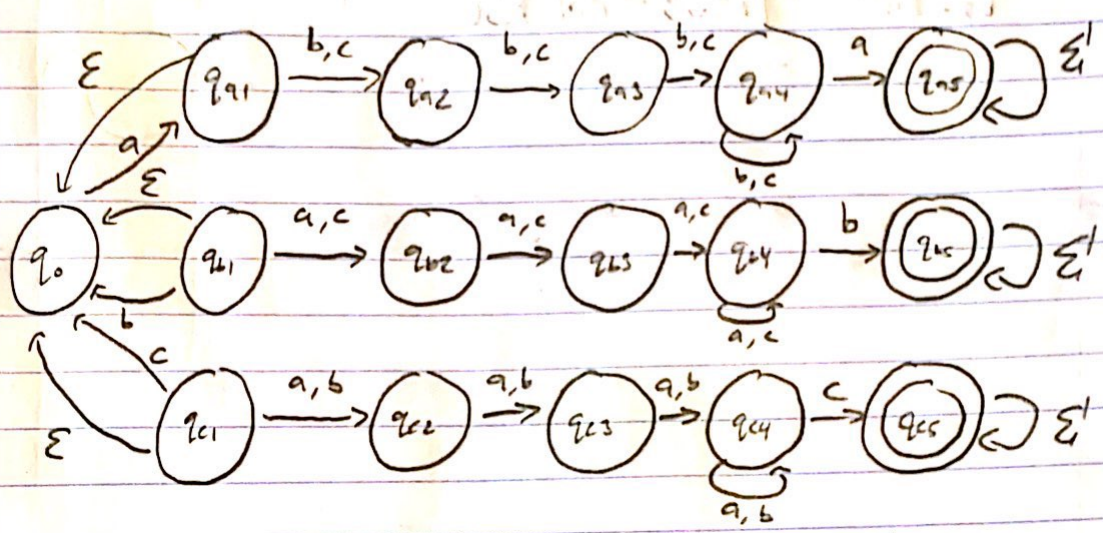
Charles Zhang
305413659

CS181 HW#2

1) $\Sigma' = \{a, b, c\}$

$L_1 = \{w \in \Sigma'^* \mid w$ contains at least one substring of two of the same symbol separated by at least three occurrances of the other two symbols$\}$



This design splits the language into 3 parts: substrings that begin with a, those that begin with b, and those that begin with c. It uses 3 intermediary states for each case to fulfill the condition that the start/end symbol are separated by 3+ occurrences of the other 2 symbols. At the 3rd state (q_{a4}, q_{b4}, q_{c4}), the design loops into the same state if it receives anything other than the symbol the substring starts with (accounting for 3+) and enters an accepting state if the target symbol is read. The accepting states loop into themselves, since, once a valid substring is found, the string is in the language. This makes use of nondeterminism by using the blocking convention and ε-arrows to simplify the design.

2) Cancelled

3) $L_3 = \{ w \in \Sigma^* \mid \#(a,w) = 2\#(b,w) \}$

$\Sigma = \{a, b\}$

$L_2 = \{a^{(2n)} b^n \mid n \geq 0\}$ → not FSL

Union, ~~Concatenation~~, Kleene

Complementation, Intersection ✓

~~Reverse~~

Proof:

- Assume L_3 is regular
- By closure properties of FSL, L_3 must be closed under intersection
- By the notation of regular expressions, $L_3 \cap a^*b^* = \{a^{(2n)} b^n \mid n \geq 0\}$
- In other words $L_3 \cap a^*b^* = L_2$
- a^*b^* is regular by closure under Kleene star and concatenation
- Therefore, if L_3 is regular, $L_3 \cap a^*b^*$ is regular by closure under intersection
- However, $L_3 \cap a^*b^* = L_2$, and L_2 is known to be nonregular, leading to a contradiction →←

4) $\Sigma' = \{0, 1, \#\}$

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

~~$(1 \cup \#)^* (0 (0 \cup 1)^* (1 \# (1 \cup \#)^*)^*)^*$~~

$\boxed{(1 \cup \#)^* (0 (0^* (1 \# (1 \cup \#)^* \cup 1)^*)^*)^*}$

This reg. exp. first checks for any prefixes of w that don't contain a 0, which should all be accepted. If it finds a 0, it then accepts any 0s that follow it. Afterwards, it looks for a 1# or a 1. If a # is found instead, the string will not be accepted. If a 1# is found, it will then accept everything past it until it hits another 0, where it once again must check the language's condition. If a 1 is found, it will look for a 1# or 1 again.

5) If T has $k^h + 1$ leaf nodes, then T has height of $h+1$

Let T be a k-ary directed rooted tree. Show by induction on h that for any degree $k > 1$

Basis: $h = 2$

T has $k^2 + 1$ leaf nodes at a height of 3.

Since each node has at most k children, the most leaf nodes possible at height 2 is $k \cdot k$, or k^2.

Since the number of leaf nodes is $k^2 + 1$, the minimum height possible is 3.

Basis solved ✓

Induction hypothesis: assume that this properly holds for trees up to height h, where $h \geq 1$

Inductive step: Prove that this properly holds for trees with height h+1

- Let T' be a tree of height h+1
- This tells us that T's subtrees must have a height $\leq h$
- By the inductive hypothesis, we know these subtrees contain at most k^h leaf nodes each
- The total number of leaf nodes in T' is equivalent to the number of leaves in its subtrees
- T' has at most k subtrees by the definition of a k-ary tree
- T' has at most $k(k^h)$ or k^{h+1} leaf nodes
- In order for it to have $k^{h+1}+1$ leaf nodes, it must have height h+2
- Inductive case proved ✓