CSCE A351: Automata
26 September 2020

HW #1

- Code for stubbed methods was written and organized identical to given concatenation method
- Breakdown is given below showing proof equations side-by-side with implemented code
- Testing was done with various examples shown on page 3
- Automata drawings are found on page 4. They were created using the .get_delta_as_dictionary() function after being run through various methods.

Code Breakdown

Each method utilizing two DFAs/NFAs will check that their alphabets are identical and combine them if necessary. Additionally, methods utilizing two DFAs/NFAs will use tuples to specify which one a numbered state is referring to.

Union

<table>
<tr><td>NFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
NFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$
Create local working variables.</td><td><pre>(QA, SigmaA, deltaA, q0A, FA) = self.__convert_to_nfa()
(QB, SigmaB, deltaB, q0B, FB) = other.__convert_to_nfa()
Sigma = SigmaA
# Note: SigmaA == SigmaB == Sigma</pre></td></tr>
<tr><td>$Q = \{q_0\} \cup Q_1 \cup Q_2$
$q_0 = q' = \{q_1, q_2\}$
Iterate over all states in both NFAs and add to new Q. Add new start state, which is a set of start states from both NFAs</td><td><pre>q0 = frozenset({(1, q0A), (2, q0B)})
Q.add(q0)
for q in QA:
    Q.add((1, q))
for q in QB:
    Q.add((2, q))</pre></td></tr>
<tr><td>$F = F_1 \cup F_2$
Iterate over all states in both NFA's final states and add to new F.</td><td><pre>for q in FA:
    F.add((1, q))
for q in FB:
    F.add((2, q))</pre></td></tr>
<tr><td>$\delta(r, a) = \delta_1(r, a), \text{ if } r \in Q_1$
Iterate over all states in M for all letters in Sigma and set the new next state to all possible next states given by δ_1.</td><td><pre>for q in QA:
    for a in Sigma:
      delta[((1, q), a)] = frozenset({(1, r) for r in deltaA[(q, a)]})</pre></td></tr>
<tr><td>$\delta(r, a) = \delta_2(r, a), \text{ if } r \in Q_2$
Iterate over all states in N for all letters in Sigma and set the new next state to all possible next states given by δ_2.</td><td><pre>for q in QB:
    for a in Sigma:
      delta[((2, q), a)] = frozenset({(2, r) for r in deltaB[(q, a)]})</pre></td></tr>
<tr><td>$\delta(r, a) = \{q_1, q_2\}, \text{ if } r = q_0 \text{ and } a = \varepsilon$
Set the next state for our start state to the set of possible start states for the empty string.</td><td><pre>delta[(q0, '')] = q0
Note: We already set q0 = {(1, q0A), (2, q0B)}, so we can just set
the delta to q0</pre></td></tr>
<tr><td>$\delta(r, a) = \emptyset, \text{ if } r = q_0 \text{ and } a \neq \varepsilon$
Iterate over all letters in Sigma and set the next state of the start state to the empty set.</td><td><pre>for a in Sigma:
    delta[(q0, a)] = frozenset({})</pre></td></tr>
<tr><td>Return our new NFA.</td><td><pre>return DFA("NFA", Q, Sigma, delta, q0, F)</pre></td></tr>
</table>

CSCE A351: Automata
26 September 2020

Star

NFA $M = (Q_1, \Sigma, \delta_1, q_1, F_1)$ Create local working variables.	`(QA, SigmaA, deltaA, q0A, FA) = self.__convert_to_nfa()` `Sigma = SigmaA`

$Q = \{q_0\} \cup Q_1$
$q_0 = q' = \{q_1\}$
Iterate over all states in *M* and add to new *Q*. Add new start state, which is a set of start states from *M*.
	`q0 = frozenset({q0A})`
	`Q.add(q0)`
	`for q in QA:`
	`    Q.add(q)`

$F = \{q_0\} \cup F_1$
Iterate over all states in *M*'s final states and add to new *F*. Add new start state.
	`F.add(q0)`
	`for q in FA:`
	`    F.add(q)`

