CSCE-433 Formal Languages & Automata
CSCE-627 Theory of Computability

Spring 2022

Instructor: Dr. Jianer Chen
Office: PETR 428
Phone: 845-4259
Email: chen@cse.tamu.edu
Office Hours: MWF 10:30–11:30am

Senior Grader: Avdhi Shah
Office: N/A
Phone: tba
Email: avdhi.shah@tamu.edu
Office Hours: tba

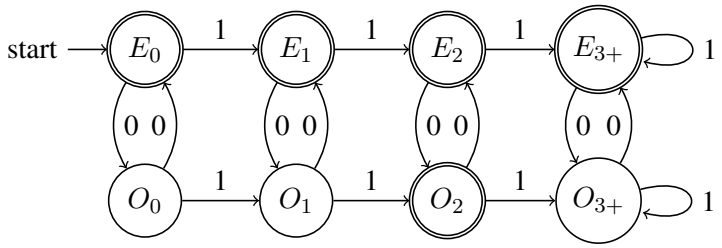
Solutions to Assignment #2

1. Draw the state diagram of a DFA that recognizes the following language:

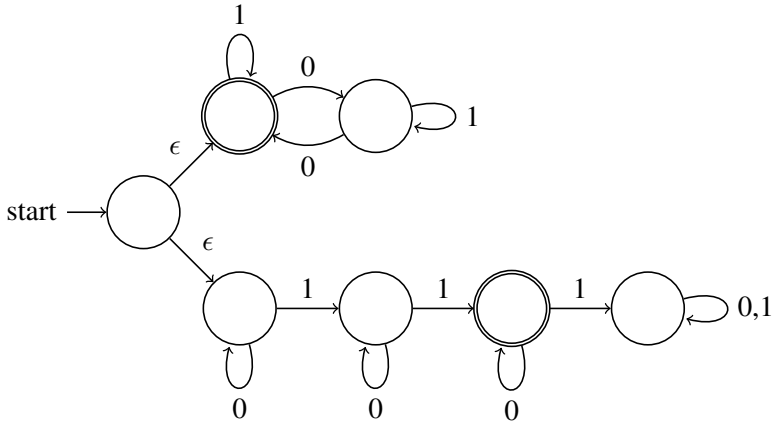
$$L = \{w \mid w \text{ contains an even number of } 0\text{'s or contains exactly two } 1\text{'s}\}$$

Then draw the state diagram of an NFA for the same language. The alphabet is $\{0, 1\}$. Try to use as few states as you can. Discuss the relative ease in designing the NFA versus the DFA.

Solution. The following is a DFA that accepts L. In the diagram, state E_i (resp. O_i), for $i = 0, 1, 2, 3+$, means that the number of 0's seen so far is even (resp. odd) and the number of 1's seen so far is i.



The following is an NFA that accepts L. The top row checks for an even number of 0's while the bottom row checks for exactly two 1's.



It was much easier to design the NFA than the DFA: we can use the two ϵ-transitions to "guess" whether the string might have an even number of 0's or might have exactly two 1's. Then we can think about how to check each condition separately. □

2. [Textbook, page 85, Exercise 1.14.]

a) Show that if M is a DFA that recognizes language B, swapping the accept and nonaccept states in M yields a new DFA recognizing the complement of B. Conclude that the class of regular languages is closed under complement.

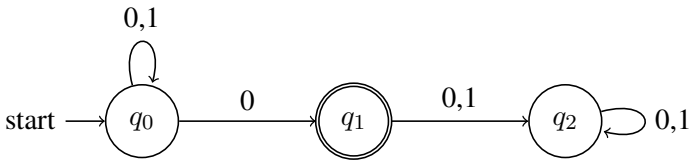
b) Show by giving an example that if M is an NFA that recognizes language C, swapping the accept and nonaccept states in M doesn't necessarily yield a new NFA that recognizes the complement of C. Is the class of languages recognized by NFAs closed under complement? Explain your answer.

