CS 181: Formal Languages and Automata Theory	**Spring 2021**
Homework 2 Solution	
Assigned: Tue 4/6	*Due: Mon 4/12, 6:00pm PDT*

Problem 1

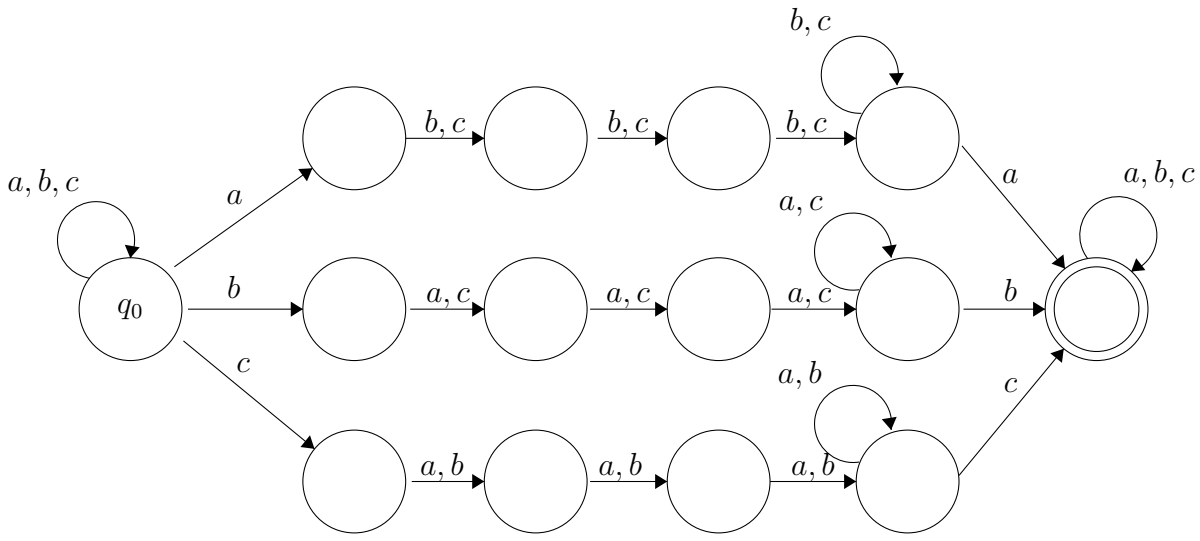
Let $\Sigma = \{a, b, c\}$. Show an NFA which recognizes the following language over Σ:

$$L_1 = \{w \in \Sigma^+ \mid w \text{ contains at least one substring consisting of two of the same symbol separated by at least three occurrences of the other two symbols}\}$$

The substring between the "two of the same symbol" can consist of one or both of the "other two symbols" but cannot contain the "same symbol".

E.g., L_1 contains: $abbcbba, caaaaaacb, bbbabcbcbaaaaa, babcbcbcaaaaaca, abbbcbababac$. L_1 does not contain: $abbaba, caacaab, bbbbcbcaaaaa$.

Specify the NFA as a state diagram. It does not have to be fully specified, but you have to use the blocking convention correctly. *Be sure to clearly indicate your initial state and accepting state(s).* You may find it helpful to use the shorthand discussed in class of putting more than one symbol on an edge of the transition diagram. Part of your score will be based on demonstrating effective use of nondeterminism in the NFA model.
Briefly describe how your design works.
Solution:



We use the "guess and check" interpretation of non-determinism, where each branch from the initial state checks for a different letter to be used as the "same symbol" in the definition of L_1. If we find one of the correct substring patterns, stay at the final accepting state.

Problem 2

Let $\Sigma = \{a, b\}$. Use the Pumping Lemma for the FSLs to show that the following language over Sigma is not a FSL:

$$L_2 = \{a^{(2n)}b^n \mid n \geq 0\}$$

Postponed.

