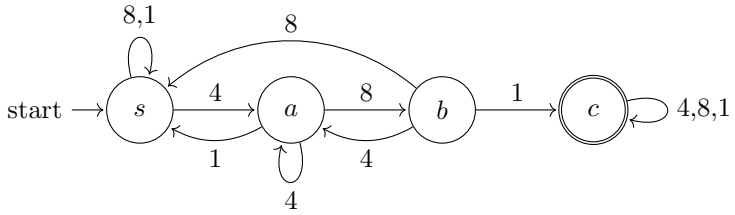
CSCI 3434 Theory of Computation Homework 1

Luke Meszar (Worked with Jeff Lipnick)

September 13, 2017

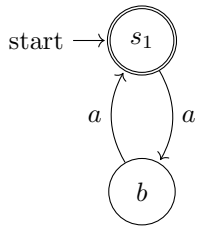
HW 1.1 Design deterministic finite automata for each of the following sets:

(a) the set of strings in $\{4, 8, 1\}^*$ containing the substring 481;
Consider the following DFA:

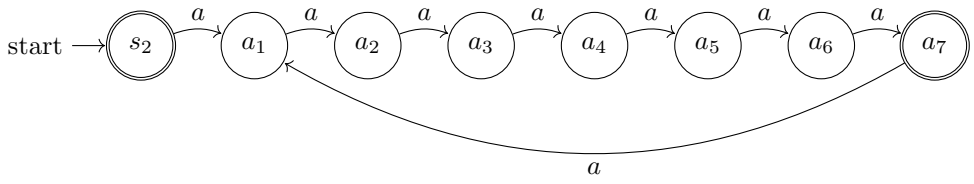


At the first state, the DFA can only from state s if there is a four in the string. Otherwise, it continues looping at state s until it sees a 4 meaning it can begin looking for 481. Then, in state a, it can only continue onwards to state b if it sees an 8. Otherwise, if it sees a 1, then it has to go back to the state s; but if it sees a four, it should stay at state a so it can check for an 8 again. Then, at state b, it continues onto state c if it sees a 1, because it has found the substring 481. If it sees an 8, then it has to go back to s so it can start the search over. If it sees a 4, then it only has to go back to a since state a is the state representing, "I have seen a 4" already. Once the DFA reaches state c, then it loops there till the end of the string since it has already found the substring 481 and it doesn't care what comes afterwards.

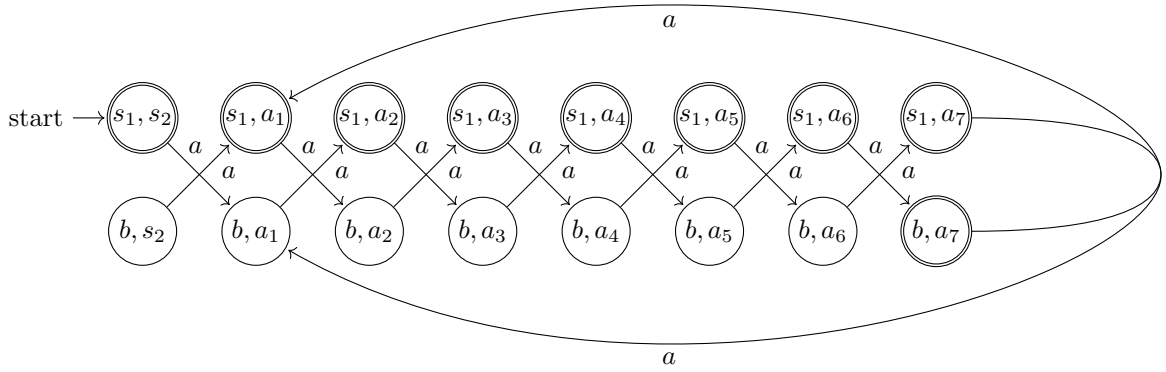
(b) the set of strings in $\{a\}^*$ whose length is divisible by either 2 or 7;
Consider the DFA M_1 that accepts strings in $\{a\}^*$ with length divisible by 2:



M_1 works by counting the number of a's mod 2. It accepts every second a so if the string has even length, then it will end in the accept state, s_1.
Also consider the DFA M_2 that accepts strings in $\{a\}^*$ with length divisible by 7