$\delta(r,a) = \delta_1(r,a)$, if $r \in Q_1$ and $r \notin F_1$
$\delta(r,a) = \delta_1(r,a)$, if $r \in F_1$ and $a \neq \varepsilon$
Iterate over all states in *M* for all letters in Sigma and set the new next state to all possible next states given by δ_1.
	`for q in QA:`
	`    for a in SigmaA:`
	`        delta[(q, a)] = frozenset({r for r in deltaA[(q, a)]})`

$\delta(r,a) = \delta_1(r,a)$, if $r \in F_1$ and $a = \varepsilon$ Iterate over all final states in *N* for the empty string and set the new next state to the start state.	`for q in FA:` `    delta[(q, '')] = q0`
$\delta(r,a) = \{q_1\}$, if $r = q_0$ and $a = \varepsilon$ Set the next state for our start state to the set of possible start states for the empty string.	`delta[(q0, '')] = q0` `# Note: We already set q0 = {q0A}, so we can just set the delta to q0`
$\delta(r,a) = \emptyset$, if $r = q_0$ and $a \neq \varepsilon$ Iterate over all letters in Sigma and set the next state of the start state to the empty set.	`for a in Sigma:` `    delta[(q0, a)] = frozenset({})`
Return our new NFA.	`return DFA("NFA", Q, Sigma, delta, q0, F)`

Complement

NFA $M = (Q_1, \Sigma, \delta_1, q_1, F_1)$ Create local working variables. Our new NFA will be identical to *M* with inverted final states.	`(QA, SigmaA, deltaA, q0A, FA) = self.__convert_to_nfa()` `(Q, Sigma, delta, q0, F) = (QA, SigmaA, deltaA, q0A, set())`

$F = \overline{F}$
Iterate over all states in *M* and add all states not in F_1 to *F*.
	`for q in QA:`
	`    if q not in FA:`
	`        F.add(q)`

Return our new NFA.	`return DFA("NFA", Q, Sigma, delta, q0, F)`

Intersection

Use DeMorgan's Law $M \cap N = \overline{(\overline{M} \cup \overline{N})}$ First get the complements of our NFAs.	`M = self.compliment()` `N = other.compliment()`
Then get the complement of the union.	`(Q, Sigma, delta, q0, F) = M.union(N).complement()._DFA__convert_to_nfa()`
Return our new NFA.	`return DFA("NFA", Q, Sigma, delta, q0, F)`

