# Homework Assignment #3

## Due Time/Date

This assignment is due 2:10PM Tuesday, March 31, 2020. Late submission will be penalized by 20% for each working day overdue.

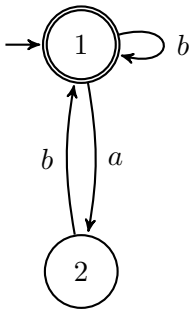
## How to Submit

Please use a word processor or scan hand-written answers to produce a single PDF file. Name your file according to this pattern: "b057050xx-hw3". Upload the PDF file to the Ceiba course site for Theory of Computing 2020: https://ceiba.ntu.edu.tw/1082theory2020. You may discuss the problems with others, but copying answers is strictly forbidden.

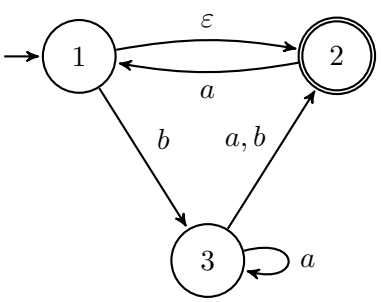
## Problems

(Note: problems marked with "Exercise X.XX" or "Problem X.XX" are taken from [Sipser 2013] with probable adaptation.)

1. (Exercise 1.7; 10 points) For each of the following languages, give the state diagram of an NFA, with the specified number of states, that recognizes the language. In all parts, the alphabet is $\{0, 1\}$.
   (a) The language $\{w \mid w$ contains the substring 1010, i.e., $w = x1010y$ for some $x$ and $y\}$ with five states
   (b) The language $1^*0^+1^*$ with three states
2. (Exercise 1.14; 10 points) 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 you answer.
3. (Exercise 1.16; 20 points) Use the construction given in Theorem 1.39 (every NFA has an equivalent DFA) to convert the following two NFAs to equivalent DFAs.



(a)



(b)

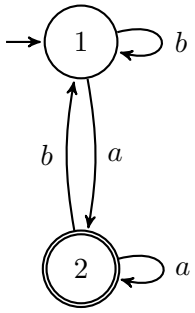
4. (Exercise 1.18; 10 points) Use the procedure described in Lemma 1.55 to convert the regular expression $(0 \cup 1)^*110(0 \cup 1)^*$ to an NFA.

5. (Exercise 1.20; 10 points) Give regular expressions generating the following languages:

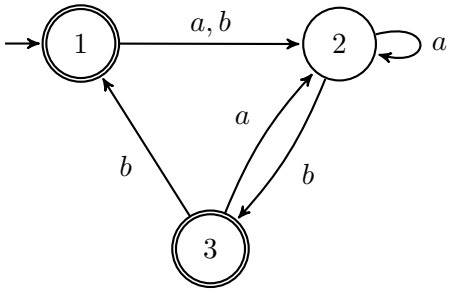
   (a) $\{w \mid$ every odd position of $w$ is a $\mathtt{1}\}$ (Note: see $w$ as $w_1w_2 \cdots w_n$, where $w_i \in \{\mathtt{0}, \mathtt{1}\}$)

   (b) $\{w \mid w$ doesn't contain the substring $\mathtt{011}\}$

6. (Exercise 1.21; 20 points) Use the procedure described in Lemma 1.60 to convert the following finite automata to regular expressions.

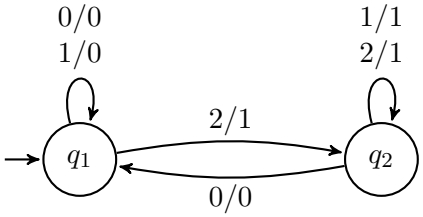


(a)

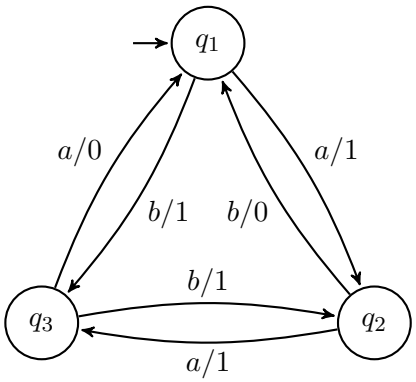


(b)

7. (Exercise 1.24; 10 points) A *finite-state transducer* (FST) is a type of deterministic finite automaton whose output is a string rather than *accept* or *reject*. The following are state diagrams of finite state transducers $T_1$ and $T_2$.



$T_1$



$T_2$

Each transition of an FST is labeled with two symbols, one designating the input symbol for that transition and the other designating the output symbol. The two symbols are written with a slash, /, separating them. In $T_1$, the transition from $q_1$ to $q_2$ has input symbol 2 and output symbol 1. Some conditions may have multiple input-output pairs, such as the transition in $T_1$ from $q_1$ to itself. When an FST computes on an input string $w$, it takes the input symbols $w_1 \cdots w_n$ one by one and, starting from the start state, follows the transitions by matching the input labels with the sequence of symbols $w_1 \cdots w_n = w$. Every

time it goes along a transition, it outputs the corresponding output symbol. For example, on input 2212011, machine $T_1$ enters the sequence of states $q_1, q_2, q_2, q_2, q_2, q_1, q_1, q_1$ and produces output 1111000. On input `abbb`, $T_2$ outputs 1011. Give the sequence of states entered and the output produced in each of the following parts.

(a) $T_1$ on input 102012

(b) $T_2$ on input `babbba`

8. (Exercise 1.25; 10 points) Read the informal definition of the finite state transducer given in Exercise 1.24. Give a formal definition of this model, following the patterns in Definition 1.5 (Page 35). Assume that an FST has an input alphabet $\Sigma$ and an output alphabet $\Gamma$ but not a set of accept states. Include a formal definition of the computation of an FST. (Hint: an FST is a 5-tuple. Its transition function is of the form $\delta : Q \times \Sigma \longrightarrow Q \times \Gamma$.)