Automata and Computability Assignment 1

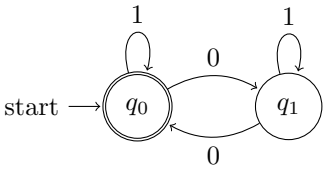
Saksham Maitri - 23787

Q1 Give a DFA for the language of all strings over the alphabet {0, 1} which contain an even number of 0's and at least one 1.

Answer

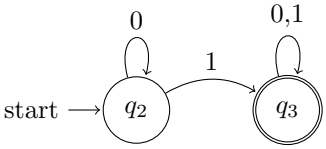
We can use 2 simpler DFAs to construct this DFA.

DFA 1: Accepting even number of 0's

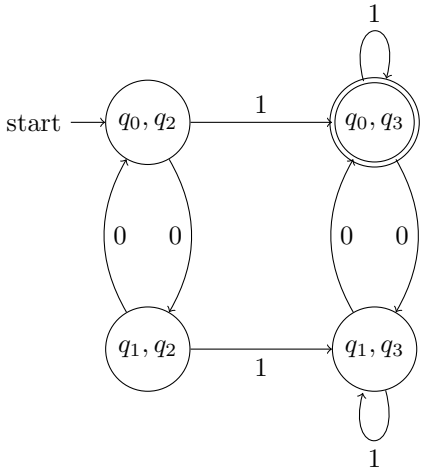


State $q0$ represents ending state for strings with even number of 0s and state $q1$ represents ending state for strings with odd number of 0s.

DFA 2: Accepting at least one 1



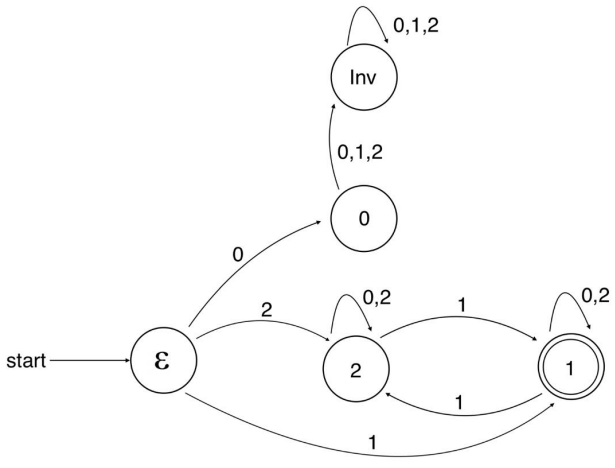
DFA that requires one 1 to reach the final state.
Now we take the intersection of these 2 DFAs.
The set of new states is the cartesian product of the sets of states of DFA 1 and DFA 2.
The set of new final states is the cartesian product of the sets of final states of DFA 1 and DFA 2.
The start state would be (*s1,s2*) , both being the start state of DFA 1 and DFA 2 respectively.



Q2 Show that the set of strings in $\{0, 1, 2\}^*$ which are base 3 representations of odd numbers is regular.

Answer

Let L be the set mentioned in the question. We take $\Sigma = \{1, 2, 3\}$. So we only draw transitions for these 3 symbols, if we were to recieve any other symbol apart from these 3, we can make a new *dead* state that is non accepting and connect all the states to that state. That way we can stop it from getting accepted by our DFA.



DFA accepting this language

When we convert a base-3 string to an integer($base10$), we can express it as:

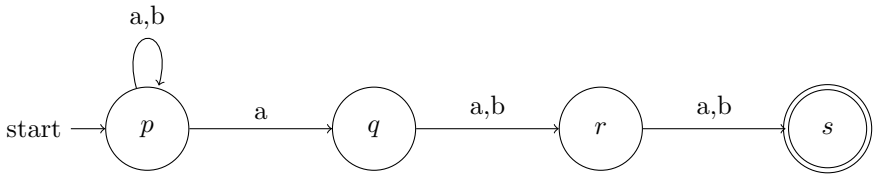
$$\text{String } (a_n a_{n-1} \dots a_1 a_0) = a_n \cdot 3^n + a_{n-1} \cdot 3^{n-1} + \dots + a_0 \cdot 1.$$

