hw2

Graded

Student

Giancarlos Marte

Total Points

19 / 30 pts

Question 1

NFAs — **7.5** / 10 pts

1.1 **nfa formal description** — **5** / 5 pts

- ✔ **– 0 pts** Correct

1.2 **accepting computation or no** — **2.5** / 5 pts

- ✔ **– 0.5 pts** a correct, wrong reason
- ✔ **– 1 pt** b incorrect
- ✔ **– 1 pt** e incorrect
- computation must start in start state, missing q3 in a

Question 2

DFAs vs NFAs — **9** / 11 pts

2.1 **differences** — **4** / 4 pts

- ✔ **– 0 pts** Correct

2.2 **equivalent machines** — **2** / 2 pts

- ✔ **– 0 pts** Correct

2.3 **DFA to NFA** — **3** / 5 pts

- ✔ **– 1 pt** incorrect delta, output must be set of one DFA state
- ✔ **– 1 pt** incorrect states, should be same as DFA

Question 3

closed operation

1.5 / 8 pts

- **− 1 pt** Did not state what is to be proved
- **− 1.5 pts** Not enough explanation to how B and C become OP(B,C)
- **− 1 pt** Jumped to conclusions
- **− 1 pt** Final statement does not justify closure
- **− 2 pts** Did not use FSM/sets/other valid way to prove

Although you do use some theorems, you do not actually prove anything, you need to use FSM's to prove this theorem. Look back at the slides for more info

Question 4

readme

1 / 1 pt

- **− 0 pts** Correct

Question assigned to the following page: 1.1

1.1

$N = (Q, \Sigma, \delta, q_{start}, F)$

$Q = (q_0, q_1, q_2, q_3, q_4, q_5, q_6)$

$\Sigma = (0, 1)$

	0	1	ε
q_0	$\{q_5\}$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$\{q_0, q_2\}$
q_2	$\emptyset$	$\{q_3\}$	$\emptyset$
q_3	$\{q_3\}$	$\{q_3, q_4\}$	$\emptyset$
q_4	$\emptyset$	$\emptyset$	$\emptyset$
q_5	$\{q_5, q_6\}$	$\{q_5\}$	$\emptyset$
q_6	$\emptyset$	$\emptyset$	$\emptyset$

