CSCI 510, Fall 2016, Homework # 1

SOLUTION SKETCHES

Due date: Tuesday, October 4, Midnight

1. Transforming a NFA with n states into a DFA may require as many as 2^n states. Give examples of NFAs with 2 and 3 states that *require* 4 and 8 states, respectively, in their equivalent DFAs. Use the alphabet $\Sigma = \{0, 1\}$. Explain why they require 2^n states and find the DFAs in question.

   2 states: The NFA in Figure 1 requires 2^n states because there exist strings that lead to each possible subset in the power set of states:

String	Leads to
ε	a
0	$\emptyset$
1	b
10	a, b

   Further, none of these states can be eliminated from the resulting DFA since, in each state, there exist extensions of the current string that disagree on whether the current string is accepted or rejected. Therefore, none of the states are equivalent. (For example, if b had been the accepting state instead of a, then the states b and a, b in the resulting DFA would be equivalent, and could be merged.)

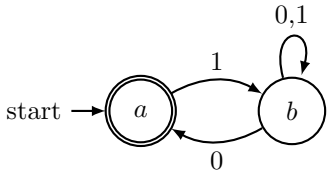

Figure 1: 2-state NFA that requires 4 states for corresponding DFA.

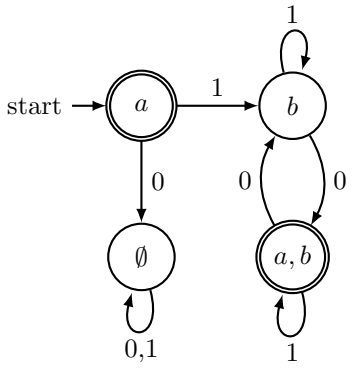

Figure 2: 4-state DFA corresponding to 2-state NFA.

3 states: The NFA is in Figure 3 and the DFA in Figure 4. The argument for this machine is identical to the argument for the previous case: there exists a unique string that leads to each of the states in the DFA, and none of the states is equivalent to any of the others because of the distribution of final states.

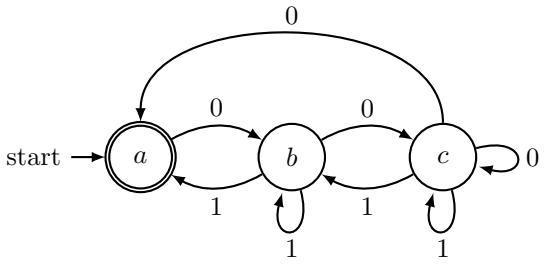

Figure 3: 3-state NFA that requires 8 states for corresponding DFA.

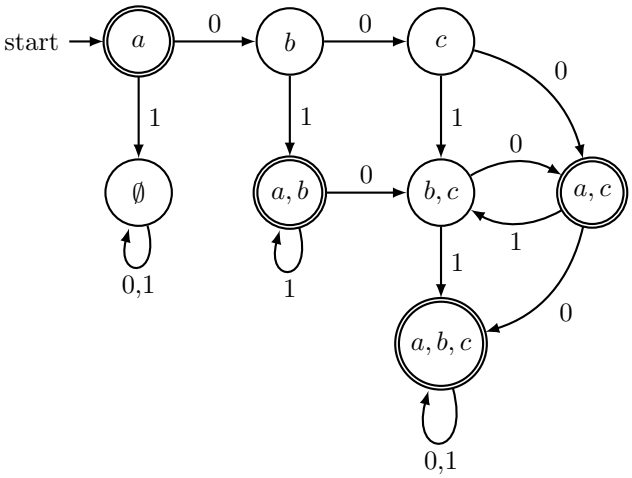

Figure 4: 8-state DFA corresponding to 3-state NFA.

2. Let

$$\Sigma = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.$$

So Σ contains all columns of 0s and 1s of height two. A string of symbols in Σ give two rows of 0s and 1s. Consider each row to be a binary number and let

$$C = \{w \in \Sigma^* | \text{the bottom row is three times the top row}\}.$$

For example, $\left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right]\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]\left[\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right]\left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right] \in C$ because the binary number 0110 is three times the binary number 0010, but $\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right] \notin C$ because the binary number 110 is not three times the binary number 001. Show that C is regular.

For this problem you may assume the truth of the theorem that says that the reverse of a regular language is regular. The reverse of a language A, written A^R, is the set of all strings $w^R = w_n \dots w_2 w_1$ such that $w = w_1 w_2 \dots w_n \in A$,

$$A^R = \{w^R |\ w^R = w_n \dots w_2 w_1 \text{ where } w = w_1 w_2 \dots w_n \in A\},$$

Solution. Let's look at the process of multiplying a number by 3 in binary. In Figure 5, the carries are written as superscripts on the second line, for reasons that will be clear later.

	1	0	0	1	1	1	$\Leftarrow n$
	$\times$				1	1	$\Leftarrow 3$
	1	0	0	1	1	1	
1^0	0^0	0^1	1^1	1^1	1^0		
1	1	0	0	1	0	1	$\Leftarrow 3n$

Figure 5: Example of multiplying by 3 in binary.

Basically we just shift the number one place to the left, and then add. The first digit must be the same and cannot result in a carry. From there on, moving from right to left (taking advantage of the theorem that says we can do this right-to-left instead of left-to-right), at each position we can compute the answer by remembering what the bit to the right was (that will be the bit shifted to the left in the second row) and whether there was a carry from the previous addition. For example, in the second column from the right, we need to remember that the shifted bit is a 1 and the carry bit is a 0. In the second line this is written 1^0. In the third column, the shifted bit was a 1 and the carry was a 1. We wrote this as 1^1.