Observations

1. Each $a_i \in \{0, 1, 2\}$.
2. All powers of 3 are odd. $\forall k \in \mathbb{N}, 3^k$ is odd
3. - Odd $\times$ Odd = Odd
   - Even $\times$ Odd = Even

Since, Even $\times$ Odd is Even, Our result does not depend on the number of 0 and 2 we see in the string.(Since we want the numbers accepted to be odd) Since 1 is the only odd number in the set $\{0, 1, 2\}$, the parity of the integer is determined by the number of 1's that appear in the string. Therefore, we derive a DFA as shown in the figure.
#1(x) is even $\implies$ x is not accepted by DFA and is not in L. This can be shown that if there are even number of 1s, their sum along with the power of 3(their coefficient) will be even as $(a_1 + a_2 + ... + a_k) mod 2 = (a_1 mod 2 + a_2 mod 2 + ... + a_k mod 2) mod 2 = (1 * k) mod 2 = 0$ (k is the number of 1s in string which is even). Hence we show that the parity of x when expressed in $base10$ depends on #1(x).

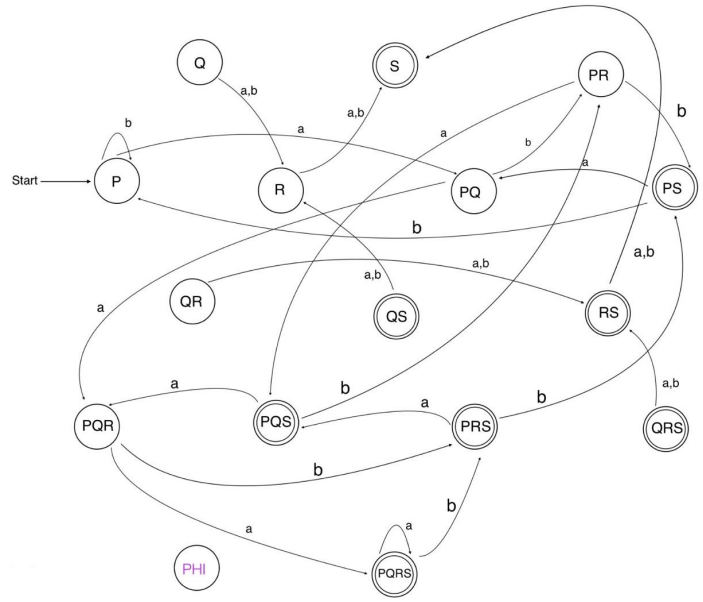
Q3 Consider the NFA below:



(a) Use the subset construction to obtain an equivalent DFA for the NFA below. Label each state of the DFA with the subset of states of the NFA that it corresponds to.

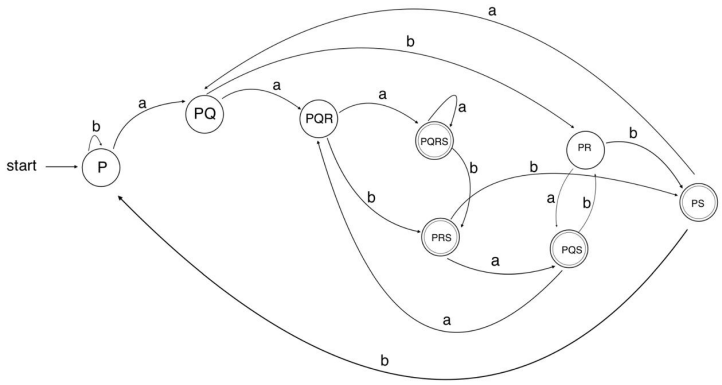
States	a-Transition	b-Transition
∅	∅	∅
{p}	{p,q}	{p}
{q}	{r}	{r}
{r}	{s}	{s}
{s}	∅	∅
{p,q}	{p,q,r}	{p,r}
{p,r}	{p,q,s}	{p,s}
{p,s}	{p,q}	{p}
{q,r}	{r,s}	{r,s}
{q,s}	{r}	{r}
{r,s}	{s}	{s}
{p,q,r}	{p,q,r,s}	{p,r,s}
{p,q,s}	{p,q,r}	{p,r}
{p,r,s}	{p,q,s}	{p,s}
{q,r,s}	{r,s}	{r,s}
{p,q,r,s}	{p,q,r,s}	{p,r,s}

The sets containing **s** as one of the states are the Accepting states.



