NFA? ① accept all strings ...
② reject all strings ...

CS 181 EXAM #1
SPRING 2015

NAME Junkyum Kim
UCLA ID 704443811

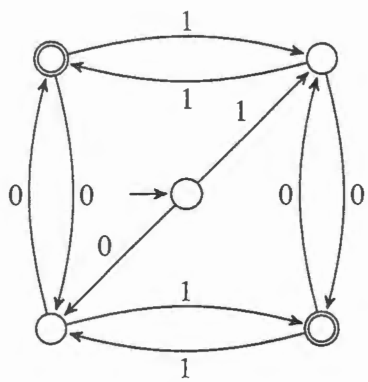
You have 90 minutes to complete this exam. You may assume without proof any statement proved in class.

11 10
00

11.5

(3 pts) **1** Give a simple verbal description of the language recognized by the following DFA.

3



All strings of even length with length at least 2 over the binary alphabet

2 Draw NFAs for the following languages over {0, 1}, taking full advantage of nondeterminism:

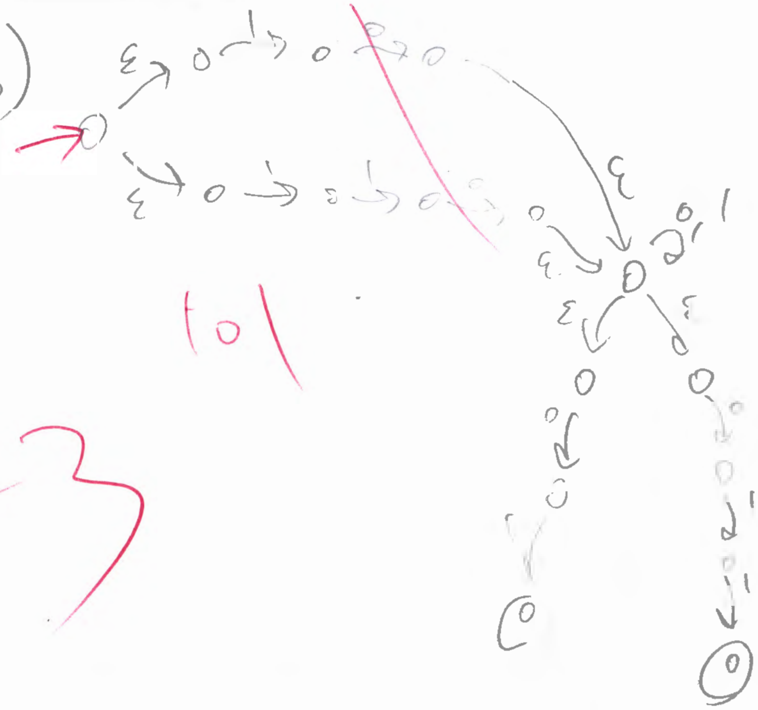
1

(2 pts) **a.** strings such that in every prefix, the numbers of zeroes and ones differ by at most 2;

(2 pts) **b.** strings that begin with 10 or 110, and end with 01 or 011.

a)

b)



101

3

3 Prove that the following languages over a given alphabet Σ are regular:

(2 pts) **a.** strings in which no pair of consecutive characters are identical;

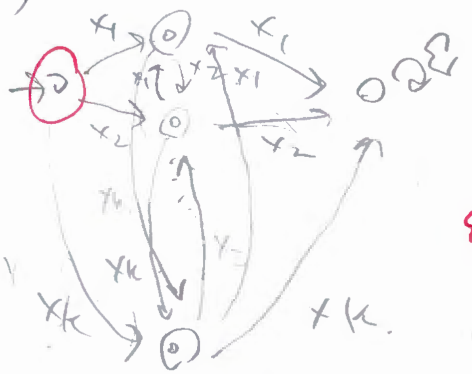
(2 pts) **b.** binary strings in which every even-numbered character is a 0;

(2 pts) **c.** the language $\{3, 6, 9, 12, 15, 18, 21, \ldots\}$ over the $\boxed{\text{decimal alphabet}}$ corresponding to natural numbers that are divisible by 3.

4.5

?

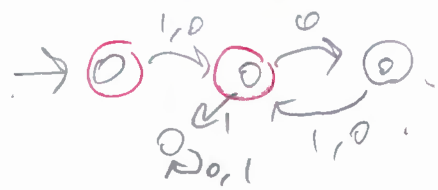
a) For $\Sigma = \{x_1, x_2, \ldots, x_k\}$,



ε: -.25

whenever 2 consecutive characters are identical, go to sink. else, go to a state corresponding to that specific character being consumed, and accept, for some alphabet w/ characters $x_1, x_2, \ldots x_k$. DFA exists for such language, so language is regular.

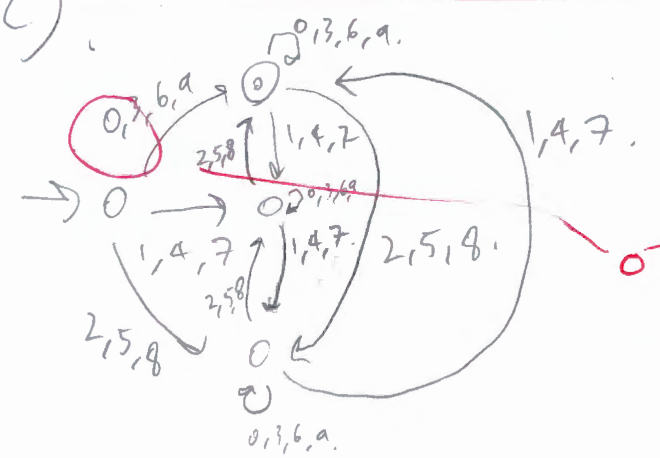
b)



DFA exists for language, so language is regular.

Doesn't accept ε, 1, or 100. -1.

c)



03: -.25

Each state other than initial state represents current string % 3. Since DFA exists for a given language, language is regular.

(3 pts) **4** The symmetric difference of two languages L' and L'', denoted $L'\Delta L''$, is the set of strings that belong to L' or L'' but not both. Prove that regular languages are closed under symmetric difference.

3

$$L'\Delta L'' = \left(\left(L' \cap \overline{(L'\cap L'')}\right) \cup \left(L'' \cap \overline{(L'\cap L'')}\right)\right)$$

Let $L''' = L' \cap L''$. L''' is closed under intersection.

$\overline{L'''}$ is closed under complement.

$L' \cap \overline{L'''}$ is closed under intersection. This represents set of strings that belong to L' but not both $L' \Delta L''$,

$L'' \cap \overline{L'''}$ is closed under intersection. this represents set of strings that belong to L'' but not both $L' \Delta L''$.

finally $(L' \cap \overline{L'''}) \cup (L'' \cap \overline{L'''})$ is closed under union. This represents set of strings that belong to L' or L'' but not both. This is equivalent to $L'\Delta L''$, b/c results, ∴ regular languages are closed under symmetric difference.

(3 pts) **5** Prove or argue to the contrary: adding a finite ~~number of~~ strings to a regular language necessarily results in a regular language.

0

"Adding" means union, not concatenation. −3

True. Proof.

① For any language L, and any finite number of strings w of length k,

② $w = w_1 w_2 \dots w_k$, over any alphabet Σ, w can be represented by a DFA, where

$\bigcirc \xrightarrow{w_1} \circ \xrightarrow{w_2} \circ \dots \xrightarrow{w_k} \circledcirc$

Let this DFA be called L'.

LL' is closed under concatenation

③ For any finite number of times,

$L \underbrace{L'L'L' \dots L'}_{n}$ where n is finite,

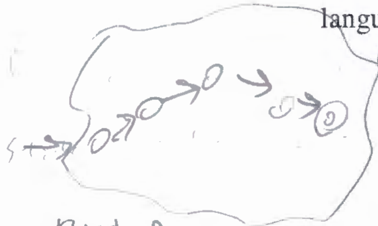
is still closed under concatenation.

∴ adding finite number of strings to a regular language necessarily results in a regular language.

L can be infinite; −3

(3 pts) 6 The *circular shift* of a language L is defined as $L^{\circlearrowleft} = \{uv : vu \in L\}$, where u and v stand for arbitrary strings. For example, $\{1234\}^{\circlearrowleft} = \{1234, 2341, 3412, 4123\}$. Prove that regular languages are closed under circular shift.

0



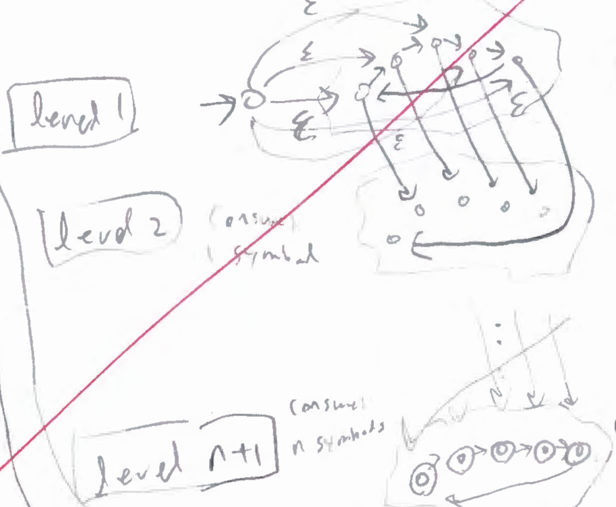
DFA for each string in language L.

(1) For each ~~string~~ of length n, make n+1 copies of original DFA. Make all states rejecting states except (n+1)th copy.

(2) Draw a new start state, & draw ε transition to all states in L.

(3) Make the last accepting state loop back to the original start state (not the newly created start state).

(4) Every time you consume a symbol, loop along as you would in new NFA, while also moving down in level.



(5) Repeat until you consume n symbols. If you did not "blow up" before reaching level n+1, accept that string.

(6) Repeat steps (1)~(5) for each string in L. Draw ultimate start state that connects all other start states w/ ε transition.

(7) b/c NFA exists for circular shift, regular languages are closed under circular shift.

(3 pts) 7 Describe an algorithm that takes as input a DFA and determines whether the automaton recognizes the empty language, ∅.

0

You would take the DFA, start the automaton, and not give any symbols.

$\varepsilon \notin \emptyset$. −3

If automaton accepts, given DFA accepts empty language, ∅.

If automaton doesn't accept, given DFA doesn't accept empty language, ∅.

To make sure your automaton fully loaded up, you could test by inputting a symbol. If state changes, your automaton was fully functioning, so whatever previous output was determines whether empty language is accepted or not by that DFA.

If you input a symbol & there's no change in states, your automaton wasn't fully functioning, meaning you had to wait longer & give the automaton more time.