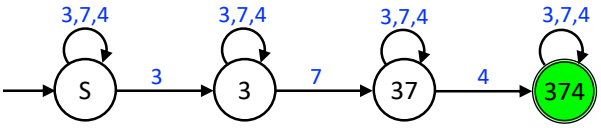
7 (100 PTS.) Construct NFAs

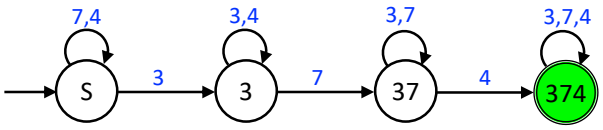
For each of the following languages over $\Sigma = \{3, 7, 4\}$, draw an NFA that accepts them. Your NFA should have a small number of states (at most say 14 states). Provide a brief explanation for your solution.

7.A. (20 PTS.) $\Sigma^* 3 \Sigma^* 7 \Sigma^* 4 \Sigma^*$

Solution:

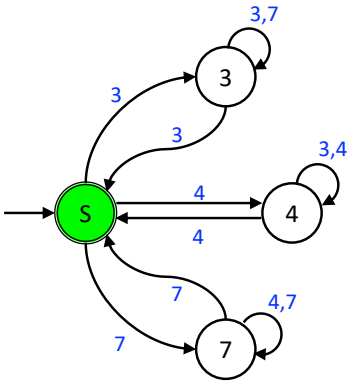


We must ensure that the subsequence 374 appears at least once. A correct answe could also be the below NFA with turns out to be a DFA as well



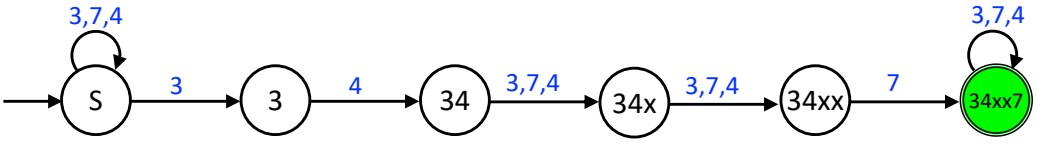
7.B. (20 PTS.) $\left(3(3+7)^*3 + 4(3+4)^*4 + 7(4+7)^*7\right)^*$

Solution:



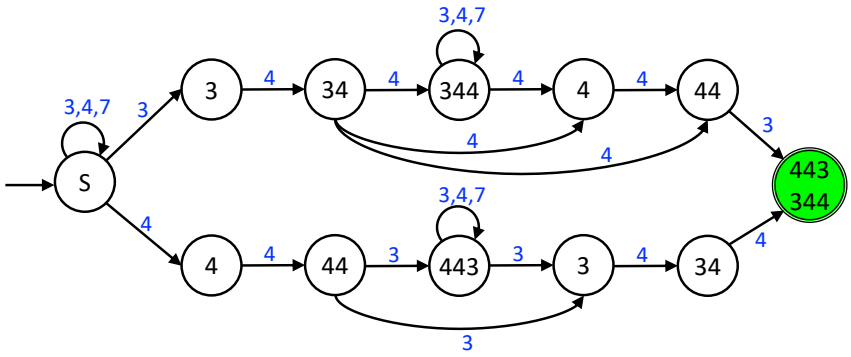
7.C. (20 PTS.) All strings in Σ^* that have a substring in $34(3+4+7)^2 7$.

Solution:



7.D. (20 PTS.) All strings in Σ^* that contain the substrings 344 and 443.

Solution:

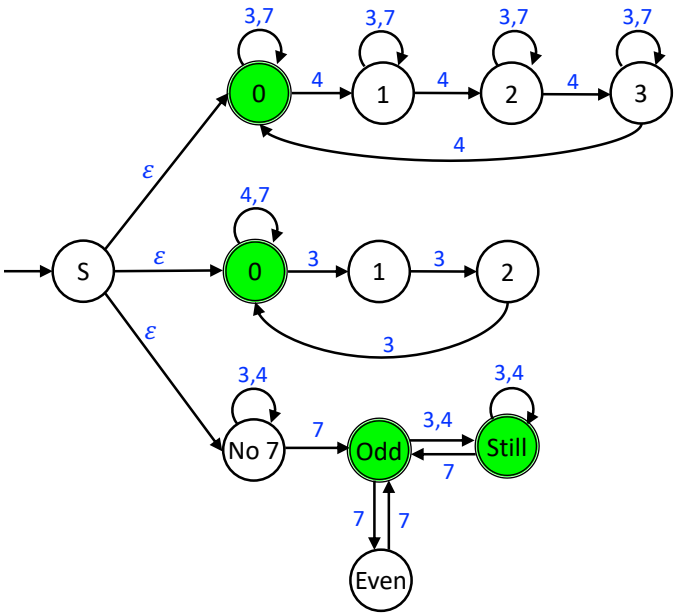


The top branch simulates 344 appearing before 443. It also simulates th substrings appearing in 34443 and 3443. The bottom branch simulates 443 appearing before 344. It also simulates th substrings appearing in 44344.