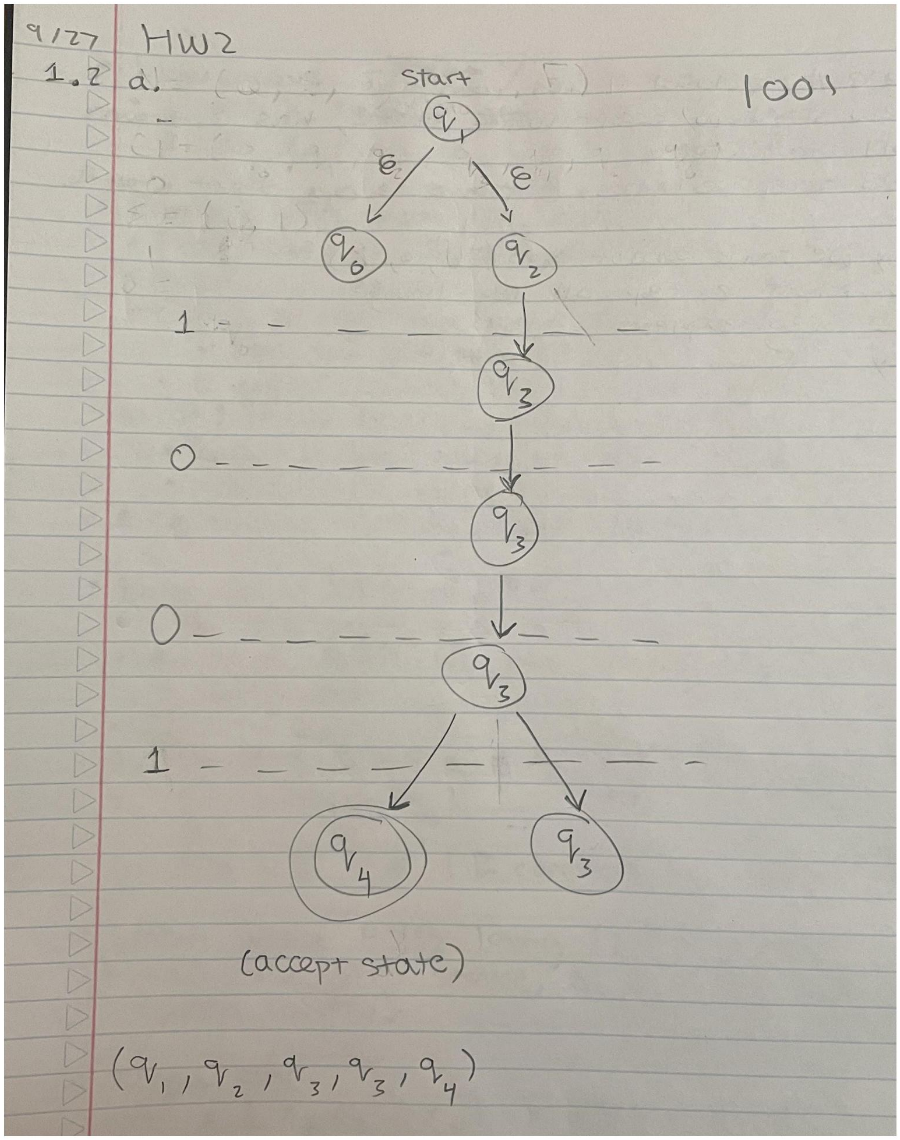
$q_{start} = q_1$

$F = \{q_4, q_6\}$

Question assigned to the following page: 1.2

1.2

a. $\hat{\delta}(q_1, 1001)$