16-state DFA for part (a).

(b) Give an 8-state DFA which accepts the same language.



An 8-state DFA for part (b).

Q4Consider the language of nested C-style comments. The alphabet comprises the characters "/", "*", and "c" (the latter symbol representing any ASCII character apart from "/" and "*"). The language allows all well-nested and complete comments.

Thus, strings like

`cc/*ccc*/c` and `cc/*cc/*ccc*/c*/ccc`

are in the language, but not

`cc/*c/*cc*/cc`.

Is this language regular? Justify your answer.

Answer

We will use the *demon game* to prove this.

Let the demon choose a number k. Then we take:

$$x = (/*)^k, \quad y = (*/)^k, \quad z = \epsilon$$

Now, the demon will decompose y into u, v, w where $|v| > 0$.

- If $|v|$ is odd, we take $i = 2$. This creates an imbalance in the number of `*` and `/` in y, making the resulting string invalid in the language.
- If $|v|$ is even, we can take any $i > 1$. This results in an incorrect nesting structure, making the string invalid in the language.

Thus, the language does not satisfy the Pumping Lemma for regular languages, proving that it is **not regular**.

Q5 For a set of natural numbers X, define binary(X) to be the set of binary representations of numbers in X. Similarly, define unary(X) to be the set of ***unary*** representations of numbers in X:

$$\mathit{unary}(X) = \{1^n \mid n \in X\}.$$

Thus, for $X = \{2, 3, 6\}$, $\mathit{binary}(X) = \{10, 11, 110\}$ and $\mathit{unary}(X) = \{11, 111, 111111\}$.

Consider the two propositions below:

(a) For all X, if $\mathit{binary}(X)$ is regular then so is unary(X).

(b) For all X, if $\mathit{unary}(X)$ is regular then so is binary(X).

One of the statements above is true, and the other is false. Which is which? Justify your answer.

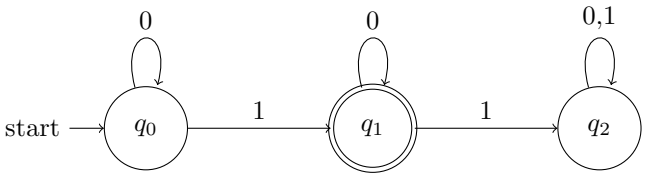


Figure 1: DFA that accepts strings containing exactly one '1'.

Answer

Proposition A: For all X, if *binary(X)* is regular then so is *unary(X)*
We can disprove this using counter example. Let $X = \{2^k \mid k \in \mathbb{N}\}$. Here strings of *binary(X)* contains exactly one 1 and the rest of the digits are 0 as they represent powers of 2 in $base10$. The DFA for this is:

Now, we can prove that $unary(X)$ is not regular. We can do this using the *demon game*:

Demon: takes K

Player: takes $unary(2^q)$ such that $2^q > k$, divides it into 3 parts: $1, 2^q - 1, \epsilon$ matching x, y, z.

CASE 1: Demon divides y in u, v, w such that v is a number $unary(2^n)$ $(2^n < 2^q)$.

Player: takes $i = 2$.

So, now the resulting string has $2^q + 2^n$ ones. We can take $2^n(2^{q-n} + 1)$. The number $2^{q-n} + 1$ is odd. Therefore, we can say that this number cannot be of the form 2^k, where $k \in \mathbb{N}$.

CASE 2: Demon takes k ($k \neq 2^n$ for $n \in \mathbb{N}$).

Player: takes $i = 2$.

So, now the string has $2^q + k$ ones, where $k < 2^q$. The next closest number of the form 2^z, where z is a natural number, is 2×2^q. Since $k < 2^q$, we can say that $2^q + k$ is not of the form 2^n, where n is a natural number (since

$$len(unary(k)) < len(unary(2^q))$$

$2^q + k$ is actually less than 2×2^q).

