Lectures and Reading. After today’s lecture on regular expressions and their equivalent NFAs (with \( \epsilon \)-transitions), Friday’s lecture (continued into next Monday) will complete the cycle of showing those two formalisms recognize the same class of languages as DFAs. The class is standardly denoted by \( \text{REG} \) for the class of regular languages. Then lectures will move into the Myhill-Nerode technique for showing that certain other languages are non-regular and for showing that certain sets of strings must all be processed to distinguished states by a DFA. So read Debray’s notes up through the end of section 3 on page 11, in tandem with the slides notes called “Extra notes for Weeks 2–3” on the course webpage (third numbered item under Required Reading). Then please also read the “Myhill-Nerode” handout (in the 5th item).

My regular office hours are Tuesdays 2–3:30pm, Wednesdays 1–1:50pm (before class). But on the 10th I am giving a presentation in CSE501 at 2pm, so I’ll have hours MW 1–1:50pm instead. It is also often possible to see me right after a class.

The first two problems are based on lectures through this week; the latter two need material next week. With a weekly-HW scheme, this would be divided into two smaller sets.

(1) Let \( L_3 \) be the language of binary strings that represent positive multiples of 3. Define \( L \) to be the language of binary strings that do not have a substring in \( L_3 \). That is, \( L \) is the complement of \( (0 + 1)^* \cdot L_3 \cdot (0 + 1)^* \). Follow these steps to characterize the language \( L \).

(a) Design a DFA \( M_3 \) such that \( L(M) = L_3 \). It is possible that you might find this just in the course of innocently reading online notes, and that’s OK, actually. (Just 3 pts., anyway.)

(b) Now make some small additions to your DFA \( M_3 \) to create an NFA \( N_3 \) such that \( L(N_3) = (0 + 1)^* \cdot L_3 \cdot (0 + 1)^* \). (3 pts.)

(c) Convert \( N_3 \) into an equivalent DFA \( M' \) such that \( L(M') = (0 + 1)^* \cdot L_3 \cdot (0 + 1)^* \). The sets-of-states construction to do this will be covered Friday, but you may well have had it in a previous course. (12 pts.)

(d) Complement the final states of \( M' \) to get the needed \( M \). (3 pts.)

(e) Then answer some more short questions: First, is \( L \) infinite? Explain your answer briefly. (3 pts.)

(f) The string 11 sends \( M \) to a dead state wherever you start from because it is 3 in binary, so it makes a substring that belongs to \( L_3 \). Find a similar “dead substring” that does not have two consecutive 1s in it. (3 pts.)

(g) By “sight-reading” your machine \( M \)—or by following the formal algorithm—find a regular expression \( \alpha \) such that \( L(\alpha) = L_3 \). (9 pts., for 36 total on the problem)

(2) Design both nondeterministic finite automata \( N_a, N_b, N_c \) and regular expressions \( r_a, r_b, r_c \) that denote the following three languages described in prose. (It is OK for your
NFAs to have ε-arcs, and it is fine if one or more are DFAs since a DFA “Is-A” NFA.) All use the alphabet Σ = {0, 1}. (3 × (6 + 6) = 36 points total)

(a) \( L_a \) = the set of binary strings in which the substring 10 occurs an odd number of times.
(b) \( L_b \) = the set of binary strings of the form \( x = y00z \) where \(|z|\) is odd.
(c) \( L_c \) = the set of binary strings having an occurrence of the substring 10 that do not have an occurrence of 11 after it.

(3) Convert the following NFA \( N \) into an equivalent DFA \( M \):

![Diagram of NFA](image)

Also answer (for \( 18 + 12 = 30 \) pts.):

(a) Is there a string \( u \) that \( N \) can process from its start state to each and every one of its four states? Viewing your DFA \( M \), give a shortest \( u \) if so.

(b) Is there a string \( v \) that \( N \) cannot process from start to any state? Again give a shortest such \( v \).

(c) Is there a string \( w \) such that no matter what string \( y \) follows it, the string \( wy \) is accepted? Again use your \( M \).

(d) Stronger than (b) and counter to (c), is there a string \( z \) that \( N \) cannot process at all, not from any of its states \( p \) to any state \( q \)? Again use your \( M \) to explain your answer.

(4) Define \( L = \{y00z : |y| = |z| \} \). Use the Myhill-Nerode technique to prove that \( L \) is not a regular language. (18 pts., for 120 total on the set)