Final computational work

Luis Mantilla
código:2182929

March 1, 2021

Given the language: $\mathcal{L} = \{w \in \Sigma^* || w|_a \equiv 0 \ (mod\ 2) \ \wedge \ |w|_b + |w|_c \leq 3\}$.

1 AUTOMATON

Let's note that we can divide this language into two types of Automata, one for $|w|_a \equiv 0 \ (mod\ 2)$ and the other for $|w|_b + |w|_c \leq 3$, separately remain as follows:

1. $|w|_a \equiv 0 \ (mod\ 2)$: Naming the initial state q_0 , it can be seen that it is necessary two states, calling the lack of q_1,According to Figure 1, we can ensure that every time a word ends in the state q_0 has an even number of letters a, this way the only state of acceptance for this automaton is the q_0.

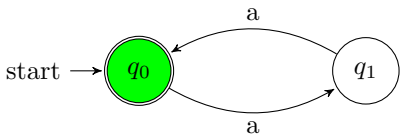

Figure 1: $|w|_a \equiv 0 \ (mod\ 2)$

If we find the adjacency mtariz of this automata, we have the following, the green color relates to the number of words accepted with a certain number of letters.

$$M_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1}$$

Now if we calculate e characteristic polynomial, then $P_{M_1} = det(M_1 - \lambda I_4)$ we have the following polynomial already factored:

$$P_{M_1} = \lambda^2 - 1 = (\lambda - 1)(\lambda + 1)$$

That is, the recursive function for this language replacing λ=$|\mathcal{L}_n + 1|$ is:

$$|\mathcal{L}_{n+2}| = |\mathcal{L}_n|$$

this above is clear since if the number n of letters is for there will only be one word, which will be $|aaaa \cdots a| = n$ and if it is odd it is clear that the automaton does not accept any words.

2. $|w|_b + |w|_c \leq 3$: For this automaton it is clear that four Automata are needed as in Figure 2, since at most the sum of the number of $b - s$ plus the number of $c - s$ it has to give a maximum of 3, and as acceptance states are all.

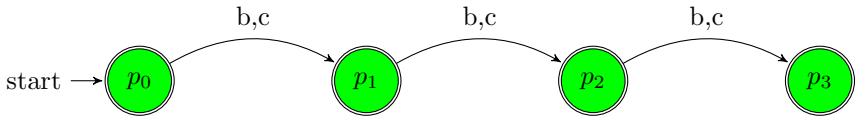

Figure 2: $|w|_b + |w|_c \leq 3$

It is clear to see that for a letter there are 2 accepted words, for 2 words there are 4, for 3 letters there are 8 words, and for a greater number of letters there is no word that accepts the automaton, that is, we can define a recursive formula of the language accepted by this automaton as follows $|\mathcal{L}_n| = 2^n$ para $n < 4$ with n natural.

Now let's define the components of the language interception accepted by the above-mentioned Automata:

- $M = (\Sigma, Q, F, q_0, \delta)$
- $\Sigma = \{a, b, c\}$
- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$
- Initial state: q_0
- Acceptance States: $F = \{q_0, q_2, q_4, q_6\}$
- Transition function δ

Below is the table representing the transition function:

δ	a	b	c
q_0	q_1	q_2	q_2
q_1	q_0	q_3	q_3
q_2	q_3	q_4	q_4
q_3	q_2	q_5	q_5
q_4	q_5	q_6	q_6
q_5	q_4	q_7	q_7
q_6	q_7	-	-
q_7	q_6	-	-