Hence, we can prove the irregularity of *unary(X)*.

Proposition B: For all X, if *unary(X)* is regular then so is *binary(X)*
We can prove that this statement is true. Lets assume that *unary(X)* is regular, then it must satisfy the ultimate periodicity criteria. If it satisfies the criteria of ultimate periodicity, then *len(unary(X))* should consist of a finite set along with finite number of arithmetic progressions with same common difference.
$len(unary(X)) = F \cup \bigcup_{i=1}^{k} A_i$, where A_i is an arithmetic progression. Our strategy is that we construct DFA for each AP and each element of set F, then we

take the union to accept the required language. Now we want to create a DFA that accepts *binary* of an arithemtic progression. We can do that as following To construct a DFA that accepts a particular arithmetic progression of the form $a, a+d, a+2d, \ldots$, we can use the following procedure:

1. Let the arithmetic progression be $a, a+d, a+2d, \ldots$.
2. Define d states in the DFA as $Q = \{q_0, q_1, q_2, \ldots, q_{d-1}\}$, where q_n represents the numbers that give a remainder of n when divided by d.
3. Set the start state as q_0, and set the accepting state as q_j, where $j = a \bmod d$.
4. Start reading the binary string from left to right. For each digit:
   - Multiply the current value by 2.
   - Add the value of the current digit to it.
   - Take the result modulo d.
   - Transition to the state corresponding to this modulo value.

Formally, the DFA is defined as follows:

$$\Sigma = \{0,1\}, \quad Q = \{q_0, q_1, q_2, \ldots, q_{d-1}\}, \quad s = q_0, \quad F = \{q_j\}, \text{ where } j = a \bmod d.$$

The transition function δ is defined as:

$$\delta(q_j, l) = q_{(2j+l) \bmod d}, \quad \text{for } j \in \{0, 1, 2, \ldots, d-1\} \text{ and } l \in \Sigma.$$

Here:

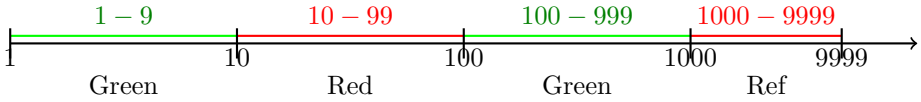
- q_j: Current state.
- l: Current input symbol (either 0 or 1).
- $2j + l$: Update the current value by doubling it (left-shifting) and adding the current input symbol.
- $(2j + l) \bmod d$: Take the modulo d of the updated value to determine the next state.

Then we can create each DFA for the finite number of Arithmetic progressions and create DFA for each entry in that finite set. Then we can take the union of those DFA to get the resulting DFA. Hence, making *binary(X)* a regular language. Hence this proposition is true.

Q6 Construct a language $L \subseteq \{a, b\}^*$ such that neither L nor $\{a, b\}^* - L$ contains an infinite regular subset. Provide an example of such a language.

Answer

$L = \{w \in \{a, b\}^* : |w| \text{ has odd number of digits } \}$
Lets assume $H \subseteq L$ and H is regular language. Then $length(H)$ satisfies the property of ultimate periodicity. Then H should contain an arithemetic progression (Using the criterion $m \in \text{length}(H) \implies m + p \in \text{length}(H)$.) Lets assume there is an Arithemetic progression in $len(H)$ with first term a and common difference d.
Now we need to look at the members of set *len(H)*. This set will contain numbers like 1-9,100-999,10000-99999,etc. (since $H \subseteq L$).
We can divide this realine into gaps according to this,



In this number line, numbers belonging to the green regions are included in $len(L)$, while the red regions correspond to $len(L^C)$.

Claim: The width will increase as we move right in the numberline for both regions.

Proof:

Assume we are in a region, so the region width is 10^n to $10^{n+1} - 1$, the width is $10^{n+1} - 1 - 10^n = 9 \times 10^n - 1$. We can see that this is dependent on n and as we go in the right side of numberline, n will increase $\implies$ gap will increase.