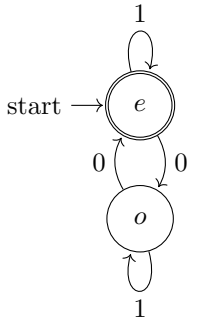
M_2 works by counting the number of a's mod 7. From the start, s_2, it moves forward 7 states and then accepts. It is important that from a_7, the DFA transitions to a_1. If it instead went back to s_2, then on every subsequent loop through M_2, it would be string whose length were of the form $7 + 8k$ for $k \in \mathbf{N}$. By going from a_7 to a_1, it only counts 7 a's on successive loops through the automaton.
Then using the product construction for unions, we get:



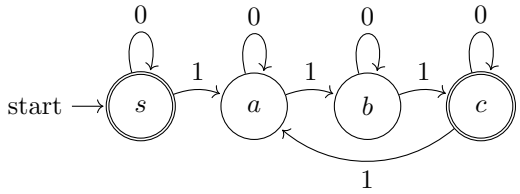
This DFA has states corresponding to the Cartesian product of the states of M_1 and M_2. The transition function is defined based on where each state in the pair went to on a given input in their original machines. The final states consist of any state in which one of the elements in the Cartesian product was a final state in either M_1 or M_2.

(c) the set of strings $x \in \{0,1\}^*$ such that $\#0(x)$ is even and $\#1(x)$ is a multiple of three.

Let M_1 be the DFA that accepts strings $x \in \{0,1\}^*$ such that $\#0(x)$ is even.

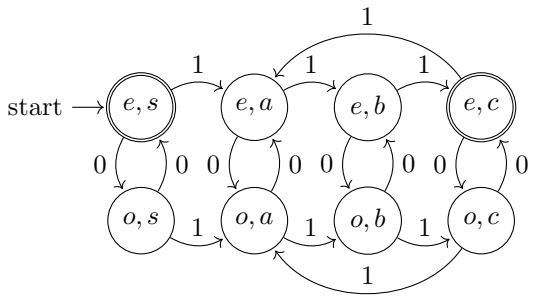


This DFA works very similarly to the first one in the previous problem. The only difference is that it doesn't let the number of 1's affect the result so it doesn't change states when it sees a 1, only when it sees a zero. Then, let M_2 be the DFA that accepts strings $x \in \{0,1\}^*$ such that $\#1(x)$ is a multiple of 3.



This DFA works very similarly to the second one in the previous problem. Again, the only difference is that it doesn't want the number of 0's to affect the result so it doesn't change states when it sees a 0, only when it sees a 1.

Then using the product construction for intersections, we get



Here, the states are the Cartesian product of the states of M_1 and M_2. The transition function is defined based on where each state in the pair went to on a given input in their original machines. The final state is the one where both elements in the ordered pair of the Cartesian product were finals states in their respective machines. In this case, that is the state (e, c) since state e was the final state in M_1 and state c was the final state in M_2.m

HW 1.2 Consider the following two deterministic finite automata.

		a	b
→	1	1	2
	2F	2	1

		a	b
→	1	2	3
	2	3	1
	3F	1	2

Use the product construction to produce deterministic automata accepting (a) the intersection and (b) the union of the two sets accepted by these automata.

For the intersection, we use the product construction. The final state is the ordered pair where both elements are final states in their respective machines. In this example, this corresponds to (2,3).

		a	b
→	(1,1)	(1,2)	(2,3)
	(1,2)	(1,3)	(2,1)
	(1,3)	(1,1)	(2,2)
	(2,1)	(2,2)	(1,3)
	(2,2)	(2,3)	(1,1)
	(2,3)F	(2,1)	(1,2)

For the union, we use the product construction. The final states are the ordered pairs where at least on of the elements in a final state in its respective machine. In this example, such states are (1,3), (2,1), (2,2), and (2,3).

		a	b
→	(1,1)	(1,2)	(2,3)
	(1,2)	(1,3)	(2,1)
	(1,3)F	(1,1)	(2,2)
	(2,1)F	(2,2)	(1,3)
	(2,2)F	(2,3)	(1,1)
	(2,3)F	(2,1)	(1,2)

HW 1.3 Let $M = (Q, \Sigma, \delta, s, F)$ be an arbitrary DFA. Prove by induction on $|y|$ that for all strings $x, y \in \Sigma^*$ and $q \in Q$,

$$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y),$$

where $\hat{\delta}$ is the extended version of δ defined on all strings described in Lecture 3.

