CS 334 ~ Problem Set 4

Harris Spahic (I pledge my honor that I have abided by the Stevens Honor System)
10/01/21

1) Suppose $A=\{w : w = w^R, w \in \{0,1\}^*\}$ is regular.
Then, it has a pumping length p for A.

→ Since w is a palindrome we can seperate it into three possible substrings w_1, w_2, w_3. Let
$|w| = 2k+1$ for some $k \in \mathbb{Z}^+$

→ w_1 = the first k characters of w
w_2 = the $k+1^{th}$ character
w_3 = the next k characters of $w = w_1^R$

Since A is regular, $w = w_1 w_2 w_3 = w_1 w_2 w_1^R = xyz$ s.t $|y|>0$ & $|xy| \leq p$.
Let $p = k$.

→ xy must $\subseteq w_1$ since $|w_1| = p_{=k}$, but $\forall i \geq 0\ xy^i z \in A$ must also be true.

Say $w_1 = 0^p$, $w_2 = 1$, $w_3 = 0^p$ → $w = w_1 w_2 w_3 = 0^p 1 0^p$

→ y must be string consisting of some number of consecutive 0's less than or equal to p. Call that # $c \in \mathbb{Z}^+$.

→ If $i=0$, $0^{p-c} 1 0^p$ which doesn't belong to A.

This violates the third property of the pumping lemma, giving us a contradiction, for any p. ∎

1b) Yes, notice $\{w : w = xyx^R, x,y \in \{0,1\}^*\}$, w is made of 3 substrings $x \circ y \circ x^R$. Both x & y can be represented as regular expressions $(\Sigma)^* = x$, $(\Sigma)^* = y$. We also proved in problem set 3 that if x is a regular expression, then x^R is also a regular expression. Since the concatonation of regular expressions is regular, xyx^R & the langage it expresses is regular.

#2) Reflexive:

Let $u \in \Sigma^*$,

$\rightarrow \forall x \in \Sigma^*, ux \in L \leftrightarrow ux \in L$

Thus compatibility is reflexive.

Symmetric:

Let $u \in \Sigma^*$, $v \in \Sigma^*$

Given $\forall x \in \Sigma^*, ux \in L \leftrightarrow vx \in L$, then the reverse $vx \in L \leftrightarrow ux \in L$ is also true by iff.

Transitive:

Let $u \in \Sigma^*$, $v \in \Sigma^*$, $t \in \Sigma^*$

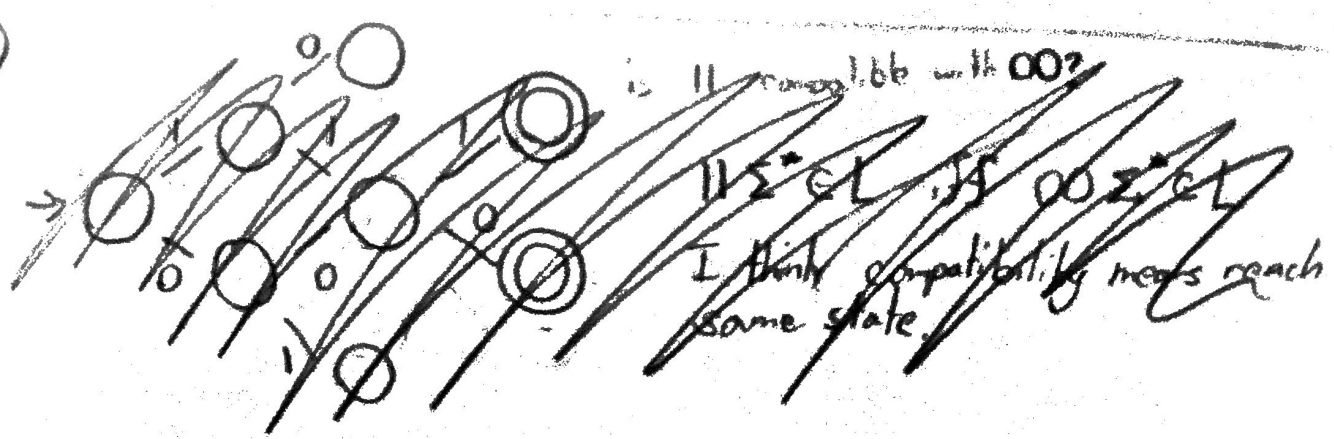
Given $\forall x \in \Sigma^*, ux \in L \leftrightarrow vx \in L \wedge vx \in L \leftrightarrow tx \in L$

