Problem 1(a)

Find the Regex for the complement language of L(aa^*bb^*)

The essense of the regex is 1 or more a followed by 1 or more b, so the complement is a string that has zero a and/or zero b, or any instance of ba, becuase it violates the order.

So the regex is: $a^* + b^* + (a+b)^* \, ba \, (a+b)^*$

Problem 4

Show that if a language family is closed under union and complementation, it must also be closed under intersection.

If $L = L_1 \cap L_2$, then $L = \overline{\overline{L_1 \cap L_2}}$ which is $\overline{\overline{L_1} \cup \overline{L_2}}$, by DeMorgan's law. This is expressed using only union and complementation, which regular languages are closed over, so regular languages must also be closed under union because it can be expressed via other closed operations

Problem 10

Show that regular languages are closed under symmetric difference

The symmetric difference is defined by $(L_1 - (L_1 \cap L_2)) \cup (L_2 - (L_1 \cap L_2))$, creating a set of elements that are in only one of the two input sets.

Regular languages are closed under difference, as if $L = L_1 - L_2$, then $L = L_1 \cap \overline{L_2}$

We just showed that regular languages are closed under union, using the fact that they are closed under intersect and complement, so because we can represent the symmetric difference exclusively using operators unver which regular languages are closed, symmetric difference is also closed

Problem 3

Find an algorithm to determine whether a given regular language L is a palindrome language.

The general idea is that the NFA that accepts a string in L should accept its reverse, too. We know that reglangs are effectively closed under reversal, so this is possible.

For an NFA M that defines L, construct another NFA M^R as follows:

1. Create a copy of M and reverse all edges, call this M^R
2. Add a new state I and a new state F to M^R

3. Add epsilon transitions from I to all accepting states in M^R
4. Add epsilon transitions from the initial state of M^R to F
5. Designate F as the only accepting state, and I as the only initial state

This generates an NFA M^R which accepts L^R. A language is palindromic if $L(M) = L(M \cap M^R)$. This is decidable, being an instance of the equivalence problem, so you can apply the algorithm to solve the equivalence problem and decide whether or not L is palindromic using this equivalence

Problem 6

Show that there exists an algorithm to determine whether $\mathbf{L_1}$ is a proper subset of $\mathbf{L_2}$ for any regular languages $\mathbf{L_1}$ and $\mathbf{L_2}$.

L_1 is a proper subset of L_2 iff $L_1 \subseteq L_2$ and $L_1 \neq L_2$. $L_1 \subseteq L_2$ is decidable, as it is equivalent to $L_1 \cap \overline{L_2} = \emptyset$, which uses exclusively operations that reglangs are effectively closed under. $L_1 \neq L_2$ is also decidable, as equivalence is decidable and you can take the opposite of whatever the decision is to find non-equivalence.

Thus, there exists an algorithm to determine proper subset-ness, as there exists an algorithm to determine whether L_1 is a subset of L_2 and an algorithm to determine if they are equal.

Problem 15

Find an algorithm to determine whether a regular language L contains a finite number of even-length strings.

Let $\Sigma = \{\sigma_1, \sigma_2, \dots\}$ be the alphabet over which L is constructed.

Let $M = (Q, \Sigma, \delta, q_o, F)$ be an NFA where $Q = \{\text{init}, \text{loop}\}$, $q_o = \text{init}$, $F = \text{loop}$, and δ is defined as follows:

$\delta(\text{init}, x) = \text{loop}$ for all $x \in \Sigma$, and $\delta^*(\text{loop}, uv) = \text{loop}$ for all u, $v \in \Sigma$.

Let $L_{\text{odd}} = L(M)$, a language which consists of all odd-length strings over Σ.

A language L consists of a finite number of even-length strings if $L - L_{\text{odd}}$ is finite, which is a decidable problem as mentioned above (accomplished via a depth-first search to identify cycles)