Table 1: Transition function

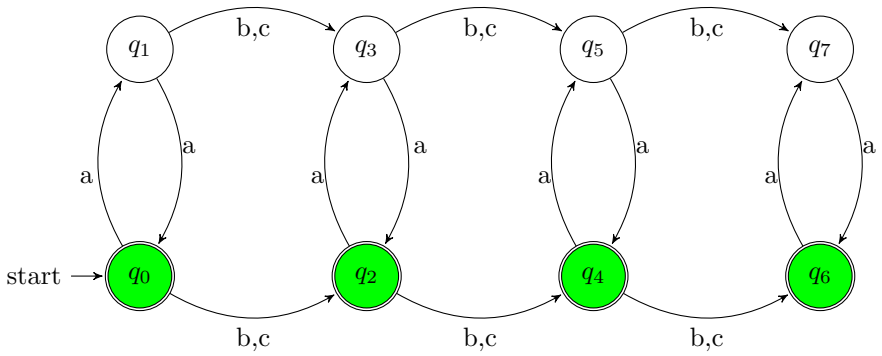

Figure 3: Automaton

So we have the automaton that represents language: $\mathcal{L} = \{w \in \Sigma^* || w|_a \equiv 0 \ (mod\ 2) \wedge |w|_b + |w|_c \leq 3\}$.

1.1 Observation

Now if we look closely, we have that the automaton of Figure 1 is repeating, the latter without being returned, four times thanks to the nature of the automaton of Figure 2, that shows that we have the same language repeated four times, later we will see the importance of this result.

2 Regular expression

Let's find the regular expression, removing the states, as the first part let's remove the state q_0.

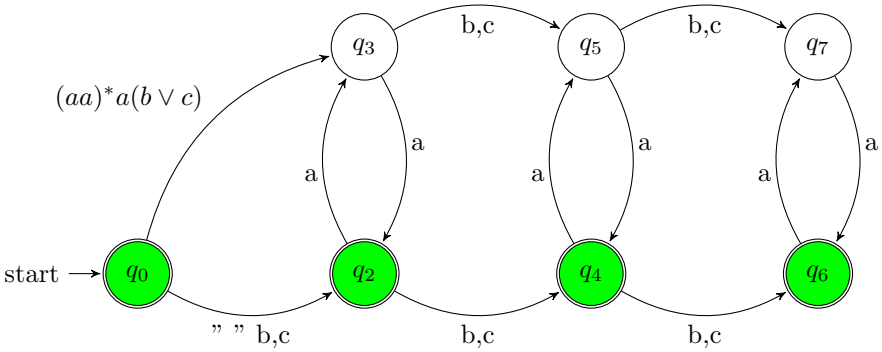

Figure 4: without q_0

Now let's remove the states q_2 y q_3, así:

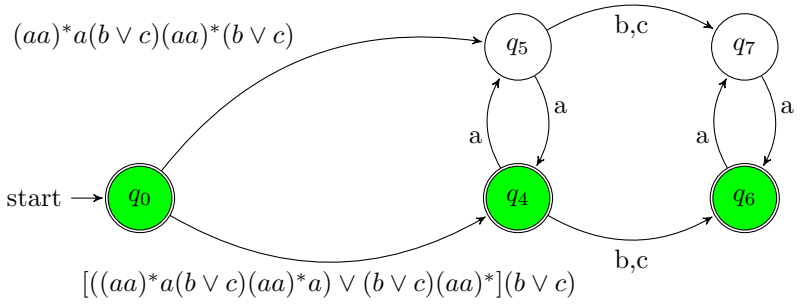

Figure 5: without q_2 y q_3

We continue to take away the states q_5 y q_4

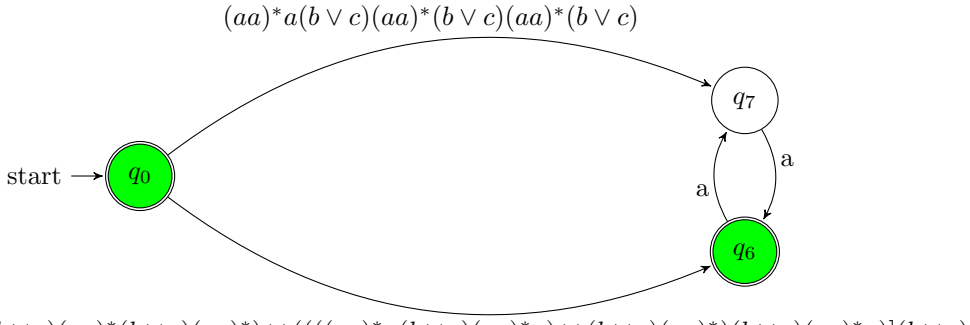

Figure 6: without q_5 y q_4

Now let's move on to the state q_7:

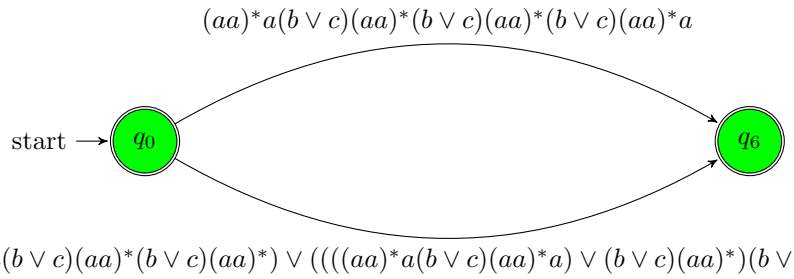

Figure 7: without q_7

Finally let's remove the state q_6:

$$[[((aa)^*a(b \vee c)(aa)^*(b \vee c)(aa)^*) \vee ((((aa)^*a(b \vee c)(aa)^*a) \vee (b \vee c)(aa)^*)(b \vee c)(aa)^*a)](b \vee c)] \vee [(aa)^*a(b \vee c)(aa)^*(b \vee c)(aa)^*(b \vee c)(aa)^*a]$$

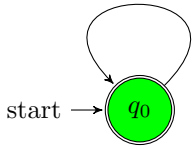

Figure 8: without q_6

From the above we can deduce the regular expression, which that as follows:

$$[[((aa)^*a(b \vee c)(aa)^*(b \vee c)(aa)^*) \vee ((((aa)^*a(b \vee c)(aa)^*a) \vee (b \vee c)(aa)^*)(b \vee c)(aa)^*a)](b \vee c)]$$

$$\vee[(aa)^*a(b \vee c)(aa)^*(b \vee c)(aa)^*(b \vee c)(aa)^*a]$$

3 Census function

First of all we will find the adjacency matrix which is as follows:

$$M = \begin{pmatrix} 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{2}$$

Let us observe that the terms in green are the words of length one that ends up in a state of acceptance, which are q_0,q_2,q_4,q_6 respectively, Now let's look at the different powers of this matrix adjacency:

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

$$M^2 = \begin{pmatrix} 1 & 0 & 0 & 4 & 4 & 0 & 0 & 0 \\ 0 & 1 & 4 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 4 & 4 & 0 \\ 0 & 0 & 0 & 1 & 4 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

$$M^3 = \begin{pmatrix} 0 & 1 & 6 & 0 & 0 & 12 & 8 & 0 \\ 1 & 0 & 0 & 6 & 12 & 0 & 0 & 8 \\ 0 & 0 & 0 & 1 & 6 & 0 & 0 & 12 \\ 0 & 0 & 1 & 0 & 0 & 6 & 12 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 6 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{5}$$

$$M^4 = \begin{pmatrix} 1 & 0 & 0 & 8 & 24 & 0 & 0 & 32 \\ 0 & 1 & 8 & 0 & 0 & 24 & 32 & 0 \\ 0 & 0 & 1 & 0 & 0 & 8 & 24 & 0 \\ 0 & 0 & 0 & 1 & 8 & 0 & 0 & 24 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 1 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{6}$$

$$M^5 = \begin{pmatrix} 0 & 1 & 10 & 0 & 0 & 40 & 80 & 0 \\ 1 & 0 & 0 & 10 & 40 & 0 & 0 & 80 \\ 0 & 0 & 0 & 1 & 10 & 0 & 0 & 40 \\ 0 & 0 & 1 & 0 & 0 & 10 & 40 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 10 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{7}$$

$$M^6 = \begin{pmatrix} 1 & 0 & 0 & 12 & 60 & 0 & 0 & 160 \\ 0 & 1 & 12 & 0 & 0 & 60 & 160 & 0 \\ 0 & 0 & 1 & 0 & 0 & 12 & 60 & 0 \\ 0 & 0 & 0 & 1 & 12 & 0 & 0 & 60 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 1 & 12 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{8}$$

$$M^7 = \begin{pmatrix} 0 & 1 & 14 & 0 & 0 & 84 & 280 & 0 \\ 1 & 0 & 0 & 14 & 84 & 0 & 0 & 280 \\ 0 & 0 & 0 & 1 & 14 & 0 & 0 & 84 \\ 0 & 0 & 1 & 0 & 0 & 14 & 84 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 14 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 14 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{9}$$

Now if we make the table that represents me the first row of each array, the words that start in the state q_0, we have the following table:

n	a_n	b_n	c_n	d_n	e_n	f_n	g_n	h_n	$\|\mathcal{L}_n\|$
0	1	0	0	0	0	0	0	0	1
1	0	1	2	0	0	0	0	0	2
2	1	0	0	4	4	0	0	0	5
3	0	1	6	0	0	12	8	0	14
4	1	0	0	8	24	0	0	32	25
5	0	1	10	0	0	40	80	0	90
6	1	0	0	12	60	0	0	160	61
7	0	1	14	0	0	84	280	0	294

Table 2: censo

The Column $|\mathcal{L}_n|$ count accepted words as the sum of the number of words in the acceptance States. Which we can say that the census function has the first 8 initial values:

$$\{1, 2, 5, 14, 25, 90, 61, 294, \cdots\}$$

Now to find the explicit census function, we will first find the characteristic polynomial, first pass the adjacency matrix to the normal form of Jordan, calculating $P_M(\lambda) = det(M - \lambda I_8)$ be I_8 is the identity matrix of size 8x8 y λ the random variable as well:

$$M - \lambda I_8 = \begin{pmatrix} 0-\lambda & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0-\lambda & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0-\lambda & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0-\lambda & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0-\lambda & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0-\lambda & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0-\lambda & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0-\lambda \end{pmatrix} \tag{10}$$

them:

$$P_M(\lambda) = \lambda^8 - 4\lambda^6 + 6\lambda^4 - 4\lambda^2 + 1 \tag{11}$$

Therefore the law of recurrence is:

$$|\mathcal{L}_{n+8}| = 4|\mathcal{L}_{n+6}| - 6|\mathcal{L}_{n+4}| + 4|\mathcal{L}_{n+2}| - |\mathcal{L}_n| \tag{12}$$

with initial values $|\mathcal{L}_0| = 1, |\mathcal{L}_1| = 2, |\mathcal{L}_2| = 5, |\mathcal{L}_3| = 14, |\mathcal{L}_4| = 25, |\mathcal{L}_5| = 90, |\mathcal{L}_6| = 61, |\mathcal{L}_7| = 294$.
The characteristic polynomial can be factorized, let's find the roots of this polynomial with their respective multiplicities, first make a variable change, $\mu = \lambda^2$, then the polynomial equal to zero remains as follows:

$$0 = \mu^4 - 4\mu^3 + 6\mu^2 - 4\mu + 1$$

It is easy to see that the above polynomial can be written as follows:

$$0 = (\mu - 1)^4$$

Now substituting for the variable change, we have the following:

$$0 = (\lambda^2 - 1)^4 = ((\lambda - 1)(\lambda + 1))^4$$

The latter is evidenced by observation 1.1, since it had been concluded that the automaton of Figure 1, is repeated 4 times in the latter automaton M, for this reason and by the Cayley-Hamilton Theorem, The because of the equation (11) and the law of recurrence of the equation (12) can be deduced.

Now if we calculate the normal Jordan form of the Matrix M, we have the following:

$$J = \left(\begin{array}{cccc|cccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{array}\right) \tag{13}$$

Given this matrix, let's look for a generalization of this, since if we have the general form of the matrix, it will give us an idea of how to construct our census function $|\mathcal{L}_n|$.

Let's prove by induction that the general form and we will call it J_n:

$$\left(\begin{array}{cccc|cccc} 1 & n & \frac{1}{2}n^2 - \frac{1}{2}n & \frac{1}{6}n^3 - \frac{1}{2}n^2 + \frac{1}{3}n & 0 & 0 & 0 & 0 \\ 0 & 1 & n & \frac{1}{2}n^2 - \frac{1}{2}n & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & n & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & (-1)^n & -(-1)^n n & \frac{1}{2}(n^2 - n)(-1)^n - \frac{1}{6}(n^3 - 3n^2 - 2^n)(-1)^n & 0 \\ 0 & 0 & 0 & 0 & 0 & (-1)^n & -(-1)^n n & \frac{1}{2}(n^2 - n)(-1)^n \\ 0 & 0 & 0 & 0 & 0 & 0 & (-1)^n & -(-1)^n n \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & (-1)^n \end{array}\right)$$

Let's note that for $n = 1$, it is clear that the matrix is left to us of the form:

$$J = \left(\begin{array}{cccc|cccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{array}\right) \tag{14}$$

Now suppose that it is fulfilled for n,missing show for $n+1$, for this let's make the product of arrays of $J_n \cdot J$and as a result we have the following matrix, for the sake of space, we are going to use the quadrants of the matrices that interest us, which are, the upper left end and the lower right end:

$$\begin{pmatrix} 1 & n+1 & \frac{1}{2}(n+1)^2 - \frac{1}{2}(n+1) & \frac{1}{6}(n+1)^3 - \frac{1}{2}(n+1)^2 + \frac{1}{3}(n+1) \\ 0 & 1 & n & \frac{1}{2}n^2 - \frac{1}{2}n \\ 0 & 0 & 1 & n \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and

$$\begin{pmatrix} (-1)^{(n+1)} & -(-1)^{(n+1)}(n+1) & \frac{1}{2}((n+1)^2-(n+1))(-1)^{(n+1)} - \frac{1}{6}((n+1)^3 - 3(n+1)^2 - 2^{(n+1)})(-1)^{(n+1)} & 0 \\ 0 & (-1)^{(n+1)} & -(-1)^{(n+1)}(n+1) & \frac{1}{2}((n+1)^2-(n+1))(-1)^{(n+1)} \\ 0 & 0 & (-1)^{(n+1)} & -(-1)^{(n+1)}(n+1) \\ 0 & 0 & 0 & (-1)^{(n+1)} \end{pmatrix}$$

From the above it is shown that it is fulfilled for $n+1$ entonces then by induction we can say that the general form of the normal Jordan form of the Matrix M.

Looking at The Matrix J podemos we can realize from the behavior of its diagonal that the census function has to behave as follows:

$$|\mathcal{L}_n| = \alpha_1 + \alpha_2 n + \alpha_3 n^2 + \alpha_4 n^3 + (-1)^n \alpha_5 + (-1)^n \alpha_6 n + (-1)^n \alpha_7 n^2 + (-1)^n \alpha_8 n^3$$

Be α_i constant.
Now for Table 2, we have the following system of equations with the variables $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8\}$:

$$\left(\begin{array}{cccccccc|c} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 2 \\ 1 & 2 & 4 & 8 & 1 & 2 & 4 & 8 & 5 \\ 1 & 3 & 9 & 27 & -1 & -3 & -9 & -27 & 14 \\ 1 & 4 & 16 & 67 & 1 & 4 & 16 & 64 & 25 \\ 1 & 5 & 25 & 125 & -1 & -5 & -25 & -125 & 90 \\ 1 & 6 & 36 & 216 & 1 & 6 & 36 & 216 & 61 \\ 1 & 7 & 49 & 343 & -1 & -7 & -49 & -343 & 294 \end{array}\right)$$

now since the Matrix has determinant other than zero, we say that it has only solution then it remains only to solve the system of equations, which we will have the following result:
$\alpha_1 = 1/2$
$\alpha_2 = 4/3$
$\alpha_3 = -1$
$\alpha_4 = 2/3$
$\alpha_5 = 1/2$
$\alpha_6 = -10/3$
$\alpha_7 = 3$
$\alpha_8 = -2/3$

Given the above we deduce the census function as follows:

$$|\mathcal{L}_n| = \frac{1}{2} + \frac{4}{3}n - n^2 + \frac{2}{3}n^3 + \frac{1}{2}(-1)^n - \frac{10}{3}(-1)^n n + 3(-1)^n n^2 - \frac{2}{3}(-1)^n n^3$$

4 Simulation in SAGEMATH

In this section we will perform a simulation using the language of SageMAth, about the alphabet $\{a, b, c\}$.

Let's find the words with only one letter, which are 2 words which are:

$$\{b, c\}$$

```
In [1]: A=range(3)                                                                    0.017 seconds 1
        L=cartesian_product([A])
        def delta (aut,qi,n):
            return aut[qi][n]

        def delta2(aut,pal):
            estado=0
            for ff in pal:
                estado=delta(aut,estado,ff)
            return estado

In [2]: aut=([1,2,2],[0,3,3],[3,4,4],[2,5,5],[5,6,6],[4,7,7],[7,8,8],[6,8,8],[8,8,8])   0.004 seconds 2

In [3]: aceptada=[k for k in L if delta2(aut,k)==0 or delta2(aut,k)==2 or delta2(aut,k)==4 or delta2(aut,k)==6]   0.004 seconds 3

In [4]: len(aceptada)                                                                 0.006 seconds 4
Out[4]: 2
```

Let's find the words with two letters, which are 5 words which are:

$$\{aa, bc, cb, bb, cc\}$$

```
In [5]: A=range(3)                                                                    0.003 seconds 1
        L=cartesian_product([A,A])
        def delta (aut,qi,n):
            return aut[qi][n]

        def delta2(aut,pal):
            estado=0
            for ff in pal:
                estado=delta(aut,estado,ff)
            return estado

In [6]: aut=([1,2,2],[0,3,3],[3,4,4],[2,5,5],[5,6,6],[4,7,7],[7,8,8],[6,8,8],[8,8,8])   0.004 seconds 2

In [7]: aceptada=[k for k in L if delta2(aut,k)==0 or delta2(aut,k)==2 or delta2(aut,k)==4 or delta2(aut,k)==6]   0.004 seconds 3

In [8]: len(aceptada)                                                                 0.008 seconds 4
Out[8]: 5
```

Let's find the words with 3 letters, which are 14 words which are:

$$\{aab, aac, aba, aca, baa, bbb, bbc, bcb, bcc, caa, cbb, cbc, ccb, bbb\}$$

```
In [9]:  A=range(3)
         L=cartesian_product([A,A,A])
         def delta (aut,qi,n):
             return aut[qi][n]

         def delta2(aut,pal):
             estado=0
             for ff in pal:
                 estado=delta(aut,estado,ff)
             return estado
```
0.003 seconds 1

```
In [10]: aut=([1,2,2],[0,3,3],[3,4,4],[2,5,5],[5,6,6],[4,7,7],[7,8,8],[6,8,8],[8,8,8])
```
0.004 seconds 2

```
In [11]:  aceptada=[k for k in L if delta2(aut,k)==0 or delta2(aut,k)==2 or delta2(aut,k)==4 or delta2(aut,k)==6]
```
0.004 seconds 3

```
In [12]: len(aceptada)
Out[12]: 14
```
0.005 seconds 4

Let's find the words with 4 letters, which are 25 words which are:

$$\{aaaa, aabb, aabc, aacb, aacc, abab, abac, abba, abca, acab, acac, acba$$

$$, baab, baac, baba, baca, bbaa, bcaa, caab, caac, caba, caca, cbaa, ccaa\}$$

```
In [13]: A=range(3)
         L=cartesian_product([A,A,A,A])
         def delta (aut,qi,n):
             return aut[qi][n]

         def delta2(aut,pal):
             estado=0
             for ff in pal:
                 estado=delta(aut,estado,ff)
             return estado
```
0.006 seconds 1

```
In [14]: aut=([1,2,2],[0,3,3],[3,4,4],[2,5,5],[5,6,6],[4,7,7],[7,8,8],[6,8,8],[8,8,8])
```
0.003 seconds 2

```
In [15]:  aceptada=[k for k in L if delta2(aut,k)==0 or delta2(aut,k)==2 or delta2(aut,k)==4 or delta2(aut,k)==6]
```
0.007 seconds 3

```
In [16]: len(aceptada)
Out[16]: 25
```
0.005 seconds 4

5 BIBLIOGRAPHY

- De Castro, Rodrigo Teoría de la Computación Universidad Nacional de Colombia.
- Sagemath. https://cocalc.com.
- OEIS On-line Enciclopedy of Integer Sequences. https://oeis.org/?language=spanish.