Proof.

a) Let M' be the DFA that is obtained by swapping the final (i.e., accept) and nonfinal (i.e., noaccept) states in the DFA M. Suppose w is a string that is in B, and hence is accepted by M. When M' has w as its input, it follows the same sequence of states, but the last state of the computation, which was a final state in M, thus is a nonfinal state in M'. Therefore, M' rejects w. On the other hand, suppose that w' is a string that is not in B, and hence is rejected by M. Then when M' has w' as its input, it follows the same sequence of states, but the last state of the computation, which was a nonfinal state in M, thus is a final state in M'. Therefore, M' accepts w'. This proves that the DFA M' accepts exactly those strings that are not in B, i.e., M' accepts the complement of B.

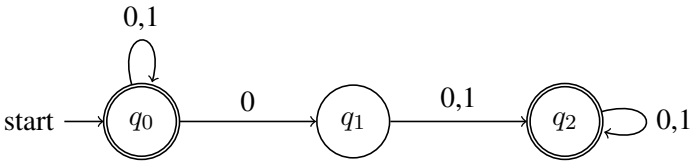
This shows that if B is a regular language, then the complement $\overline{B}$ of B is also a regular language. In conclusion, the class of regular languages is closed under complement.

b) Let $\Sigma = \{0, 1\}$ and $C = \{w | w \text{ ends in } 0\}$. Here is an NFA M for C:



Note that for any string w ending in 0, we have a path, which loops on state q_0 until it reads the last 0. Then the path moves from state q_0 to the final state q_1. That is, the NFA M accepts w.

Swapping the accept and non-accept states in M gives M':



The NFA M' actually accepts *all* 0-1 strings: for any string w, the path simply looping at state q_0 is a path accepting w. In particular, the NFA M' accepts all strings ending in 0, which are in C so are not in $\overline{C}$. This proves that M' does not accept the complement of C.

Is the class of languages recognized by NFAs closed under complement?

Yes: by the theorem proved in class, languages accepted by NFAs are regular languages (i.e., they are accepted by DFAs), while part a) showed that regular languages are closed under complement. □

3. [Textbook, page 86, Exercise 1.16.] Covert the following two nondeterministic finite automata to equivalent deterministic finite automata.

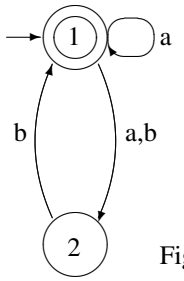


Figure (a)

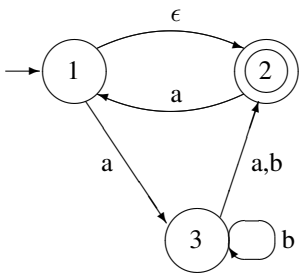


Figure (b)

Solution. Instead of using the algorithm given in the textbook, we follow the algorithm given in class. That is, we start with the start state $s_0 = \epsilon$-closure$(\{1\})$ of the DFA M_D, which is the ϵ-closure of the start state 1 in the NFA M_N, and construct only those states in the DFA M_D that are reachable from s_0.

The constructions are given in the following tables, where each row corresponds to a state in the DFA M_D. The first column gives the state name in the DFA M_D, which is a subset of states in the NFA M_N. The second and third columns give the subsets reacheable from the subset in the first column following the arc labeled with the symbols a and b, respectively. Note that the states in the DFA M_D are the ϵ-closures of the subsets obtained in columns 2 and 3.

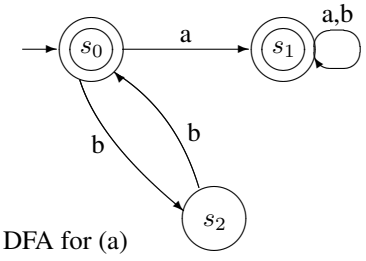
(a) The DFA M_D for figure (a) is (states s_0 and s_1 are final states because they contain the final state 1 of the given NFA M_N):

State in M_D	a	b
$\mathbf{s_0} = \epsilon$-closure$(\{1\}) = \{1\}$	$\{1,2\}$	$\{2\}$
$\mathbf{s_1} = \epsilon$-closure$(\{1,2\}) = \{1,2\}$	$\{1,2\}$	$\{1,2\}$
$s_2 = \epsilon$-closure$(\{2\}) = \{2\}$		$\{1\}$

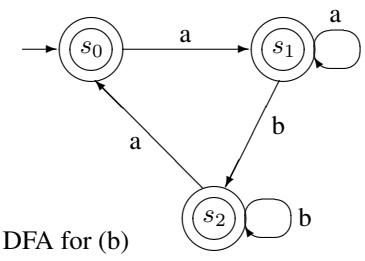
(b) The DFA M_D for figure (b) is (states s_0, s_1 and s_2 are final states because they all contain the final state 2 of the given NFA M_N):