CSCE A351: Automata
26 September 2020

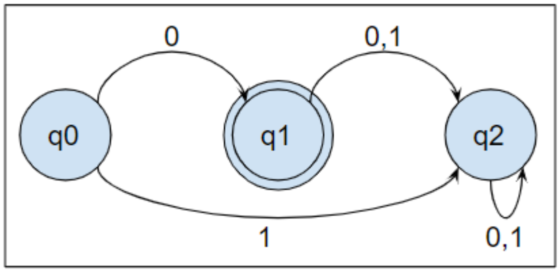
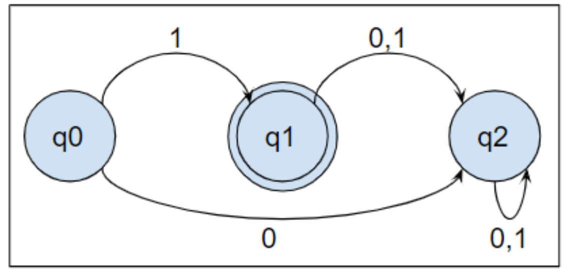
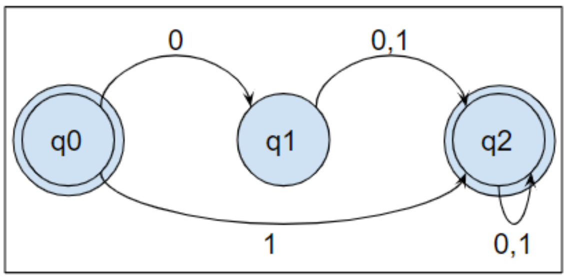
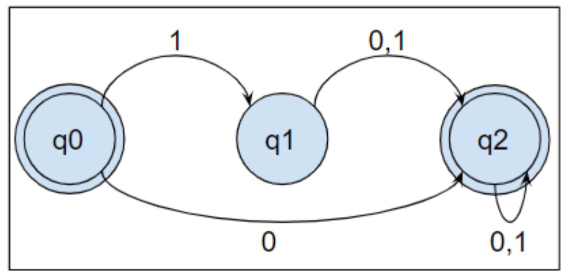
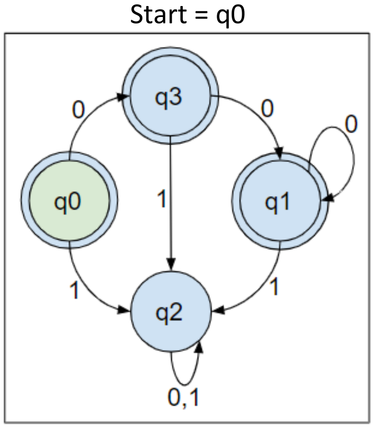
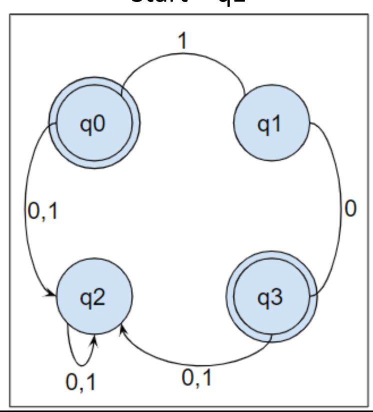
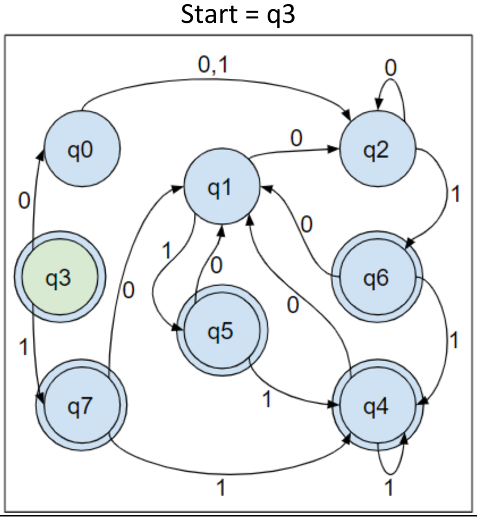
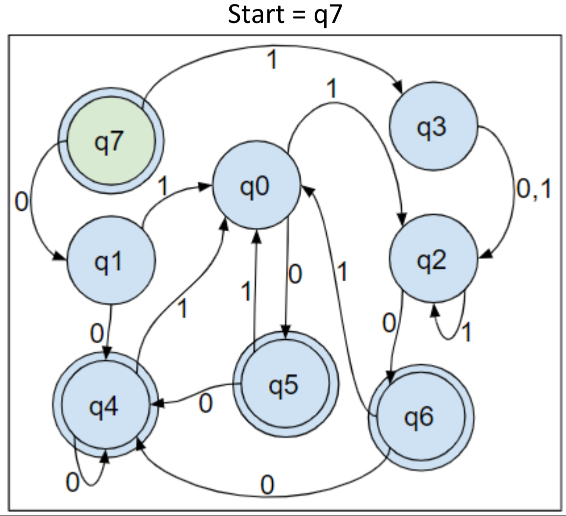
Testing

