Formal Languages and Computability 2

Ragnar Björn Ingvarsson, rbi3

10. september 2024

1 Which of the statements are true for every regular language A?

a) $A = A^*$

For this case we see that the empty string, ϵ is not neccessarily in A, however it is guaranteed to be a possibility within A^*.

Therefore these are not always equal.

b) $A^* = A \circ A^*$

We see here that it is possible to say that $A = \{0, 1\}$ for example, and then the left side can become just the empty string, but the right side must contain a 0 or a 1 before it can give the empty string, rendering it useless.

Therefore these are not always equal.

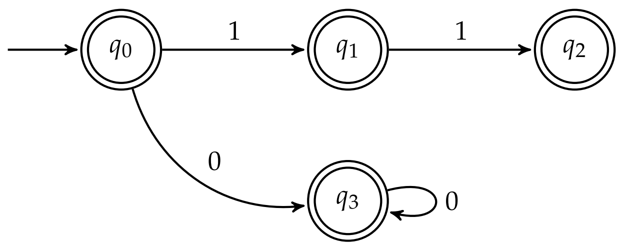
c) $A^* \circ A = A \circ A^*$

Here we notice that the empty string, ϵ, has no effect since both sides contain at least one character. Because of this, there is no possible string that one side can construct that the other cannot.

So this statement is *true*.

2 Let $\Sigma = \{0, 1\}$ and let A denote a language of strings that consist soley of any number of zeroes, or at most two ones, including the empty string.

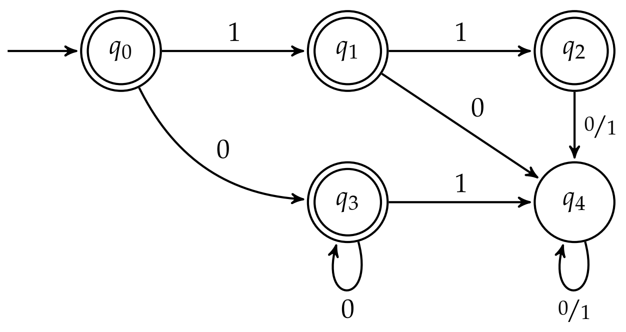
a) Draw a state diagram of an NFA that recognizes A.



b) How many states do you need with a DFA to do the same task?

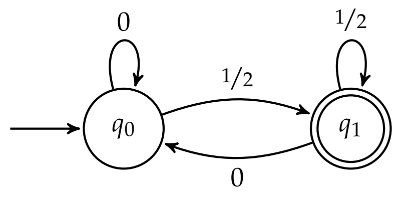
You simply need one extra state, a sort of *purgatory state*, where to there exists a path from all states but q_0 on the 'incorrect' input.

So we can say that in a) we have two paths, the 0 path and the 1 path. We'll then include paths to the purgatory state on 1's on the 0 path and vice versa. Finally, we'll have the purgatory state loop back on itself on both 0 and 1 while not being an exit state:

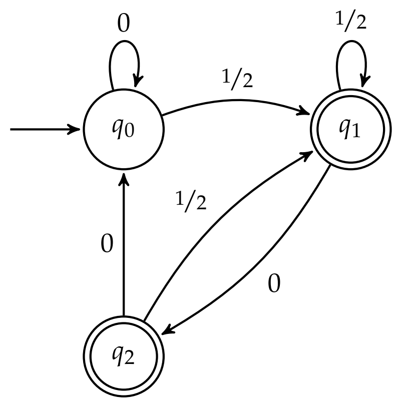


3 Consider the ternary number system.

a) Draw a state diagram for a DFA that accepts ternary strings that are *not* divisible by three.



b) Draw a state diagram for a DFA that accepts ternary strings that are *not* divisible by nine.



4 Design a finite state automata that accepts binary numbers corresponing to perfect squares that can give false positives, but not false negatives.

a) Show that $n^2 \bmod 3 \neq 2$ for all $n = 0, 1, 2, ...$

We prove by induction.

Base case $k = 0$ shows that $0^2 \bmod 3 = 0 \bmod 3 = 0 \neq 2$

So we continue.
Lets assume that the statement is true for some $k \in \mathbb{N}$ so that $k^2 \bmod 3 \neq 2$.
Then: $(k+1)^2 \bmod 3$
$= k^2 + 2k + 1 \bmod 3$
$= ((k^2 \bmod 3) + (2k + 1 \bmod 3)) \bmod 3$
$= ((k^2 \bmod 3) + (2k \bmod 3) + (1 \bmod 3)) \bmod 3$
$= ((k^2 \bmod 3) + 1 + (2k \bmod 3)) \bmod 3$

We see here that since $k^2 \bmod 3 \neq 2$, then $k^2 \bmod 3$ is either 0 or 1.

$k^2 \bmod 3 = 0$: Here we get

$$(2k \bmod 3 + 1) \bmod 3$$

And since $k^2 \bmod 3 = 0$, we notice that the prime factors of k^2 must include 3^2, one for each k. Therefore $k \bmod 3 = 0$ must also be true, and then $2k \bmod 3 = 0$.

So $(2k \bmod 3 + 1) \bmod 3 = 1 \bmod 3 = 1 \neq 2$

$k^2 \bmod 3 = 1$: Here we get

$$(2k \bmod 3 + 2) \bmod 3$$

And since $k^2 \bmod 3 = 1$ we see that the prime factors of k^2 must *not* include 3. Therefore, $k \bmod 3 \neq 0$ so that $2k \bmod 3 \neq 0$ so

$$2k \bmod 3 = 1 \text{ or } 2k \bmod 3 = 2$$

So we get

$$\begin{aligned}&(2k \bmod 3 + 2) \bmod 3\\&= (1+2) \bmod 3\\&= 0\end{aligned}$$

Or

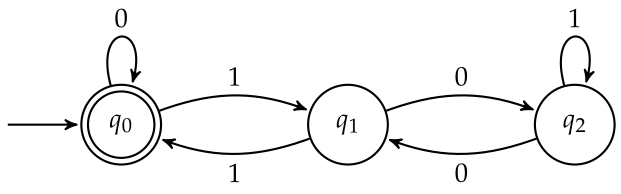
$$\begin{aligned}&(2k \bmod 3 + 2) \bmod 3\\&= (2+2) \bmod 3\\&= 1\end{aligned}$$

Both of which are not 2. So $(k+1)^2 \bmod 3 \neq 2$.

■

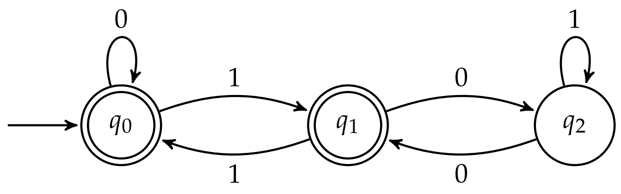
b) By using the above result, design a DFA/NFA that rejects 33% of all possible numbers without any false negatives. Submit the state diagram of your automata.

We notice that the DFA for binary numbers that are divisible by 3 is as follows:



And to satisfy the requirement we only need to accept numbers that, when divided by 3, do not give a remainder of 2.

Therefore, as the automata works with each state signifying what the current remainder of the number is, we just need to make q_1 an exit state along with q_0 and that only leaves numbers giving a remainder of 2 behind.



0
1
1
0
0
1
q_0
q_1
q_2