State in M_D	a	b
$s_0 = \epsilon$-closure$(\{1\}) = \{1,2\}$	$\{1,3\}$	
$\mathbf{s_1} = \epsilon$-closure$(\{1,3\}) = \{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$
$\mathbf{s_2} = \epsilon$-closure$(\{2,3\}) = \{2,3\}$	$\{1,2\}$	$\{2,3\}$

The following figures are the DFAs given in the above tables:



DFA for (a)



DFA for (b)

□

4. Give regular expressions that generate the following languages, assuming the alphabet $\{0, 1\}$:

a) $L_1 = \{w \mid w$ starts with 0 and has odd length, or starts with 1 and has even length$\}$;

b) $L_2 = \{w \mid w$ does not contain the substring $110\}$;

c) $L_3 = \{w \mid w$ contains an even number of 0's, or contains exactly two 1's$\}$.

Solution. **a)** Strings of even length can be given by the regular expression $((0 \mid 1)(0 \mid 1))^*$. Thus, the regular expression for language L_1 is:

$$(0((0 \mid 1)(0 \mid 1))^*) \mid (1(0 \mid 1)((0 \mid 1)(0 \mid 1))^*)$$

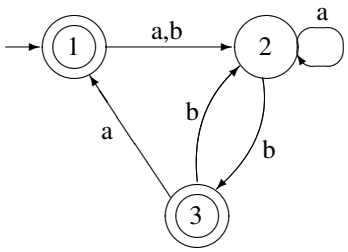
b) Let x be such a string in L_2. If for a symbol 1 in x there are still symbol 0's on the right of this symbol 1, then this symbol 1 must be followed immediately by one or more symbol 0's (otherwise we will always be able to fine a string 110 in x). That is, this symbol 1 must have the pattern 100^*. On the other hand, if there is no symbol 0 on the right of the symbol 1, then all symbols from this symbol 1 to the end of the string x are 1's, whose pattern should be 1^*. Moverover, the string x can start with a subtring with arbitrarily many 0's, i.e., a pattern 0^*. Thus, the regular expression for the language L_2 is

$$0^*(100^*)^*1^*$$

c) The strings that contain an even number of 0's can be given by the regular expression $1^*(01^*01^*)^*1^*$, while the strings that contain exactly two 1's can be given by the regular expression $0^*10^*10^*$. Thus, the regular expression for the language L_3 is

$$(1^*(01^*01^*)^*1^*) \mid (0^*10^*10^*)$$ □

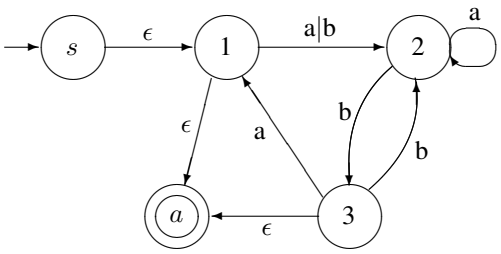
5. [Textbook, page 86, Exercise 1.21(b)] Use the procedure described in Lemma 1.60 to convert the following automaton to an regular expression.



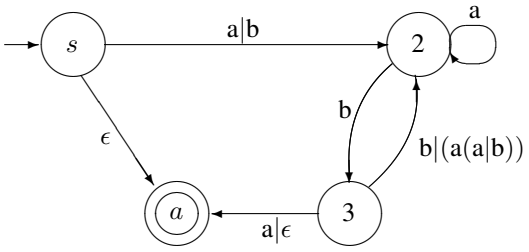
Solution. We follow the procedure given in Lemma 1.60, which was also discussed in detail in class.

See the next page.

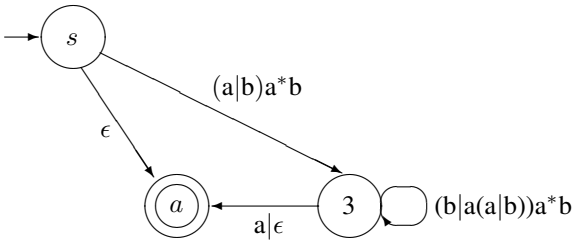
Step 1. Introduce a new start state s and a new final state a:



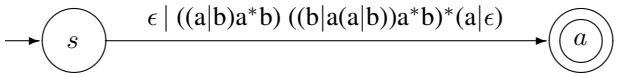
Step 2: remove state 1.



