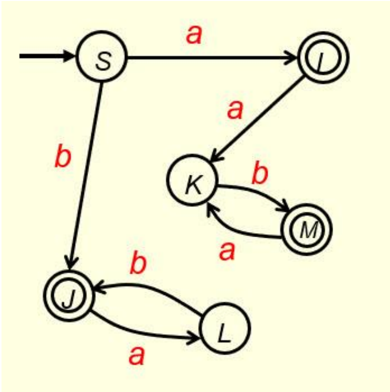
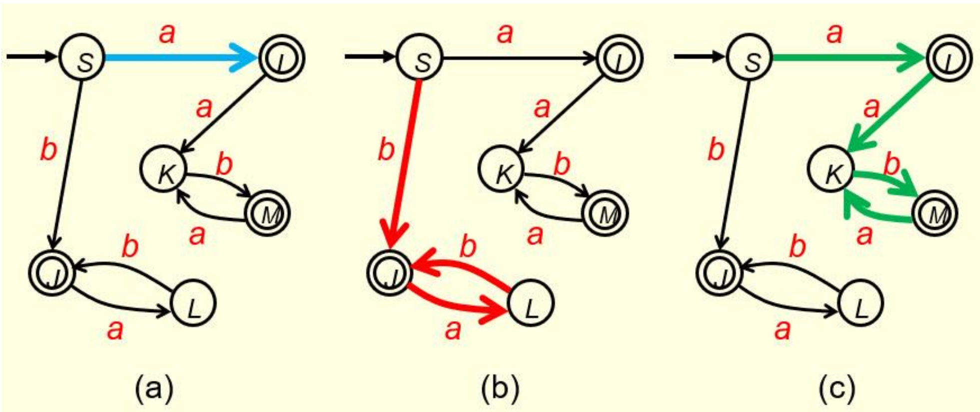
CS375 Homework Assignment 4-2 Solution Set (19 points)

Due date: 02/20/2025

1. In homework assignment 4-1, the finite automaton (FA) considered in question #5 is the one shown below.



To find the regular expression of this FA, one can use a direct approach or an elimination process. For a direct approach, all the possible acceptance paths are shown below. Edges in the acceptable paths in cases (a), (b) and (c) are shown in blue, red and green, respectively.



In case (a), we get this expression: a (0.5 points)

In case (b), we get this expression: b(ab)* = (ba)*b (0.5 points)

In case ©, we get this expression: aab(ab)* = $a(ab)^+$ (0.5 points)

The expressions for case (a) and case (c) can be combined and simplified to get this expression: (1 point)

a(ab)*

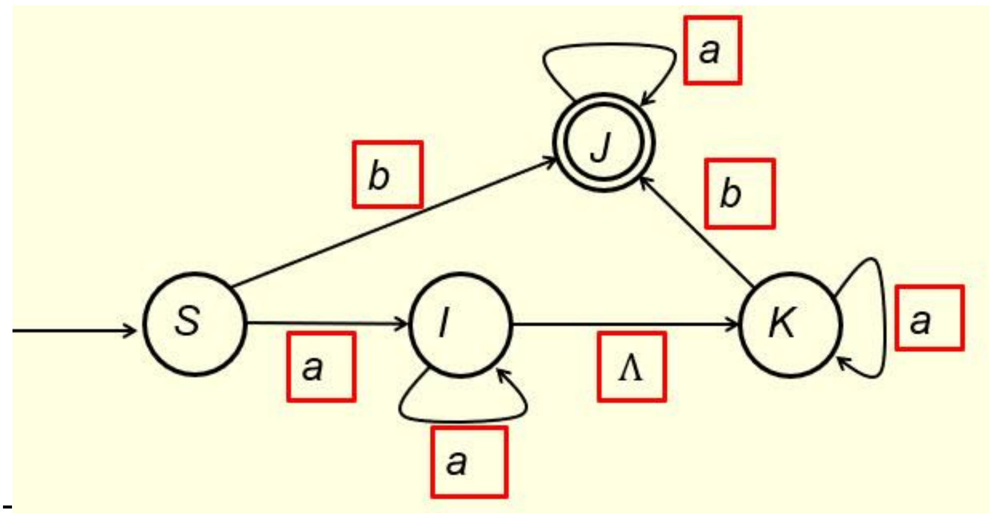
Combining the above expression with the expression for case (b), we get the following expression. This expression is the same as the expression given in Question 3 of HW4-1.

a(ab)* + (ba)*b

2. Given the following regular grammar:

S → aI | bJ,   I → K | aI,   J → aJ | Λ,   K → aK | bJ

we can find regular expression of the language of this regular grammar by first using the algorithm given in slide 39 of the notes "Regular Languages and Finite Automata- IV" to convert the grammar to an NFA that recognizes the language of this grammar, then find regular expression of this NFA using the approach shown in Question 1. First fill out blanks in the following figure so that the resulting NFA would recognize the language of the given grammar.   (2 points)



-

Then use the approach shown in Question 1 to find the regular expression of this NFA and put it in the following blank.   (2 points)

Regular expression = *ba* + aa*a*ba* = a*ba**

3. Fill out the following blanks to make it a regular grammar for the given regular expression with S being the start symbol: a*b*c* + d   (5 points)

S → d | A | B | C | Λ

A → aA | B | C | Λ

B → bB | C | Λ

C → cC | Λ

or

S → d | aA | bB | cC | Λ

A → aA | bB | cC | Λ

B → bB | cC | Λ

C → cC | Λ

4. Using **Pumping Lemma** (slides 42-48 of the notes 'Regular Languages & Finite Automata-IV') one can show the language $L = \{a^n b^n \mid n \in \mathbf{N}\}$ is not regular (We need this property in the notes 'Context-free Languages and Pushdown Automata I). This is done by way of contradiction. We assume L is regular. Since L is infinite, Pumping Lemma applies. We then consider the string $s = a^m b^m$ where m is the number of states in the DFA that recognizes L. Since the length of s is bigger than m, by Pumping Lemma, there exists strings x, y *and* z such that $s = xyz$, $y \neq \Lambda$, $|xy| \leq 2m$ and $xy^kz \in L$ for all $k \in N$. If $|xy| < 2m$ then the first repeated state on the acceptance path cannot be a final state. Why? (4 points)

This statement is not true. The condition that $|xy| < 2m$ implies the substring z is not an empty string. Hence, when the execution of xy is finished, the DFA will continue executing the remaining portion of the input string, z, until the last symbol of z is executed, it doesn't matter if the last symbol of y ends in a final state.

5. Fill out the following blanks to make it a context-free grammar for the given language over the alphabet {a, b}: $\{ a^{2n}b^{3n+2} \mid n >= 0\}$ (2 points)

aaSbbb | bb

S → |

6. Fill out the following blanks to make it a context-free grammar for the given language:

{ $a^{2n} b^{n+2}$ | n >= 0 } { $a^{n+2} b^{2n}$ | n >= 0 } (2 points)

S → AB, A → aaAb | bb , B → aBbb | aa