Proof. Base Case: Let $|y| = 0$ which implies $y = \varepsilon$. Then, on the left hand side we get

$$\hat{\delta}(q, x\varepsilon) = \delta(\hat{\delta}(q, x), \varepsilon) = \hat{\delta}(q, x)$$

by the definition of $\hat{\delta}(q, xa)$ and the definition of $\delta(q, \varepsilon)$. On the right hand side we get

$$\hat{\delta}(\hat{\delta}(q, x), \varepsilon) = \hat{\delta}(q, x)$$

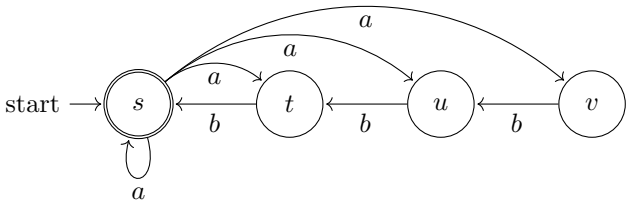
by the definition of $\hat{\delta}(q, \varepsilon)$. Since the left and right hand sides agree, the base case holds.

Induction Step: Assume $|y| = k$ for some $k \in \mathbf{N}$ and that $\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$. Now, consider ya, a string of length $k + 1$. Then we get:

$$\begin{aligned}
\hat{\delta}(q, xya) &= \delta(\hat{\delta}(q, xy), a) \\
&= \delta(\hat{\delta}(\hat{\delta}(q, x), y), a) \text{ by the induction hypothesis} \\
&= \hat{\delta}(\hat{\delta}(\hat{\delta}(q, x), y), a) \text{ since } \delta \text{ and } \hat{\delta} \text{ are equivalent for a single symbol} \\
&= \hat{\delta}(\hat{\delta}(q, x), ya).
\end{aligned}$$

Thus, we have shown by induction that $\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y)$ for all $x, y \in \Sigma^*$ and $q \in Q$. ∎

ME 3 Consider the following nondeterministic finite automaton.



(a) Give a string beginning with a that is *not* accepted.
The string *abbbb* will not be accepted. The "closest" it can get to being accepted is if it takes the path

$$s \xrightarrow{a} v \xrightarrow{b} u \xrightarrow{b} t \xrightarrow{b} s \xrightarrow{b} \varnothing.$$

Any other possible path will end in the empty set even sooner. Any string of the form ab^n for $n \geq 4$ will not be accepted.

(b) Construct an equivalent deterministic finite automaton using the subset construction. Assuming the states are names s, t, u, v from left to right, show clearly which subset of $\{s, t, u, v\}$ corresponds to each state of the deterministic automaton. Omit inaccessible states.
Using the subset construction, we get

			a	b
→	s	F	$stuv$	d
	$stuv$	F	$stuv$	stu
	stu	F	$stuv$	st
	st	F	$stuv$	s
	d		d	d

HW 2.2 The *reverse* of a string x, denoted rev x, is x written backwards. Formally,

$$\text{rev } \varepsilon \stackrel{\text{def}}{=} \varepsilon, \quad \text{rev } xa \stackrel{\text{def}}{=} a \text{ rev } x.$$

For example, rev $bbaaab$ = $baaabba$. For $A \subseteq \Sigma^*$, define

$$\text{rev } A \stackrel{\text{def}}{=} \{\text{rev } x \mid x \in A\}.$$

For example, rev $\{a, ab, aab, aaab\} = \{a, ba, baa, baaa\}$. Show that for any $A \subseteq \Sigma^*$, if A is regular, then so is rev A.