Step 3: remove state 2.



Step 4: remove state 3.

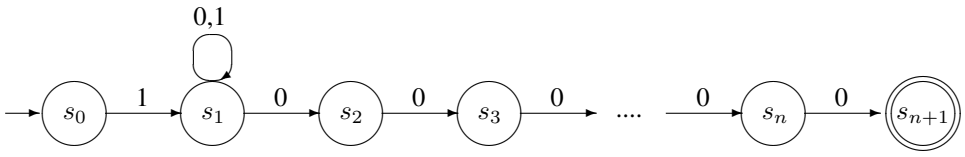


Therefore, the regular expression for the given automaton is: $\epsilon \mid$ ((a|b)a*b) ((b|a(a|b))a*b)*(a|ϵ). □

6. [CSCE-433 Students only] Let L_n be the set of all binary strings of length at least n that are the base-2 representations of integers that are a multiple of 2^n. Show that for any fixed $n \geq 2$, L_n is a regular language.

Proof. It is imperative to note here that n is a *fixed* integer. Thus, we are considering a template such as L_2, L_3, or L_{100}, etc. Therefore, when we consider the language L_n, we assume that n is fixed for all strings in the language.

A binary number is a multiple of 2^n if its binary representation ends with n 0's. It is quite easy to construct a NFA M that accepts such strings, i.e., M accepts the language L_n. To be more specific, we assume that a positive binary number must start with a 1. Note that 0 is not in the language L_n because by definition, all strings in L_n have length at least n. The NFA M is given by the following state diagram:



Alternatively, it is easy to see that the regular expression $1(0|1)^*0^n$ describes the language L_n, where 0^n is a shorthand for the contatenation of n 0's.

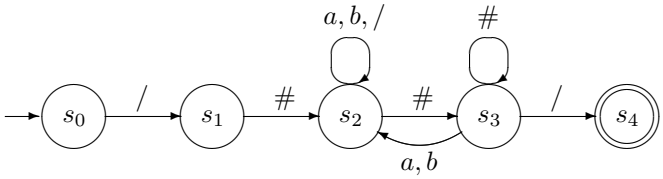
Thus, both the construction of the NFA M above and the regular expression given above prove that the language L_n is a regular language. □

7. [CSCE-627 Students only. Textbook, page 87, Exercise 1.22.] In certain programming languages, comments appear between delimiters such as /# and #/. Let C be the language of all valid delimited comment strings. A member of C must begin with /# and end with #/ but have no intervening #/. For simplicity, assume that the alphabet for C is $\Sigma = \{a, b, /, \#\}$.

a) Give a DFA that recognizes C.

b) Give a regular expression that generates C.

Solution. a) The DFA that recognizes C is shown below:



The two states s_2 and s_3 need some explanations. The state s_2 marks the situation that after seeing the head "/#" of the comment, the last symbol seen is not "#". Thus, we are safely in the middle of the comment. The state s_3 marks the situation that the last seen symbol is "#". Thus, (1) if the next symbol is "/", then we should have reached the end of the comment; (2) if the next symbol is "#", then we stay in state s_3; and (3) if the next symbol is "a" or "b", then we move back to the "safe" state s_2. Therefore, once we see the first appearance of the pattern "#/", we should move to the final state s_4. Note that by our convention, we assume that there is an implicit "trap state" such that if an arc labeled with a symbol is not shown from a state, then we assume that arc goes to the trap state so the input will not be accepted. In particular, if "#/" is not at the end of the string, then at the state s_4, the symbol after "#/" will lead to the trap state so the input will not be accepted by the automaton.

b) We can use the procedure given in the textbook (and in class) to construct a regular expression that describes the language accepted by the automaton constructed in a), i.e., the language C. The regular expression is:

$$(/\#)(a|b|/)^*\#(\#|((a|b)(a|b|/)^*\#))^*/$$

The construction detail is quite simple (much simpler than that for Question 5), thus is omitted. □