*, so $\boxed{p = 1}$.

(11) **(Pumping lemma applied to RLs)** When we try to apply the Pumping Lemma to a Regular Language the **Prover** wins, and the **Falsifier** loses.

Show why **Falsifier** loses when L is one of the following RLs:

a) $\{\mathtt{00}, \mathtt{11}\}$

b) $(\mathtt{aa} + \mathtt{bb})^*$

c) $\mathtt{01^*0^*1}$

d) $\emptyset$

Solution

a) $\{\texttt{00}, \texttt{11}\}$

This is a finite language, so **Prover** chooses $p = 3$. **Falsifier** cannot choose a string that is long enough. ($|w| \geq 3$ but the two available strings are only 2 symbols long.)

b) $(\texttt{aa} + \texttt{bb})^*$

Prover chooses $p = 2$ and $y = \texttt{aa}$ or $\texttt{bb}$, depending on the string chosen by the **Falsifier** .

c) $\texttt{01}^*\texttt{0}^*\texttt{1}$

Prover chooses $p = 3$ and $y = \texttt{0}$ or $\texttt{1}$, depending on the string chosen by the **Falsifier** .

d) $\emptyset$

Falsifier has no strings to choose from! (**Prover** may set $p = 0$ or any other value.)