Since $ux \in L \rightarrow vx \in L \wedge vx \in L \rightarrow tx \in L$ then $ux \in L \rightarrow tx \in L$

The reverse direction is also true aka $tx \in L \rightarrow vx \in L$ ....

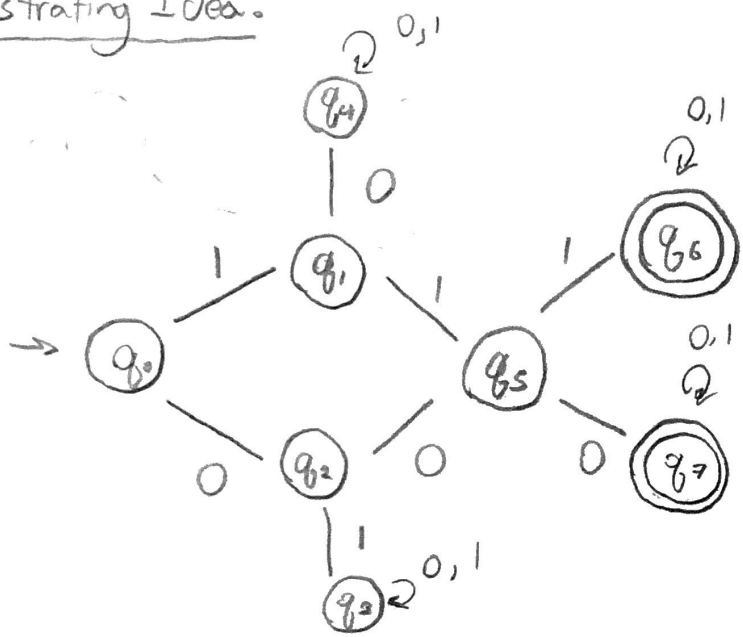
Thus compatibility is an equivalence relation.

#2b)



#2b) Notice that two strings that end in the same state will always be compatible. That is because once in that state, any accepted string x from this state will be shared between the strings of the compatibility class!

Illustrating Idea:



String 11 & 00 are compatible. Why? Because once we travel along 00 & 11, we reach q_5. We can then treat q_5 as a new start state, that only accepts $x \in \Sigma^*$, which is shared between 11 & 00.

Now if L has n states, lets suppose L has greater than n compatibility classes.

⇒ By pigeonhole principle, there must exist a state in L s.t. there are 2 or more compatibility classes that lead up to L.

→ But we know if two strings end in the same state, they will always be compatible. Thus there can only be one equivalence class per state. Which leads to a contradiction.

Thus, there are at most n compatibility classes for a n-state DFA. ▨

2c) Note, compatibility is nearly identical to indistinguishability. Their main difference is compatibility refers to the string leading upto an indistinguishable states & indistinguishability starts at that state.

Thus all we need to do is reduce our DFA to its minimal form, then each unique state will correspond to exactly one equivalence class of the DFA.

Algorithm: Step 1: Find each distinguishable pair of states in our DFA using the algorithm we went over in class.

Step 2: Reduce all indistinguishable states to a single state.

Step 3: Count # of states in our new minimized DFA.

→ The result of step 3 is the # of equivalence classes in our original DFA.

2d) Suppose $L = \{0^n 1^n : n \geq 0\}$ has a finite # of equivalence classes for the compatibility relation.

→ We should be able to convert L to a DFA with finite # of states. (2c)

But of course we can't do that b/c in order to accept a string $w \in L$, we'd need a state to represent each # of 0's & each number of 1's, but since n can be changed arbitrarily we'd have to construct a seperate FSA for each n.