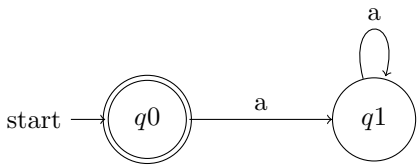
By claim, the width of both regions will increase. Consequently, for any arithmetic progression within $len(H)$, there will eventually be a point where the width of a region will be greater than the arithmetic progression, so there will be a point where first term is in green region and the next term is in red region, violating the ultimate periodicity of $len(H)$, hence the set $len(H)$ is not regular as it doesn't satisfy ultimate periodicity.

We can check the case for L^C in similar manner. Assume L^C contains an infinite regular subset H', then $len(H')$ contains an Arithemetic progression, let the first term and common difference be a' and d' respectively. We can say that there will be a point where the common difference will be less than the width of the region and hence there will be a point where $a' + nd', n \in N$ will occur in green region, violating ultimate periodicity of $len(H')$.

Q7 Let L be an arbitrary subset of $\{a\}^*$. Prove that L^* is regular.

Answer

If $L = \emptyset$, then we are done since $\emptyset^* = \{\epsilon\}$, which is a regular language. There exists a DFA that accepts only this language.



For the case L is non-empty, We can choose $n_0 = p = min(\text{length(L*)})$. We will use the following definition for Ultimate periodicity for set S subset of N. There exist $n_0 \geq 0$, $p \geq 1$ in $\mathbb{N}$, such that for all $m \geq n_0$,

$$m \in X \implies m + p \in X.$$

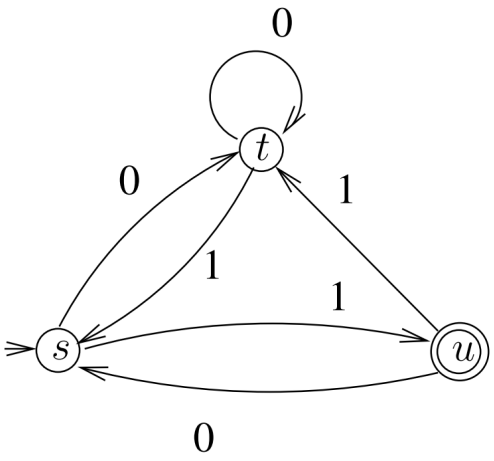
Let the string with the smallest positive length in L^* be c. We have taken $n_0 = p = \text{len}(c)$.

We know that length(L^*) is a subset of $\mathbb{N}$. If $m \geq n_0$ belongs to length(L^*), that means that there exists $x \in L^*$ such that $\text{len}(x) = m$.

Now, since x must be made by concatenating some elements from L (because $x \in L^*$), we can say that the concatenation of x and c, xc is also formed by elements of L, and hence $xc \in L^*$.

Thus, $\text{len}(xc) \in \text{length}(L^*)$, which implies that $m + p \in \text{length}(L^*)$, completing our proof.

Q8. Use the construction done in class to construct a regular expression corresponding to the language accepted by the DFA below (i.e., the expression corresponding to $L_{su}^{\{s,t,u\}}$).



Answer

We will follow the construction done in class. We will remove the state t initially.
so the expression will look like:

$$L_{su}^{\{s,t,u\}} = L_{su}^{\{s,u\}} + L_{st}^{\{s,u\}}(L_{tt}^{\{s,u\}})^* L_{tu}^{\{s,u\}}$$

We can solve this using the recursive method discussed in class.

$L_{su}^{X} = \{$all strings $x \mid \hat{\delta}(s,x) = u$ for $x \in X$ and all the intermediate states (apart from start and end state) be

$$L_{su}^{\{\}} = \{x \in \Sigma : \hat{\delta}(s,x) = u\} \quad \text{if} \quad s \neq u$$

$$L_{su}^{\{\}} = \{x \in \Sigma : \hat{\delta}(s,x) = s\} \cup \{\epsilon\} \quad \text{if} \quad s = u$$

Then:

$$L_{su}^{\{s,t,u\}} = L_{su}^{\{s,u\}} + L_{st}^{\{s,u\}} \left(L_{tt}^{\{s,u\}}\right)^* L_{su}^{\{s,u\}}$$