Problem 3

Show that $L_3 = \{w \in \Sigma^* \mid \#(a, w) = 2\#(b, w)\}$ is not FSL using the technique discussed in class of proof by contradiction using the closure properties of the family of FSLs. You may find some of the other languages mentioned in this homework set useful for solving this problem.

Solution:
Assume L_3 is a FSL, and we know that a^*b^* is a finite state language. Then by the closure property of finite state languages we know that $L_2 = L_3 \cap (a^*b^*)$ is also a finite state language, since finite state languages are closed under intersection. However, from problem 2 we know that L_2 is not a FSL; therefore a contradiction is reached. As a result, our assumption that L_3 is FSL is not correct. Thus, we conclude that L_3 is not FSL.

Problem 4

Let $\Sigma = \{0, 1, \#\}$. Give a regular expression for the following language:

$$L_4 = \{w \in \Sigma^* \mid \text{in } w, \text{ to the right of every 0 there is at least one 1 before any \#'s}\}$$

E.g., L_4 contains: $1011\#1, 000, 01$. L_4 does not contain: $\#00\#11, 10101010\#$.
Briefly explain how your regular expression is designed to correctly represent this language.

Solution:

$$(1 \cup \#)^* \left(0\,(1\,(1 \cup \#)^*)^*\right)^*$$

A Brief explanation:

- $(1 \cup \#)^*$ is used to generate a prefix that does not contain any 0's. This expression can also generate a whole string that does not contain any 0's when the rest part $\left(0\,(1\,(1 \cup \#)^*)^*\right)^*$ is ϵ.
- $\left(0\,(1\,(1 \cup \#)^*)^*\right)^*$ will guarantee that to the right of every 0 there is at least one 1 before any #'s because if a # was generated after a 0, then there must be at least one 1 generated before the # thanks to the expression $1\,(1 \cup \#)^*$.

Problem 5

Recall from lecture that a directed rooted tree is a connected, directed graph which is acyclic, even when viewed as an undirected graph, where one node is designated as the root and all edges in the tree are directed away from the root towards the leaf nodes.

The height of a directed rooted tree is defined as the length of a longest path from the root to a leaf. The height can be expressed in terms of the number of edges on a longest path or in terms of the number of nodes on a longest path. I.e., "h edges" and "$h+1$ nodes" are two different ways of expressing the same tree height.
We define a k-ary directed rooted tree to be a directed rooted tree in which each node has at most k children.

Corrected problem statement:
Let $h > 1$, and let T be a k-ary directed rooted tree with $(k^h) + 1$ leaf nodes. Show by induction on h that for any degree $k > 1$: T must have height at least $h+1$ edges ($h+2$ nodes.)

Solution:

We will actually prove a slightly stronger statement than the problem:
Let $h > 1$, and let T be a k-ary directed rooted tree with *at least* $(k^h)+1$ leaf nodes. Show by induction on h that for any degree $k > 1$: T must have height at least $h+1$ edges ($h+2$ nodes.)

Base case $h = 2$:
Observe when $h = 2$, the largest number of leaf nodes we can have is with the root having k children, and each child also having k leaf children. This gives us k^2 total possible leaves while maintaining height at most 2. Since every node has the maximum allowed number of children, if T has at least $k^2 + 1$ total leaf nodes, then it must have height at least $3 = h + 1$.
Inductive case h:
Suppose the statement is true for all such trees with $(k^{h-1}) + 1$ leaf nodes.
T has at least $(k^h) + 1$ leaf nodes. Consider the root node of T: it has at most k children, and at least $(k^h) + 1$ total leaves. This means that there is at least 1 child which has at least $(k^h - 1) + 1$ total leaves. This is because even with k children and dividing leaves evenly, we still wind up with a child with exactly $(k^h - 1) + 1$ leaves, and dividing the nodes unevenly can only increase the maximum number of leaves.
Call the subtree rooted at this child T'. Then since T' has at least $(k^h - 1) + 1$ leaves, by the inductive hypothesis T' must have height at least h. The root of T' is a child of the root of T, so T must have height at least $h + 1$.
This concludes the proof.