Proof. Assume that $M = (Q, \Sigma, \delta, s, F)$ is a DFA that accepts A. Now, construct the following NFA $N = (Q \cup \{q_s\}, \Sigma, \Delta, q_s, s)$. We add a state q_s so that we can treat M as having only one final state. We accomplish this by adding ε transitions between q_s and all $f \in F$. The state q_s becomes our new start state and we will s, the start state in M be the final state of N. The key construct is the way Δ is defined. It is defined to reverse the direction of all the arrows in M. This is done in the following manner:

$$\Delta(q, a) = \delta^{-1}(\{q\}, a)$$

where $\delta^{-1}(\{q\}, a)$ is the set of all states r such that $\delta(r, a) = q$. We need to make the stipulation that $\Delta(s, a) = s$ for all $a \in \Sigma$. To show that this NFA accepts rev A, we need to show that

$$\{x \in \Sigma^* \mid \hat{\Delta}(q_s, x) = s\} = \text{rev } A.$$

Denote $\{x \in \Sigma^* \mid \hat{\Delta}(q_s, x) = s\}$ as Γ. We begin by showing that if $x \in \text{rev } A$, then $\hat{\Delta}(q_s, x) = s$. Since $x \in \text{rev } A$, then rev $x \in A$. This means that there is some sequence of states in M

$$s \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \longrightarrow \cdots \longrightarrow q_n \xrightarrow{a_{n+1}} f_i$$

since rev x is accepted by M. Note that $x = a_{n+1}a_n \cdots a_2 a_1$. At the beginning, $\Delta(q_s, x)$ ε-transitions to all the final states nondeterministically, including f_i. Now, $q_n \in \hat{\Delta}(q_s, a_{n+1})$ since $\delta(q_n, a_{n+1}) = f_i$ and f_i is essentially the start state. This is the base case on induction that Δ maps backwards on this sequence of states.

Now, for some $k + 1 \in \{1, \ldots, n\}$, assume $q_k \in \hat{\Delta}(q_s, a_{n+1}a_n \cdots a_{k+1})$.

Now, we need to show that $q_{k-1} \in \hat{\Delta}(q_s, a_{n+1}a_n \cdots a_k)$. This is equivalent to showing that

$$q_{k-1} \in \bigcup_{q \in D} \Delta(q, a_k)$$

where $D = \hat{\Delta}(q_s, a_{n+1}a_n \cdots a_{k+1})$. Well, by our assumption, $q_k \in D$ so we can consider $\Delta(q_k, a_k)$. Well, $q_{k-1} \in \delta^{-1}(q_k, a_k)$ since we know $\delta(q_{k-1}, a_k) = q_k$ because rev $x \in A$. Thus, $q_{k-1} \in \hat{\Delta}(q_s, a_{n+1}a_n \cdots a_k)$. This induction process stops when we get to $k = 1$ and we have $q_1 \in \hat{\Delta}(q_s, a_{n+1}a_n \cdots a_2)$. To show that $s \in \hat{\Delta}(q_s, a_{n+1}a_n \cdots a_1)$, we just note that this is equivalent to

$$s \in \bigcup_{q \in D'} \Delta(q, a_1)$$

where $D' = \hat{\Delta}(q_s, a_{n+1}a_n \cdots a_1)$. Well, $q_1 \in D'$ so we can consider $\Delta(q_1, a_1)$. We know that $s \in \delta^{-1}(q_1, a_1)$ since we know $\delta(s, a_1) = q_1$ because rev $x \in A$. Thus, we have shown that if $x \in \text{rev } A$, then $x \in \Gamma$ so rev $A \subseteq \Gamma$.

Now, let $x \in \Gamma$. We want to show that $x \in \text{rev } A$. Since $x \in \Gamma$, there must be some path that is chosen nondeterministically of the following form:

$$s \xleftarrow{a_1} q_1 \xleftarrow{a_2} q_2 \longleftarrow \cdots \longleftarrow q_n \xleftarrow{a_{n+1}} f_i.$$

If we can show that rev $x \in A$, then we know that $x \in \text{rev } A$. To do this, we want to show that $\hat{\delta}(s, \text{rev } x) = \hat{\delta}(s, a_1 a_2 \cdots a_{n+1}) = f_i$. We begin by proving the base case of

$$\hat{\delta}(s, a_1) = q_1.$$

Well, we know that $\hat{\delta}(s, a_1) = \delta(s, a_1)$. Then, from the sequence above, we know that $s \in \Delta(q_1, a_1)$ which implies $s \in \delta^{-1}(q_1, a_1)$ which means that $\delta(s, a_1) = q_1$. Thus

$$\hat{\delta}(s, a_1) = q_1.$$

Next, for some $k \in \{1, \ldots, n\}$ assume that $\hat{\delta}(s, a_1 a_2 \cdots a_k) = q_k$. Now consider

$$\hat{\delta}(s, a_1 a_2 \cdots a_{k+1}) = \delta(\hat{\delta}(s, a_1 a_2 \cdots a_k), a_{k+1}) = \delta(q_k, a_{k+1}).$$