Term $L_{su}^{\{s,u\}}$

$$L_{su}^{\{s,u\}} = L_{su}^{\{s\}} + L_{su}^{\{s\}} \left(L_{uu}^{\{s\}}\right)^* L_{uu}^{\{s\}}$$

$$L_{su}^{\{s\}} = L_{su}^{\{\}} + L_{ss}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{su}^{\{\}} = 1 + \epsilon^* 1 = 1$$

$$L_{uu}^{\{s\}} = L_{uu}^{\{\}} + L_{us}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{su}^{\{\}} = \emptyset + 0\epsilon^* 1 = 01$$

$$L_{su}^{\{s,u\}} = 1 + 1(01)^* 01 = 1(01)^*$$

Term $L_{st}^{\{s,u\}}$

$$L_{st}^{\{s,u\}} = L_{st}^{\{s\}} + L_{su}^{\{s\}} \left(L_{uu}^{\{s\}}\right)^* L_{ut}^{\{s\}}$$

$$L_{st}^{\{s\}} = L_{st}^{\{\}} + L_{ss}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{st}^{\{\}} = 0 + \epsilon(\epsilon)^* 0 = 0$$

$$L_{su}^{\{s\}} = L_{su}^{\{\}} + L_{ss}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{su}^{\{\}} = 1 + \epsilon(\epsilon)^* 1 = 1$$

$$L_{uu}^{\{s\}} = L_{uu}^{\{\}} + L_{us}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{su}^{\{\}} = \epsilon + 0(\epsilon)^* 1 = \epsilon + 01$$

$$L_{ut}^{\{s\}} = L_{ut}^{\{\}} + L_{us}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{st}^{\{\}} = 1 + 0(\epsilon)^* 0 = 1 + 00$$

$$L_{st}^{\{s,u\}} = 0 + 1(\epsilon + 01)^*(1 + 00)$$

Term $L_{tu}^{\{s,u\}}$

$$L_{tt}^{\{s,u\}} = L_{tt}^{\{s\}} + L_{tu}^{\{s\}} \left(L_{uu}^{\{s\}}\right)^* L_{ut}^{\{s\}}$$

$$L_{tt}^{\{s\}} = L_{tt}^{\{\}} + L_{ts}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{st}^{\{\}} = 0 + 1(\epsilon)^* 0 = 0 + 10$$

$$L_{tu}^{\{s\}} = L_{tu}^{\{\}} + L_{ts}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{su}^{\{\}} = \emptyset + 1(\epsilon)^* 1 = 11$$

$$L_{uu}^{\{s\}} = L_{uu}^{\{\}} + L_{us}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{su}^{\{\}} = \emptyset + 0(\epsilon)^* 1 = 01$$

$$L_{ut}^{\{s\}} = L_{ut}^{\{\}} + L_{us}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{st}^{\{\}} = 1 + 0(\epsilon)^* 0 = 1 + 00$$

$$L_{tt}^{\{s,u\}} = (0 + 10) + (11)(01)^*(1 + 00)$$

Term $L_{tt}^{\{s,u\}}$

$$L_{tu}^{\{s,u\}} = L_{tu}^{\{s\}} + L_{tu}^{\{s\}} \left(L_{uu}^{\{s\}}\right)^* L_{uu}^{\{s\}}$$

$$L_{tu}^{\{s\}} = L_{tu}^{\{\}} + L_{ts}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{su}^{\{\}} = \emptyset + 1(\epsilon)^*1 = 11$$

$$L_{uu}^{\{s\}} = L_{uu}^{\{\}} + L_{us}^{\{\}} \left(L_{ss}^{\{\}}\right)^* L_{su}^{\{\}} = \emptyset + 0(\epsilon)^*1 = 01$$

$$L_{tu}^{\{s,u\}} = 11 + 11(01)^*01 = 11(01)^*$$

Finally,

$$L_{su}^{\{s,t,u\}} = 1(01)^* + (0 + 1(01)^*(1 + 00))((0 + 10) + (11)(01)^*(1 + 00))^*(11(01)^*)$$