NAME \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CS 420/520 Theory of Computation, Spring 2019 at Indiana State University, taught by Jeff Kinne

Quiz 7 / Exam 1 - regular languages

Points - each part is graded as 1 point, half credit is possible. Total # points = 10

1) For each of these, if the following language is regular, give an RE, DFA, or NFA for the language. If it is not regular, prove it is not regular using the pumping lemma. SKIP ONE.

1a.

L = {strings of a's and b's without three b's in a row}

In the language: empty string, aabab, bbabbab, ababa, aaa

Not in the language: bbb, abbababbbaab

See exam1-1a-dfa.jff

1b.

L = {strings of a's and b's with more b's than a's}

In the language: bba, abb, bbaababab

Not in the language: bbaa, aaa, a, aba

Claim: L is not a regular language.

Proof: Suppose L is regular (for the purpose of contradiction). Let p be from the pumping lemma for L.

Let w = ap bp+1. Note that w is in L.

Then the pumping lemma guarantees that w can be written as w = xyz with |xy| <= p, |y| >= 1, and xyiz is in L for all i >= 0.

Consider xyiz for i=3. Since |xy| <= p, y is ak for some k >= 1. Then  
xy3z = ap+j bp+1 for some j >= 2. Since this has more a’s than b’s, it is not in L.

This would violate the pumping lemma, which means L cannot be regular.

1c.

L = {strings of a's and b's that do not contain the string abb}

In the language: abab, ab, a, b, bbab, abaab

Not in the language: babb, ababba, babba, baabba

See exam1-1c-dfa.jff

1d.

L = {strings of 0's and 1's that cannot be written as www,

that is - first third = middle third = last third}

In the language: 010100, 010010011, 00, 010, 10101

Not in the language: 010010010

Claim: L is not regular.

Proof: Suppose L is regular (for the purpose of contradiction). And let p be the pumping length for L from the pumping lemma.

Let w = 0p1p 0p1p 0p1p Note that w is in L. Then w can written as w = xyz with   
|xy| <= p, |y| >=1, and xyiz is in L for any i >= 0.

Consider xy0z = xz. Note that xy is all 0’s, meaning y = 0j for some j >= 1.  
Then xz = 0p-j1p 0p1p 0p1p Since there are fewer 0’s in the first part than the other 2, xz is not w’ w’ w’ for some w’. Then xz is not in L. Therefore it must be that L is not regular.

2) For each of the following, give a Python3 regular expression for the given

language. Include the begin and end marker symbols. SKIP ONE.

2a. Integers

In the language: 0001, 1010101, 1341343, -123, -0, 0

Not in the language: 1.2, pi, -,

"-?(\\d)+"

Or

r"-?(\d)+"

or

import re

re.match("-?([\\d)](file://d))+", "123")

2b. Fractions

In the language: 1/2, 3/4, 0/4, -1/3, 234234/2342334

Not in the language: 1.2, pi, -, 4/0, 234/, 2/-3, 234

"-?([\\d)+/([1-9](\\d)\*)](file://d)+/(%5b1-9%5d(//d)*))" # note this would not allow 1/023

2c. Sentence of the form - "What is your name? NAME. Hello NAME."

Note - NAME can be any combination of letters, first letter upper case and the rest lower.

Note – you should match “What is your name?” and “Hello “ literally – don’t make up RE’s for sentences like that, just match those exact sentences.

"What is your name? ([A-Z][a-zA-Z]\*). Hello \\1."

3) Prove by induction or contradiction. SKIP ONE.

3a. The cube root of 7 is irrational.

Proof by contradiction. Assume 71/3 is rational, so 71/3 = a/b where a and b are integers with no common factors. Then 7 = a3 / b3, so 7 b3 = a3. Then there must be a 7 in the prime factorization for a, so a = 7k. 7 b3 = (7k)3 = 73 k3, so b3 = 72 k3 , so there must be a 7 in the prime factorization of b. This contradicts our assumption that a and b don’t have any common factors, so it must be that 71/3 is not rational.

3b. 1/2 + 1/(2\*3) + 1/(3\*4) + ... + 1/(n\*(n+1)) = n/(n+1) for every integer n >= 1

Proof by induction.

Base case

– check for n=1. LHS is 1/(1 \* (1+1)) = 1/2. RHS = 1/(1+1) = 1/2.

- check for n=2. LHS is 1/2 + 1/(2\*3). RHS = 2/3. LHS = 1/2+1/6 = 4/6 = 2/3.

Inductive step. Assume true for all n up to k. Show it’s true for n = k+1.

1/2 + 1/(2\*3) + ... + 1/(k\*(k+1)) + 1/((k+1)\*(k+2))

= k/(k+1) + 1/((k+1)\*(k+2)) // by inductive assump

= (k(k+2) + 1) / ((k+1)\*(k+2))

= (k\*k +2\*k + 1) / ((k+1)\*(k+2))

= ((k+1)(k+1)) / ((k+1)\*(k+2))

= (k+1) / (k+2) = (k+1) / ( (k+1) + 1)

3c. For all integers k, if k\*k is even then k is even.

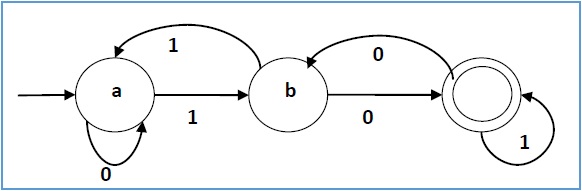
p => q

Proof by contrapositive. Contrapositive of p => q is (not q) => not p. And these are equivalent. We will show if (not k is even) => (not k\*k is even).

Let k be an integer that is not even, so k is odd and can written k = 2n + 1.  
k \* k = (2n + 1) \* (2n + 1) = 4n\*n + 4n + 1 = 2(2\*n\*n + 2\*n) + 1, which means that k\*k is odd. In other words (not k\*k is even).

Therefore done.

4) Write a table for the transition function of the following NFA or DFA. Let the third state have the label “c”, so the set of states is Q = {a, b, c}

<https://www.tutorialspoint.com/automata_theory/images/dfa_graphical_representation.jpg>  


1. 1

a a. b

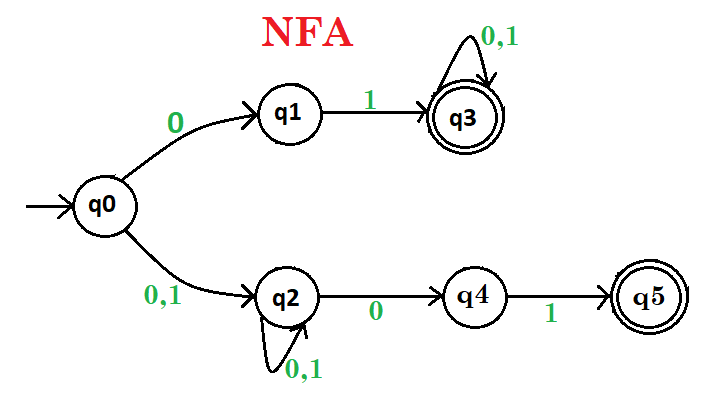
b. c. a

c. b. c

And initial state is a, final states = {c}

5) Describe the language accepted by the following NFA or DFA.

<https://cdncontribute.geeksforgeeks.org/wp-content/uploads/either01nfa-1.png>



Along the top – 01 (0 | 1)\*  
Along the bottom – (0 | 1) (0 | 1)\* 01

Language is: starts with 01, or is at least three characters and ends with 01

Simpler: starts or ends with 01

6) Prove the following. The class of regular languages is closed under

intersection.

Note: need to show that if L and L’ are two regular languages, then so is   
(L intersect L’). (L intersect L’) – the “yes” instances must be “yes” for both L and L’.

Proof: Given two regular languages L and L’, we must construct DFA or NFA or RE for   
(L intersect L’).

Proof idea: assume a DFA D for L and a DFA D’ for L’. We’ll construct a DFA D’’ that recognizes (L intersect L’). D’’ will keep track of what is happening in both D and D’. How many different states does D’’ need? # states in D \* # states in D’.  
Each state for D’’ is written like (q,q’) for q from D and q’ from D’.

Transition function for D’’  
delta( (q,q’), x) = ( delta\_D(q, x), delta\_D’(q’, x) )

And check the book again.

After describing D’’, need to show:

If x accepted by both D and D’, then it is accepted by D’’.  
If either D or D’ rejects, then D’’ rejects.