7.E. (20 PTS.) All strings in Σ^* that satisfy at least one of the following:

- The number of times 4 appears is divisible by 4.
- Every non-empty maximal substring of consecutive 7s is odd.
- Every non-empty maximal substring of consecutive 3s is divisible by 3.

Solution:



Each branch simulates one of the rules. The first ensures that the number of 4s is divisible by 4. The second ensures that the length any consecutive appearance of 3 is divisible by 3. The third ensures that any consecutive appearance of 7 is odd.

8 (100 PTS.) DFAs to NFAs

Given a DFA $M = (\Sigma, Q, \delta, s, A)$ that accepts L, construct an NFA N that accepts the following languages. You can assume $\Sigma = \{0, 1\}$ in **8.A.** and **8.B.**. Provide a brief explanation for your solution.

8.A. (25 PTS.) $\mathit{RemoveOnes}(L) := \left\{0^{\#_0(w)} \;\middle|\; w \in L\right\}$; i.e., removes all 1s from the strings.

Solution:

We construct an NFA $N = (\Sigma, Q', \delta', s', A')$ that accepts $\mathit{RemoveOnes}(L)$ as follows.

$$\begin{aligned} Q' &:= Q \\ s' &:= s \\ A' &:= A \\ \forall q \in Q,& \\ \delta'(q, 0) &= \{\delta(q, 0)\} \\ \delta'(q, 1) &= \emptyset \\ \delta'(q, \epsilon) &= \{\delta(q, 1)\} \end{aligned}$$

N is obtained from M by replacing every transition on 1 with an $\epsilon-$transition. N will simulate M and skip over the 1s. Hence, N will only accept strings of the form 0^n and will only reach an accept state after n 0 transitions if $n = \#_0(w)$ where $w \in L$.

8.B. (25 PTS.) $\mathit{RemoveOnes}^{-1}(L) := \left\{w \in \Sigma^* \;\middle|\; 0^{\#_0(w)} \in L\right\}$; i.e., puts back the 1s.

Solution:

We construct an NFA $N = (\Sigma, Q', \delta', s', A')$ that accepts $\mathit{RemoveOnes}^{-1}(L)$ as follows.

$$\begin{aligned} Q' &:= Q \\ s' &:= s \\ A' &:= A \\ \forall q \in Q,& \\ \delta'(q, 0) &= \{\delta(q, 0)\} \\ \delta'(q, 1) &= \{q\} \end{aligned}$$

N is obtained from M by a self transition on 1 for every state. N will simulate M and add as many 1s between 0s in the string. Hence, N will only reach an accept state after $\#_0(w)$ 0 transitions where $w \in L$.

8.C. (25 PTS.) *Add-k-Ones*(L) := inserts k 1s into the string. For example, *Add-3-Ones*(L) := $\{x1y1z1w \mid xyzw \in L\}$.

Solution:

We construct an NFA $N = (\Sigma, Q', \delta', s', A')$ that accepts *Add-k-Ones*(L) as follows:

$$\begin{aligned} Q' &:= Q \times \{0, 1, \cdots, k\} \\ s' &:= (s, 0) \\ A' &:= \big\{(q, k) \;\big|\; q \in A\big\} \end{aligned}$$

$$\text{For } 0 \le i < k \quad \delta'\big((q,i),a\big) = \begin{cases} \big\{\big(\delta(q,a),i\big),\ (q,i+1)\big\} & \text{if } a = 1 \\ \big\{\big(\delta(q,a),i\big)\big\} & \text{otherwise} \end{cases}$$

$$\delta'\big((q,k),a\big) = \big\{\big(\delta(q,a),k\big)\big\}$$

This is the same as the *insert*1(L) from Lab 3. However, instead of inserting a single 1, we have to insert k of them. We replicate M $k+1$ times where the ith version corresponds to the fact that we have inserted i 1s so far. N then simulates M, but inserts k 1s into M's input string at a nondeterministically chosen locations. The state (q, i) means (the simulation of) M is in state q and N has inserted i 1s so far.