To build our DFA, then, we will only need to remember two bits, and we need $2^2 = 4$ states for this. One extra state will serve for the start state, and one other state as a permanent reject state if the two strings cannot be matched. (If we want to accept the empty string, we can dispense with the extra start state.)

In Figure 6 we represent states as a circle with two bits in side, representing the pair (shifted bit, carry bit). For example, the node (0^1) represents a shifted bit of 0 and a carry bit of 1. We can now add in the necessary transitions by following the laws of arithmetic. Examples:

- If we are in state 0^0, and n has a 0 in the next position, then $3n$ will also have a 0, and we will return to the state 0^0, so the arc for $\left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right]$ goes from 0^0 to 0^0.
- If we are in state 0^0 and n has a 1 in the next position, then $3n$ will have a 1 in this position, and we will end up with no carry, and a 1 as the shifted bit. So the $\left[\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right]$ arc goes from 0^0 to 1^0.

Similarly, we can fill out the entire transition function for our machine with this transition table:

State	n bit	$3n$ bit	New state
0^0	0	0	0^0
0^0	0	1	$\otimes$
0^0	1	0	$\otimes$
0^0	1	1	0^0
1^0	0	0	$\otimes$
1^0	0	1	0^0
1^0	1	0	0^0
1^0	1	1	$\otimes$
0^1	0	0	$\otimes$
0^1	0	1	0^0
0^1	1	0	0^0
0^1	1	1	$\otimes$
1^1	0	0	0^0
1^1	0	1	$\otimes$
1^1	1	0	$\otimes$
1^1	1	1	0^0

The accepting state has to have a carry of zero, and further the last bit of the n string must be a zero. If it were a 1, then there would be at least one more bit in the $3n$ string. Thus, there is only one accepting state, namely 0^0. The DFA that results from all of these considerations is shown in Figure 6.

It should be obvious from construction that this machine recognizes the language C, but we will sketch a more formal proof as follows:

Theorem 1. *The language C is identical to the language accepted by the DFA in Figure 6.*

Proof. This theorem will follow immediately from Lemmas 2 and 3, below. □

Lemma 1. *If a string w with $|w| \geq 1$ is processed by the DFA, then at each point in the process if the state is not the trap state, then the state correctly represents both the previous bit of the n string, and the carry bit of a multiplicaiton of the n string by 3. Further, if the state is the trap state, then w cannot be in C.*

Proof. This follows from the construction of the machine and induction.

Base: The shortest possible strings are $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$, and $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$.

- $\left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ end up in the trap state, and neither one can be in C
- $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$ ends up in state 0^0, and $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$ ends up in state 1^0, and both of these are correct.

Step: Suppose $|w| = n$ and all strings of length less than n $w = w_n w_{n-1} \dots w_1$. Then by the inductive hypothesis the Lemma is true for the string $w_{n-1} \dots w_1$. By inspection of each case for w_n in the transition function, above, the Lemma will also hold true for w.

□

Lemma 2. *If $w \in C$ then w is accepted by the DFA.*

Proof. If $w \in C$ then by Lemma 1 it will not end up in the trap state. Further, if $w \in C$ then the carry bit will be 0, and the last bit of the n bitstream will be 0, then by Lemma 1 it will end up in state 0^0, which is an accept state. □

Lemma 3. *If w is accepted by the DFA, then $w \in C$.*

Proof. If w is accepted by the DFA, then it will finish in state 0^0. By Lemma 1, that implies that the carry bit in a multiplication by 3 is 0, and also that the last bit of the n string is also 0. This implies that the $3n$ string is the bit representation of the number $3n$. □

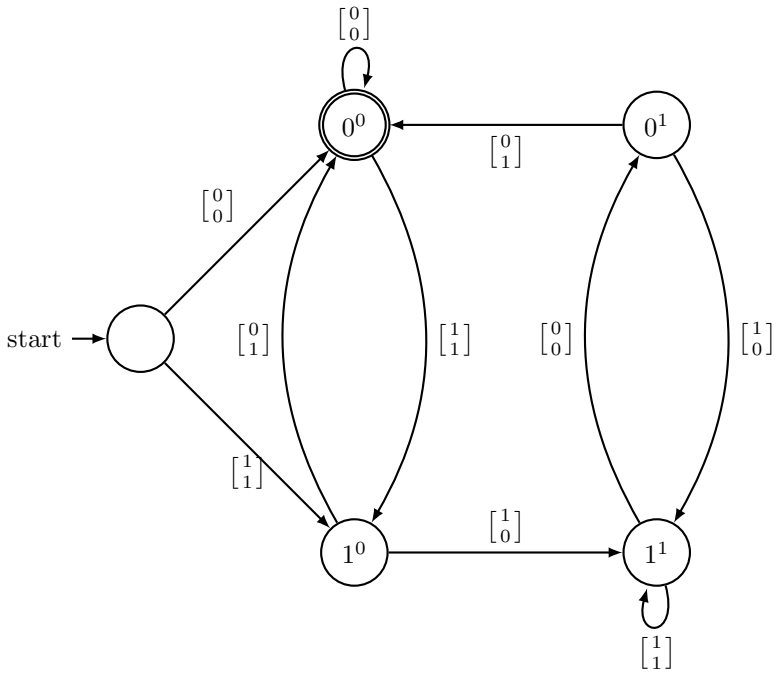

Figure 6: DFA for the $n/3n$ problem. Transitions that go to the trap state are omitted for clarity. Note: states 0^1 and 1^0 can be collapsed into one state. We only really need to remember whether 0, 1, or 2 needs to be carried into the sum.

3. Let Σ be as in the previous problem, and let

$$E = \{w \in \Sigma^* | \text{the bottom row of } w \text{ is the reverse of the top row of } w\}.$$

Show that E is not regular.