Which is a contradiction. Thus no FSA ⇒ no finite equivalence classes → L is not regular. ■

2c) Show if the number of equivalence classes in a language is finite, then that language L is regular.

To start suppose n equivalence classes in L, $C_1, \ldots, C_n$ & let $s_i \in C_i$, $1 \leq i \leq n$.

We construct the DFA M accepting L, $M = \{Q, \Sigma, \delta, q_0, F\}$

1. $Q = \{q_1, q_2 \cdots, q_n\}$ s.t strings in C_1 end at q_1, C_2 at q_2, ... etc.

2. $\Sigma = \Sigma_L$, where Σ_L is the alphabet used by our language.

4. $q_0 = q_1$, or the state corresponding to our first equivalence class C_1.

5. $F = \{q_i \mid 1 \leq i \leq n, i \in \mathbb{Z}^+ \wedge$ all $w \in C_i$ are accepted by $L\}$

3. We define the transition function δ as the following,

for any $a \in \Sigma$, if $s_i a$ is compatible with s_j, s.t

$1 \leq i \leq j \leq n$, $i, j \in \mathbb{Z}^+$, then $\delta(q_i, a) = q_j$

That is if a string belonging to C_i would belong to C_j given an additional element $a \in \Sigma_L$, then our DFA has a transition from q_i (representing C_i) to q_j (representing C_j) along a.

Thus we construct an DFA for L, showing that it is regular. ∎

#3) Let $L = \{a^i b^j c^k : i, j, k \geq 0 \text{ and } i = 1 \rightarrow j = k\}$

We involve the pumping lemma's properties. Let $p = 2$,

Then,

if $i = 1$, our accepted language becomes $ab^k c^k$.

We choose to represent this language in 3 parts, xyz where

$x = \varepsilon$, $y = a$, $z = b^k c^k$.

→ 1. $\forall i \geq 0 \; xy^i z = a^i b^k c^k \in A$, since any # of a's followed by some equal # of b's then c's is always accepted.

2. $|y| > 0$

3. $|xy| \leq p = 2$

If $i = 0$, our language becomes $b^k c^j$.

we choose $x = b$, $y = b$, $c = b^{k-2} c^j$

→ 1. $\forall i \geq 0 \; xy^i z = b^{k+i} c^j \in A$

2. $|y| > 0$

3. $|xy| \leq p = 2$

If $i \geq 2$,

then $L = a^i b^j c^k$

Let $x = a$, $y = a$, $z = a^{i-2} b^j c^k$ since $i \geq 2$, this is valid.

→ 1. $\forall t \geq 0 \; xy^t z = a^{i+t} b^j c^k \in A$

2. $|y| > 0$

3. $|xy| \leq 2 = p$

that pblm!

Thus all 3 pumping lemma properties are valid for L.

#3b) Notice $L = b^* c^* \cup aaa^* b^* c^* \cup \{ab^i c^i : i \geq 0\}$

Since regular languages are closed under differences & complements

$(L - \underbrace{b^* c^*}_{\text{regular}} - \underbrace{aaa^* b^* c^*}_{\text{regular}})^c = \{ab^j c^k : j,k \geq 0 \wedge j \neq k\}$

Let's make a string $ab^p c^k$, where p is the pumping length & $p \leq k$.

→ Let $x = a$, $y = b$, $c = b^{p-1} c^k$

Then, $|xy| \leq p$ as long as $p \geq 2$, & $|y| > 0$.

→ For property i) take $i = k-p+1$ → $xy^{k-p+1} z = ab^{k-p+1} b^{p-1} c^k = ab^k c^k \notin A$

If $p = 1$, *

→ $x = \varepsilon$, $y = a$, $z = b^j c^k$

→ $|xy| \leq p$ & $|y| > 0$

But $xy^i z = a^i b^j c^k$, then $a^i b^j c^k \notin L$ for $i = 0, \geq 2$ ∎

#3c) The pumping lemma states IF L is a regular language THEN its 3 properties hold. IF L is non-regular, the pumping lemma says nothing about its of L. Thus its properties still hold but L is non-regular.