Then, from the sequence above, we know that $q_k \in \Delta(q_{k+1}, a_{k+1})$ which implies $q_k \in \delta^{-1}(q_{k+1}, a_{k+1})$ which means that $\delta(q_k, a_{k+1}) = q_{k+1}$. Thus

$$\hat{\delta}(s, a_1 a_2 \cdots a_{k+1}) = q_k + 1.$$

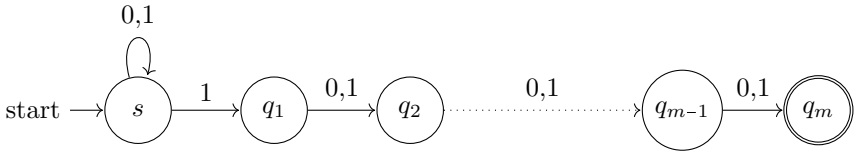
This completes the induction. The inductions stops when we reach the state q_n. Then, to show that $\delta(q_n, a_{n+1}) = f_i$, we use the same logic that we have used above about $\Delta(f_i, a_{n+1})$. Thus, $\hat{\delta}(s, \text{rev } x) = f_i$ which implies that rev $x \in A$ so $x \in \text{rev } A$. This completes the proof that if A is regular, then rev A is regular. ∎

ME 6 Prove that NFAs are exponentially more succinct than DFAS: for any m, there exists an NFA with m states such that any equivalent DFA has at least 2^{m-1} states.

Proof. Let A_{m+1} be the following language:

$$\{x \in \{0,1\}^* \mid \text{ the } m^{\text{th}} \text{ symbol from the right is a } 1\}.$$

An NFA that accepts this language will look like:



This NFA has $m+1$ states. It works by nondeterministically guessing if the m^{th} symbol from the right is a 1 and then checking if this is indeed the case. Due to the nondeterminism, the machine only needs to loop nondeterministically at the start state until it sees a 1 and then see if there are $m-1$ symbols left in the string. This machine requires $m+1$ states.

An equivalent DFA would need a state to remember the last m bits that it has seen. The accept states would be any states of the form $[1x]$ where $|x| = m-1$. We consider ε to be equivalent to 0. That means the machine begins in state $[00\cdots0]$. If the first symbol read is a 1, then it will transition to the state $[00\cdots1]$. In general, if the machine was in a state $[a_1a_2\cdots a_m]$ and it read a symbol a_i, it would transition to the state $[a_2a_3\cdots a_ma_i]$. It is easy to see that there will be 2^m states in this DFA. Now, we need to show that there is no possible DFA with fewer than 2^m states.

Assume there is a DFA with fewer than 2^m states that accepts A_{m+1}. Then there must be two different string in $(0|1)^m$ that reach equivalent states. The set $(0|1)^m$ can be defined as $\{x \in \{0,1\}^* \mid |x| = m\}$. This can be seen using the pigeonhole principle. Since there are 2^m possible strings in $(0|1)^m$, but there are less than 2^m states, then at least two of the strings must end in the same state. Denote these two different strings that end in the same state as x and y. Since $x \neq y$, then at least one symbol, say at position i has to be different. That is, $x_i \neq y_i$. Let $x = x_1x_2\cdots x_m$ and $y = y_1y_2\cdots y_m$. Since $|x| = |y| = m$, then the state x will end up in will be $[x_1x_2\cdots x_m]$ the state y will end up in will be $[y_1y_2\cdots y_m]$. Since $x \neq y$, then these two states cannot be the same. But we assumed that x and y were two of the strings that had to end up in the same state. Thus, we have reached a contradiction. Therefore, our assumption that the DFA has fewer than 2^m states must be wrong.

We have shown that there exists an NFA with $m+1$ states such that the equivalent DFA has at least 2^m states.

■