8.D. (25 PTS.) *Substrings*(L) := $\{y \mid xyz \in L$ for some $x, y, z \in \Sigma^*\}$; i.e., the language of all substrings of strings in L. For example, if $L = \{ABC\}$, *Substrings*$(L) = \{\epsilon, A, B, C, AB, BC, ABC\}$.

Solution:

We construct an NFA $N = (\Sigma, Q', \delta', s', A')$ that accepts *Substrings*(L) as follows:

$$\begin{aligned} Q' &:= Q \cup \{s', t'\} \\ s' &:= s' \\ A' &:= \{t'\} \end{aligned}$$

$$\begin{aligned} \delta'(q,a) &= \{\delta(q,a)\} \quad \forall q \in Q, a \in \Sigma \\ \delta'(s',\epsilon) &= S \\ \delta'(q,\epsilon) &= \{t'\} \quad \forall q \in S \end{aligned}$$

where, $S = \{q \in Q \mid \exists x, y \in \Sigma^*$ such that $\delta^*(s,x) = q$ and $\delta^*(q,y) \in A\}$.

S is the set of states in Q that are reachable from the start state s via some prefix x and can reach an accept state via some suffix y of some string $xy \in L$. We add two new states: a start state s' and a terminate accept state t'. We then add a $\epsilon-$transitions from s' to all states in S and from all states in S to t'.

N nondeterministically picks a location to start the substring, simulates M on a substring, and then nondeterministically picks a location to end the substring.

9 (100 PTS.) Reg. Exp. to NFA to DFA

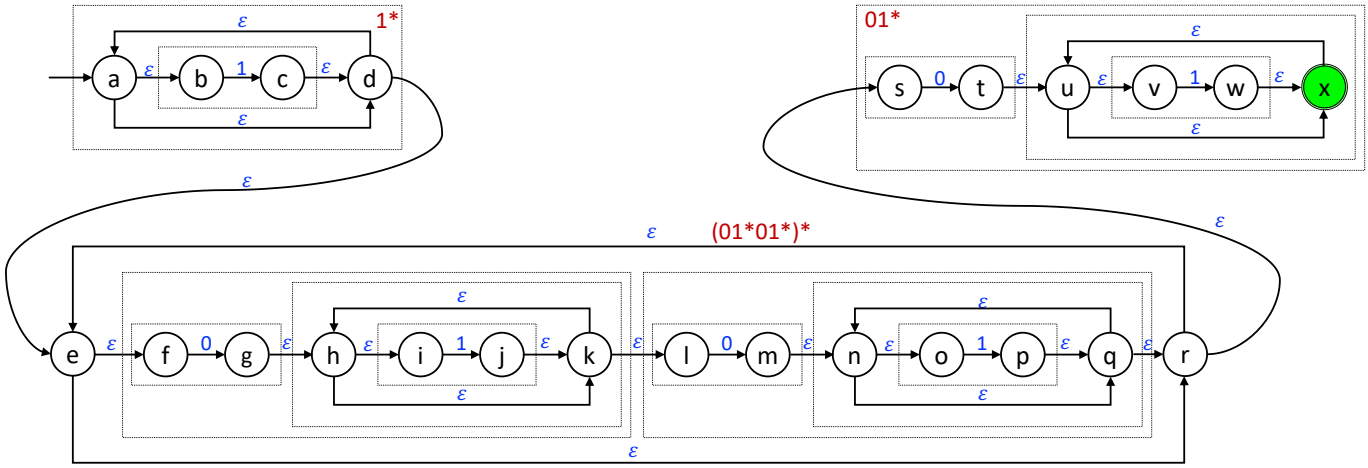
For each of the following regular expressions:

1. Construct an NFA corresponding to the regular expression using Thompson's algorithm.
2. Use the incremental subset construction to convert the NFA to a DFA
3. Describe in natural english text the language defined by the regular expression.
4. Create another DFA with at most say 4 states to recognize the language.