<table>
<tr><th>DFAs</th><th>.recognize()</th><th>M</th><th>N</th><th>Mc</th><th>Nc</th><th>MuN</th><th>MiN</th><th>Ms</th><th>Ns</th><th>MN</th><th>NM</th></tr>
<tr><td rowspan="18">M = DFA(‘hola|world’)
N = DFA(‘hello|world’)
Mc = M.complement()
Nc = N.complement()
MuN = M.union(N)
MiN = M.intersection(N)
Ms = M.star()
Ns = N.star()
MN = M.concat(N)
NM = N.concat(M)</td><td>‘hola’</td><td>T</td><td>F</td><td>F</td><td>F</td><td>T</td><td>F</td><td>T</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘hello’</td><td>F</td><td>T</td><td>F</td><td>F</td><td>T</td><td>F</td><td>F</td><td>T</td><td>F</td><td>F</td></tr>
<tr><td>‘world’</td><td>T</td><td>T</td><td>F</td><td>F</td><td>T</td><td>T</td><td>T</td><td>T</td><td>F</td><td>F</td></tr>
<tr><td>‘hol’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘hel’</td><td>F</td><td>F</td><td>F</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘wor’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘holahola’</td><td>F</td><td>F</td><td>T</td><td>F</td><td>F</td><td>F</td><td>T</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘hellohello’</td><td>F</td><td>F</td><td>F</td><td>T</td><td>F</td><td>F</td><td>F</td><td>T</td><td>F</td><td>F</td></tr>
<tr><td>‘worldworld’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>T</td><td>T</td><td>T</td><td>T</td></tr>
<tr><td>‘holahol’</td><td>F</td><td>F</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘hellohel’</td><td>F</td><td>F</td><td>F</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘worldwo’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘helloworld’</td><td>F</td><td>F</td><td>F</td><td>T</td><td>F</td><td>F</td><td>F</td><td>T</td><td>F</td><td>T</td></tr>
<tr><td>‘holaworld’</td><td>F</td><td>F</td><td>T</td><td>F</td><td>F</td><td>F</td><td>T</td><td>F</td><td>T</td><td>F</td></tr>
<tr><td>‘hhh’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘www’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘xyz’</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td></tr>
</table>

<table>
<tr><th>DFAs</th><th>.recognize()</th><th>M</th><th>N</th><th>Mc</th><th>Nc</th><th>MuN</th><th>MiN</th><th>MMs</th><th>NNs</th><th>McNs</th><th>NcMs</th></tr>
<tr><td rowspan="14">M = DFA(‘0’, {‘0’, ‘1’})
N = DFA(‘1’, {‘0’, ‘1’})
Mc = M.complement()
Nc = N.complement()
MuN = M.union(N)
MiN = M.intersection(N)
MMs = M.concat(M).star()
NNs = N.concat(N).star()
McNs =
M.complement().concat(N).star()
NcMs =
N.complement().concat(M).star()</td><td>‘0’</td><td>T</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘1’</td><td>F</td><td>T</td><td>T</td><td>F</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘01’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘10’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘00’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>T</td><td>F</td><td>F</td><td>T</td></tr>
<tr><td>‘11’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td></tr>
<tr><td>‘000’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>T</td></tr>
<tr><td>‘111’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>T</td><td>F</td></tr>
<tr><td>‘0000’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>T</td><td>F</td><td>F</td><td>T</td></tr>
<tr><td>‘1111’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td></tr>
<tr><td>‘001001’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>T</td><td>F</td></tr>
<tr><td>‘110110’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>T</td></tr>
<tr><td>‘xyz’</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td><td>F</td></tr>
<tr><td>‘’</td><td>F</td><td>F</td><td>T</td><td>T</td><td>F</td><td>F</td><td>T</td><td>T</td><td>T</td><td>T</td></tr>
</table>

CSCE A351: Automata
26 September 2020

Diagrams

<table>
<tr>
<td>M = DFA('0', {'0', '1'})
Start = q0

</td>
<td>N = DFA('1', {'0', '1'})
Start = q0

</td>
</tr>
<tr>
<td>M.complement()
Start = q0

</td>
<td>N.complement()
Start = q0

</td>
</tr>
<tr>
<td>M.star()
Start = q0

</td>
<td>M.union(N)
Start = q1

</td>
</tr>
<tr>
<td>M.complement().concat(N).star()
Start = q3

</td>
<td>M.complement().union(N.complement().concat(M).star())
Start = q7

</td>
</tr>
</table>