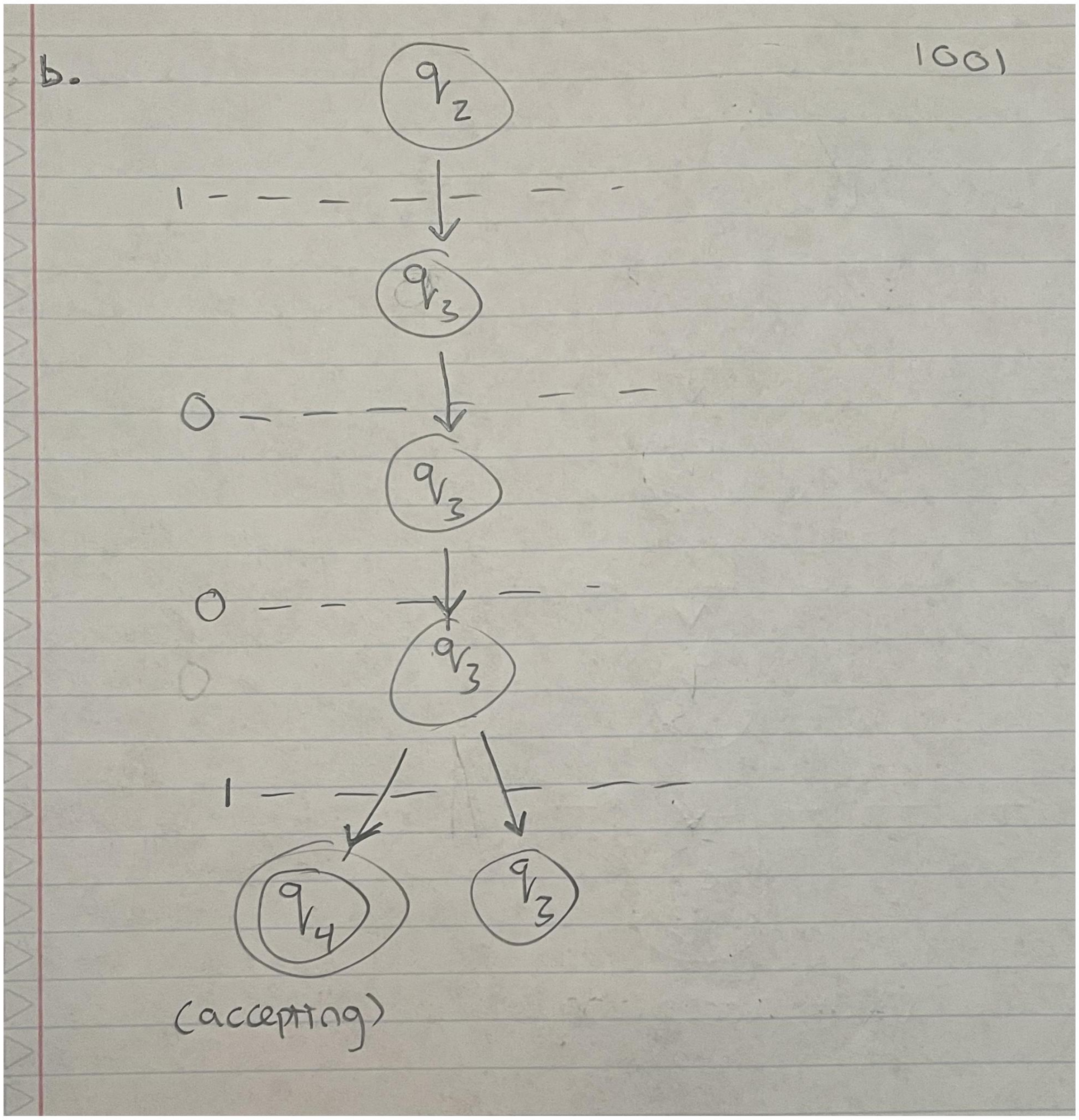


There is a resulting state that is accepting, therefore this computation is accepting. The possible sequence of states is:

$(q_1, q_2, q_3, q_3, q_4)$

Question assigned to the following page: 1.2

b. $\hat{\delta}(q_2, 1001)$

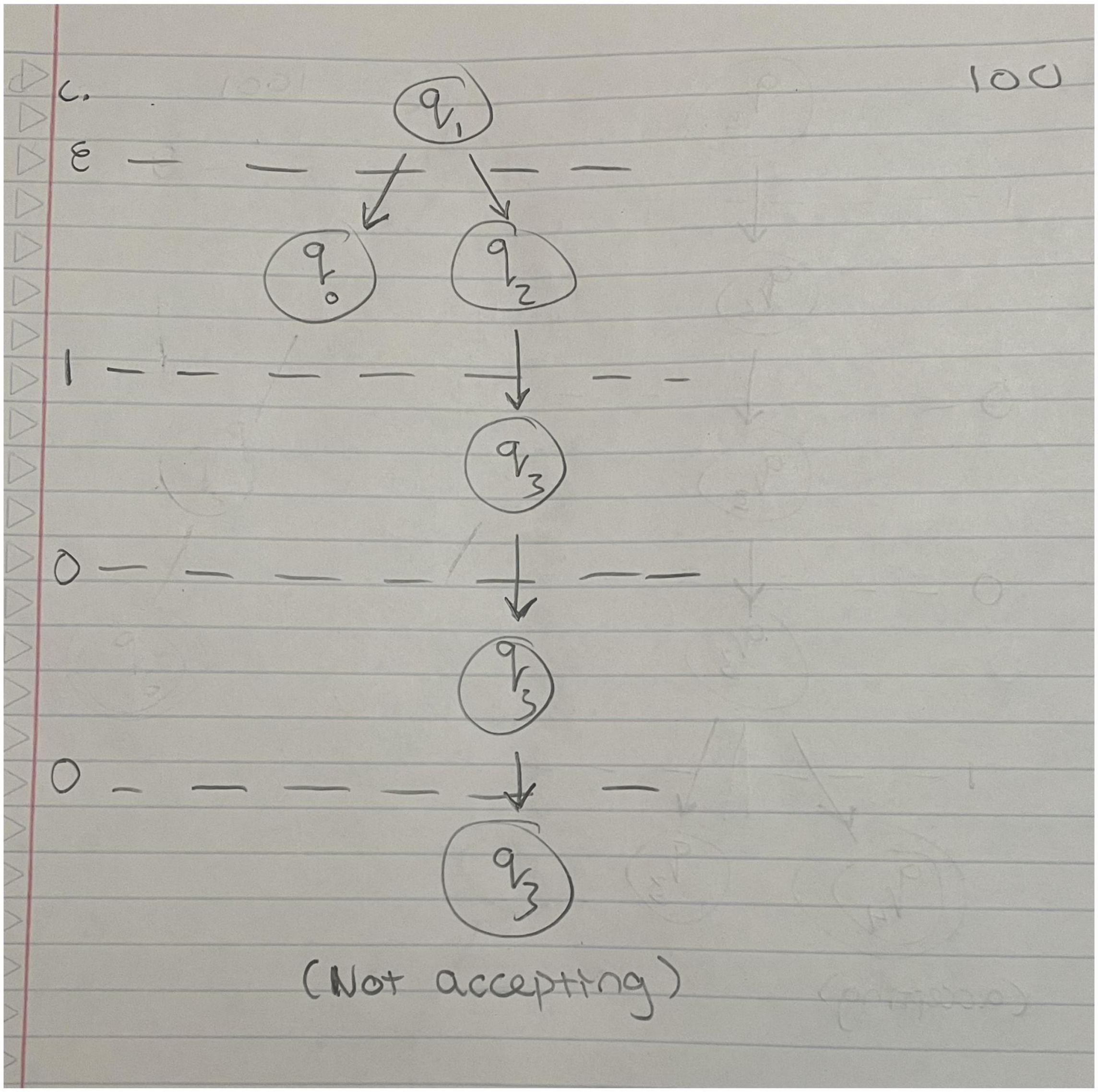


There is a resulting state that is accepting, therefore this computation is accepting. The possible sequence of states is:

$(q_2, q_3, q_3, q_3, q_4)$

Question assigned to the following page: 1.2

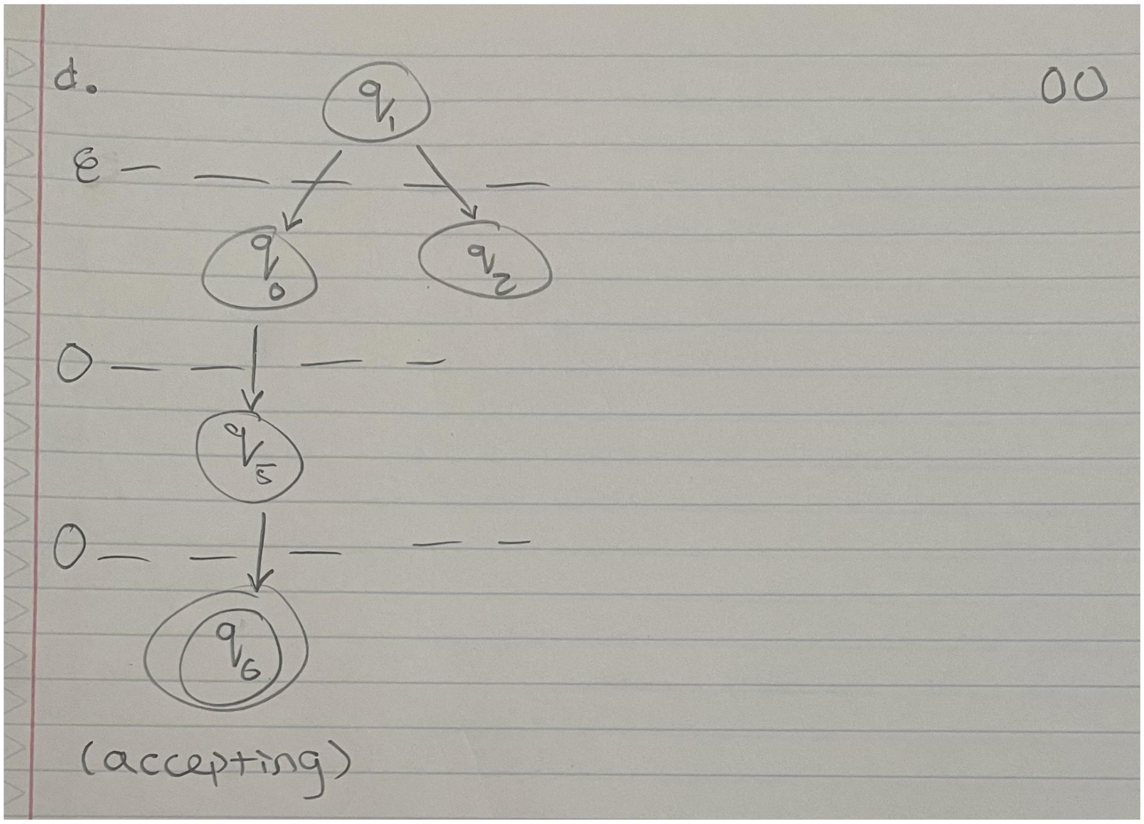
c. $\hat{\delta}(q_1, 100)$



All of the resulting states are not accepting, therefore it cannot be an accepting computation.

Question assigned to the following page: 1.2

d. $\hat{\delta}(q_1, 00)$

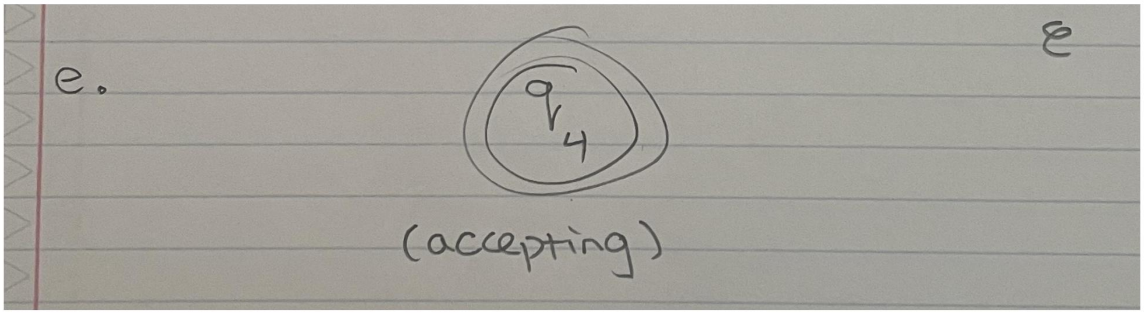


There is a resulting state that is accepting, therefore this computation is accepting. The possible sequence of states is:

(q_1, q_0, q_5, q_6)

Question assigned to the following page: 1.2

e. $\hat{\delta}(q_4, \varepsilon)$



There is a resulting state that is accepting, therefore this computation is accepting. The possible sequence of states is:

(q_4)

Questions assigned to the following page: 2.1, 2.2, and 2.3

2.1

Let A be a DFA, $A = (Q, \Sigma, \delta, q_0, F)$

Let B be an NFA, $B = (Q', \Sigma, \delta', q'_0, F')$

Then δ' is a transition function whose results are a set of states. Specifically, δ': $Q' \times \Sigma_\varepsilon \to P(Q')$, where $P(Q')$ means the output of a transition is in the form of a set. Also the Σ_ε means that an NFA can read no input. A DFA on the other hand, is not able to do any of these things. Its transition function outputs a single state, δ: $Q \times \Sigma \to Q$ and it cannot read no input, Σ . These are two possible differences between both a DFA and an NFA.

2.2

Two machines are equivalent when they are able to recognize the same language or the same set of strings.

2.3

The procedure to turn a DFA into an equivalent NFA is simply to put its transition function result into a set. Also technically all DFA's are NFA's so you don't even have to create a procedure if you don't want to.

Have: DFA $D = (Q', \Sigma, \delta', q0', F')$

Want: NFA $N = (Q, \Sigma, \delta, q0, F)$

1. $Q = P(Q) = 2^{Q'}$
2. For R in P(Q) and a in Σ,

$\delta = \{\delta\}$

3. q0=q0'
4. F=F'

Question assigned to the following page: 3

3.

<table>
<tr><td>steps</td><td>statement</td><td>justification</td></tr>
<tr><td>1</td><td>An operation is closed if it is applied to a set and the results are still part of that set.</td><td>definition of a closed operation</td></tr>
<tr><td>2</td><td>Let B and C be regular languages</td><td>given</td></tr>
<tr><td>3</td><td>x is not in B or x is not in C.
$(x \notin B) \cup (x \notin C)$</td><td>given</td></tr>
<tr><td>4</td><td>union is closed for regular languages</td><td>theorem; proven in class</td></tr>
<tr><td>5</td><td>T means x is not in language and F means it is in the language.
<table>
<tr><td>$(x \notin B)$</td><td>$(x \notin C)$</td><td>$(x \notin B) \cup (x \notin C)$</td></tr>
<tr><td>T</td><td>T</td><td>T (not possible in this case because x must be derived from either language B or C.)</td></tr>
<tr><td>F</td><td>T</td><td>T</td></tr>
<tr><td>T</td><td>F</td><td>T</td></tr>
<tr><td>F</td><td>F</td><td>F</td></tr>
</table>
</td><td>truth table</td></tr>
<tr><td>6</td><td>x cannot be an element that is in both B and C.
x cannot be an element that is in neither B or C.
x must be an element from either B or C.</td><td>step 5</td></tr>
<tr><td>7</td><td>The resulting set from OP is a set with elements that are either from B or C, but not both or neither. This means that these elements are still part of both sets, which are regular languages.</td><td>step 5 and 6</td></tr>
<tr><td>8</td><td>OP is closed for regular languages</td><td>step 4, 5, 6 and 7</td></tr>
</table>

Question assigned to the following page: 4

README
time spent: 3 hours
other students: none
books/websites: class slide