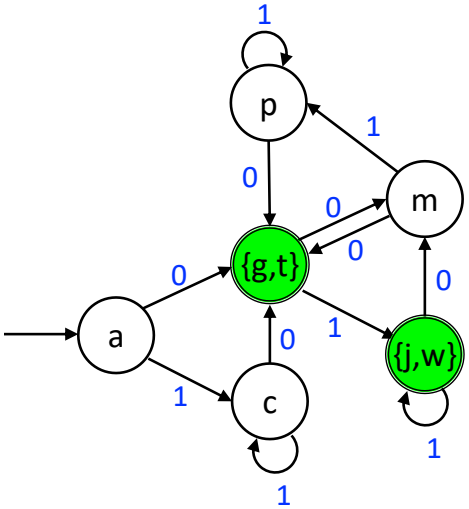
9.A. (50 PTS.) $1^*(01^*01^*)^*01^*$

Solution:

1. NFA constructed using Thompsons' Algorithm:



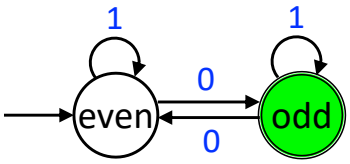
2. We now can start to construct the DFA incrementally using subset construtcion. The below table can help you construct the DFA but is not necessary to get full points on the problem.



State (q)	$\epsilon - reach(q)$	Is $q \in A$?	$\delta(q,0)$	$\delta(q,1)$
a	a, b, d, e, f, r, s	no	{g,t}	c
c	a, b, d, e, f, r, s	no	{g,t}	c
{g,t}	h, I, k, l, u, v, x	yes	m	{j,w}
m	n, o, q, r, e, f, s	no	{g,t}	p
{j,w}	k, h, i, l, x, u, v	yes	m	{j,w}
p	q, r, s, n, o, e, f	no	{g,t}	p

3. The language represents all strings with an odd number of 0s.

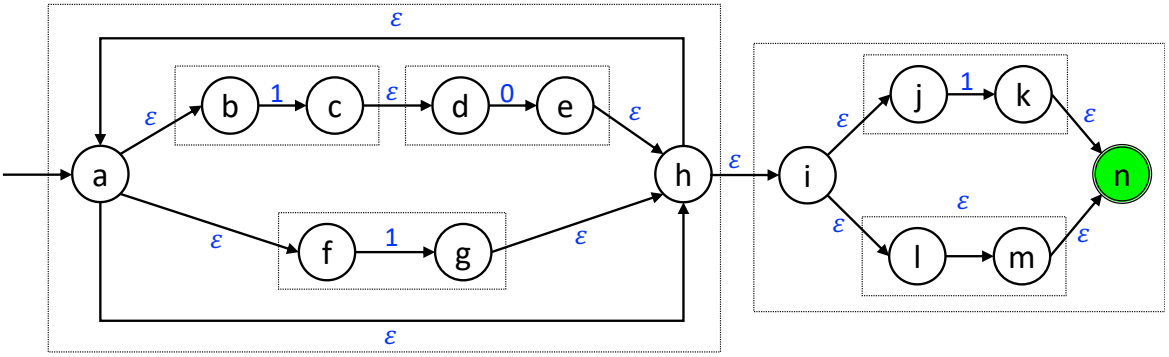
4. A simpler DFA:



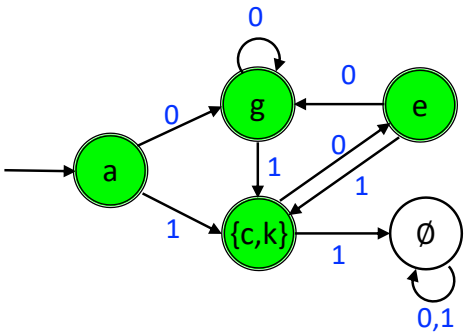
9.B. (50 PTS.) $(10+0)^*(1+\epsilon)$

Solution:

1. NFA constructed using Thompsons' Algorithm:



2. We now can start to construct the DFA incrementally using subset construtcion. The below table can help you construct the DFA but is not necessary to get full points on the problem.



State (q)	$\epsilon - reach(q)$	Is $q \in A$?	$\delta(q,0)$	$\delta(q,1)$
a	a, b, f, h, i, j, l, m, n	yes	g	{c,k}
g	h, a, b, f, i, j, l, m, n	yes	g	{c,k}
{c,k}	d, n	yes	e	$\emptyset$
e	h, a, b, f, i, j, l, m, n	yes	g	{c,k}
$\emptyset$	$\emptyset$	no	$\emptyset$	$\emptyset$

3. The language represents all strings that do not contain the substring 11s.
4. A simpler DFA: