Universität Bremen

FEB-Projekt

Aperiodic Order of Quasicrystals and Diffraktion

By: Lothar Dirks und Chris Michel Peters

06.09.2018

Supported by: Arne Mosbach und Prof. Dr. Marc Keßeböhmer

We want to thank Arne Mosbach for the creation of this work sheet on substitutions. The main goal of our own in this was the successfull completion of the tasks given in it.

Introduction

The aim of this sheet is to present some basic properties and notation of the structures in question. Along with this are tasks, which should be dealt with while progressing within the sheet. The next paragraph will introduce basic notations to work with languages.

For a finite alphabet $\mathcal{A}$ we denote by $\mathcal{A}^* := \{u \in \mathcal{A}^n : n \in \mathbb{N}\}$ the set of all finite words in $\mathcal{A}$ and by $\mathcal{A}^{\mathbb{N}}$ all infinite words. A semigroup homomorphism $\sigma\colon \mathcal{A} \to \mathcal{A}^*$ on $\mathcal{A}^*$ or $\mathcal{A}^{\mathbb{N}}$ is called substitution. The name semigroup homomorphism is from the fact that $\sigma(u) \mapsto \sigma(u_0)\sigma(u_1)\sigma(u_2)\ldots$ is well defined, for any finite of the infinite word u. As just indicated finite sequences $u = (u_i)_{i=0}^{n-1} \in \mathcal{A}^n$ for some $n \in \mathbb{N}$ may also be denoted by $u_0u_1u_2\ldots u_{n-1}$ or may even be functions $u\colon \{0,\ldots,n-1\} \to \mathcal{A}^n$. The length of any $u \in \mathcal{A}^* \cup \mathcal{A}^{\mathbb{N}}$ is given by $|u| := n$, if $u = u_0\ldots u_{n-1} \in \mathcal{A}^n$, while $|u| = \infty$, if $u \in \mathcal{A}^{\mathbb{N}}$. Moreover for any $v \in \mathcal{A}^*$ we define

$$|u|_v := |\{n \in \mathbb{N} : \forall 0 \le i \le |v|-1, u_{n-i} = v_i\}|,$$

to be the *occurences of v in u*. As already used and known from sequences, letters of u are adressed by u_n for some $n \in \mathbb{N}$, *factors* of u are all finite words of the form $u_{[n,n+m]} := \{u_i : n \le i \le n+m\}$ and *subwords* of u are factors, but may also be infinite, hence of the form $u_{[n,\infty]}$. The *prefix* of u of length n is defined to be the first n letters of $u_{|n} := u_{[0,n-1]}$, while the *suffix* of length n given by $u_{[|u|-n,|u|-1]}$ is only defined for finite words.

Definition 0.1. A word $u \in \mathcal{A}^{\mathbb{N}}$ is called *periodic*, if it exists a $v \in \mathcal{A}^*$ such that for all $m \in \mathbb{N}$, $u_{|(m|v|)} = v^m$, where $v^m \in \mathcal{A}^{m|v|}$ is the unique word that satisfies $(v^m)_j = v_{(j \mod |v|)}$ for $0 \le j \le m|v|$. u is *ultimately periodic*, if it exists an infinite periodic subword of u.

Remark 0.2. Take note that $\mathbb{N} = \{0,1,2,3,\ldots\}$, while $\mathbb{N}_+ = \{1,2,3,4,\ldots\}$. Also we make use of the convention $\mathcal{A}^0 := \{\emptyset\}$, while the empty word is also denoted by ε.

1 Rotation substitutions

From now on we will only consider the alphabet $\mathcal{A} = \{0,1\}$.

Definition 1.1. In the following let τ, ρ, θ and τ_{TM} denote the semigroup homomorphisms on $\{0,1\}^*, \{0,1\}^{\mathbb{N}}$ determined by

$$\tau\colon \begin{cases} 0 \mapsto 0 \\ 1 \mapsto 10 \end{cases}, \qquad \rho\colon \begin{cases} 0 \mapsto 01 \\ 1 \mapsto 1 \end{cases}, \qquad \theta\colon \begin{cases} 0 \mapsto 1 \\ 1 \mapsto 0 \end{cases}, \qquad \tau_{\mathrm{TM}}\colon \begin{cases} 0 \mapsto 01 \\ 1 \mapsto 10 \end{cases}$$

Further, for any finite or infinite non-empty word $u = u_0u_1u_2u_3\ldots$ we set $S(u) := u_1u_2u_3\ldots$ and if u is the empty word $S(u) := u$.

Definition 1.2. Let $(a_i)_{i\in\mathbb{N}} \in \mathbb{N}_+^{\mathbb{N}_+}$ be a sequence of natural numbers. We define for $l \in \{0,1\}$, $L \coloneqq l\theta l = l\theta(l)$ the finite words

$$\omega_l^j \coloneqq \omega_L^j \coloneqq \begin{cases} \tau^{a_1}\rho^{a_2}\tau^{a_3}\dots\tau^{a_{j-1}}\rho^{a_j-1}(L), & (-1)^j = 1 \\ \tau^{a_1}\rho^{a_2}\tau^{a_3}\dots\rho^{a_{j-1}}\tau^{a_j-1}(L), & (-1)^j = -1 \end{cases}$$
$$\omega_0^0 \coloneqq 0, \qquad \omega_1^0 \coloneqq 1, \qquad \omega_L^1 \coloneqq L,$$

where $j \geq 2$.

Remark 1.3. Some consequences of Definition 1.2 are

$$L = l\theta l = \tau_{\mathrm{TM}} l = \begin{cases} 01, & l = 0 \\ 10, & l = 1 \end{cases}.$$

and the cases $(-1)^j \in \{-1,1\}$ just gives off if j is even or odd.

Example 1.4. Let $n \in \mathbb{N}$,

$$\rho^n(01) = 01^{n+1}, \qquad \tau^n(01) = 010^n = \omega_0^1,$$
$$\rho^n(10) = 101^n, \qquad \tau^n(10) = 10^{n+1} = \omega_1^1.$$

A direct consequence of the former observations are the identities $\rho\tau\theta = \rho\theta\rho = \theta\tau\rho$. These obervations are expressed by the following diagram for all $n \in \mathbb{N}$

$$\begin{array}{ccc} \tau^n(10) & \xleftrightarrow[\theta]{} & \rho^n(01) \\ 01S^2 \updownarrow & & \updownarrow 01S^2 \\ \tau^n(01) & \xleftrightarrow[\theta]{} & \rho^n(10) \end{array}$$

We conclude this example by noting down ω_l^2 as a mixture of ω's

$$\omega_0^2 = \tau^{a_1}(01^{a_2}) = 0(10^{a_1})^{a_2} = \omega_0^0(\omega_1^1)^{a_2},$$
$$\omega_1^2 = \tau^{a_1}(101^{a_2-1}) = 10^{a_1}0(10^{a_1})^{a_2-1} = \omega_1^1\omega_0^0(\omega_1^1)^{a_2-1}.$$

This observation holds in general and will be discussed in the upcoming lemma.

Lemma 1.5. *Let $j \geq 2$ and $l \in \{0,1\}$, then ω_l^j can also be expressed by one of the following cases:*

j even:

$$\omega_l^j = \begin{cases} \omega_0^{j-2}(\omega_1^{j-1})^{a_j} & , \quad l = 0 \\ \omega_1^{j-1}\omega_0^{j-2}(\omega_1^{j-1})^{a_j-1} & , \quad l = 1 \end{cases}$$

j odd:

$$\omega_l^j = \begin{cases} \omega_0^{j-1}\omega_1^{j-2}(\omega_0^{j-1})^{a_j-1} & , \quad l = 0 \\ \omega_1^{j-2}(\omega_0^{j-1})^{a_j} & , \quad l = 1 \end{cases}$$

Proof. The proof is done by induction. In fact Example 1.4 shows the base case of the induction, whether in the following the inductive step is given by using the calculations done in the example.

For an even j that is:

$$\begin{aligned}\omega_l^j &= \begin{cases}\tau^{a_1}\rho^{a_2}\dots\tau^{a_{j-1}}(01^{a_j}) & , \quad l=0\\ \tau^{a_1}\rho^{a_2}\dots\tau^{a_{j-1}}(101^{a_j-1}) & , \quad l=1\end{cases}\\ &= \begin{cases}\tau^{a_1}\rho^{a_2}\dots\rho^{a_{j-2}-1}(01)(\tau^{a_1}\rho^{a_2}\dots\tau^{a_{j-1}-1}(10))^{a_j} & , \quad l=0\\ \tau^{a_1}\rho^{a_2}\dots\tau^{a_{j-1}-1}(10)\tau^{a_1}\rho^{a_2}\dots\rho^{a_{j-2}-1}(01)(\tau^{a_1}\rho^{a_2}\dots\tau^{a_{j-1}-1}(10))^{a_j-1} & , \quad l=1\end{cases}\\ &= \begin{cases}\omega_0^{j-2}(\omega_1^{j-1})^{a_j} & , \quad l=0\\ \omega_1^{j-1}\omega_0^{j-2}(\omega_1^{j-1})^{a_j-1} & , \quad l=1\end{cases}\end{aligned}$$

While an odd j gives:

$$\begin{aligned}\omega_l^j &= \begin{cases}\tau^{a_1}\rho^{a_2}\dots\rho^{a_{j-1}}(010^{a_j-1}) & , \quad l=0\\ \tau^{a_1}\rho^{a_2}\dots\rho^{a_{j-1}}(10^{a_j}) & , \quad l=1\end{cases}\\ &= \begin{cases}\tau^{a_1}\rho^{a_2}\dots\rho^{a_{j-1}-1}(01)\tau^{a_1}\rho^{a_2}\dots\tau^{a_{j-2}-1}(10)(\tau^{a_1}\rho^{a_2}\dots\rho^{a_{j-1}-1}(01))^{a_j-1} & , \quad l=0\\ \tau^{a_1}\rho^{a_2}\dots\tau^{a_{j-2}-1}(10)(\tau^{a_1}\rho^{a_2}\dots\rho^{a_{j-1}-1}(01))^{a_j} & , \quad l=1\end{cases}\\ &= \begin{cases}\omega_0^{j-1}\omega_1^{j-2}(\omega_0^{j-1})^{a_j-1} & , \quad l=0\\ \omega_1^{j-2}(\omega_0^{j-1})^{a_j} & , \quad l=1\end{cases}\end{aligned}$$

□

We conclude this section by showing the relation diagram between ω_l^j for different choices of l.

Corollary 1.6. *For all $j \in \mathbb{N}_+$ the following diagram commutes*

$$\begin{array}{ccc}\omega_1^j & \xleftrightarrow{\quad\theta\quad} & \theta\omega_1^j\\ 01S^2 \big\uparrow\big\downarrow 10S^2 & & 10S^2 \big\uparrow\big\downarrow 01S^2\\ \omega_0^j & \xleftrightarrow{\quad\theta\quad} & \theta\omega_0^j\end{array}$$

Proof. The proof is a straightforward inductive application of the diagram shown in Example 1.4. □

2 Subshifts

We want to look at arbitrary concatenations of substitutions generating the words ω_l^j, as given by Definition 1.2 and therefore define

$$\begin{aligned}\mathcal{Q} := &\left\{\tau^{a_1}\rho^{a_2}\tau^{a_3}\dots\tau^{a_{j-1}}\rho^{a_j-1}\tau_{\mathrm{TM}} : \forall j \in 2\mathbb{N}_+ \text{ and } (a_i)_{i=1}^j \in \mathbb{N}_+^j \text{ with } a_1, a_j \geq 2\right\} \cup\\ &\left\{\tau^{a_1}\rho^{a_2}\tau^{a_3}\dots\rho^{a_{j-1}}\tau^{a_j-1}\tau_{\mathrm{TM}} : \forall j \in (2\mathbb{N}+1) \text{ and } (a_i)_{i=1}^j \in \mathbb{N}_+^j \text{ with } a_1, a_j \geq 2\right\}.\end{aligned}$$

Note that as a consequence the Morse-Substitution $\tau_{\mathrm{TM}}^{\mathbb{N}} \notin \mathcal{Q}$.

Exercise 2.1. Make yourself familar with the Morse-Substitution, also called Thue-Morse-Substitution. It is adviced to study at least two of the following sources [1, 2, 3] for a minimum of one hour.

Remark 2.2. Any sequence $(a_i)_{i\in\mathbb{N}} \in \mathbb{N}_+^{\mathbb{N}_+}$ induces a continued fraction expansion for an irrational number $x = [0; a_1, a_2, \ldots] \in [0, 1]$. For finite continued fractions remember $[0; a_1, \ldots, a_n, 1] = [0; a_1, \ldots, a_n + 1]$ and in order to prevent uniqueness we always require the last continued fraction entry to be ≥ 2. This number x is approximated by $\frac{p_n}{q_n} = [0; a_1, a_2, \ldots, a_n]$ for every $n \in \mathbb{N}$ and especially

$$(\omega_1^n)_k = \mathbb{1}_{[0,\frac{p_n}{q_n})}\left(k\frac{p_n}{q_n} \mod 1\right) = \mathbb{1}_{[0,p_n-1]}(kp_n \mod q_n),$$

for $k \in \{0, \ldots, q_n - 1\}$ and $a_n \geq 2$. Note whenever $a_n = a_{n+1} = 1$ we can still define an approximand of x by the former equality of continued fractions. Further by $\omega_0^n = 01S^2\omega_1^n$, it follows immediately

$$(\omega_0^n)_k = \mathbb{1}_{(0,\frac{p_n}{q_n}]}\left(k\frac{p_n}{q_n} \mod 1\right) = \mathbb{1}_{[1,p_n]}(kp_n \mod q_n) = \mathbb{1}_{[0,p_n-1]}(kp_n - 1 \mod q_n),$$

for $k \in \{0, \ldots, q_n - 1\}$. In the special case $x = [0; a_1]$,

$$(\omega_1^n)_k = \mathbb{1}_{[0,0]}(k \mod a_n) = 10^{a_1-1} = \tau^{a_1-1}(10).$$

As another example take $7/16 = [0; 2, 3, 2]$, then

$$\tau^1\rho^3\tau^1(10) = 1001010100101010 = (\mathbb{1}_{[0,7)}(7k \mod 16))_{k\in\mathbb{Z}_{16}}.$$

We will further only consider $x \in [0, 1/2)$, $|\omega_1^n| = |\omega_0^n| = q_n$.

Exercise 2.3. Write a program in your favorite language, but not in $\mathscr{L}(\mathscr{Q})$, that is capable of generating arbitrary long prefixes of fixpoints associated to periodic application of elements of $\mathscr{Q}$.

Tip: In matlab you may want to use 'strrep'.

Processing of the Exercise 2.3. We decided to use matlab as our only programming language. In matlab we implemented two functions, which are able to generate prefixes of length n for a given input string. The first function 16 lets you generate the prefix of a certain element of $\mathscr{Q}$ requiring the requested length of the associated fixpoint, the sequence of numbers to map and the sequence (a_n) to represent the element of $\mathscr{Q}$ as input whilst the second function 17 generates the prefix of a random concatenation of elements of $\mathscr{Q}$ by giving the requested length of the associated fixpoint, the sequence of numbers to map and three integres to limit the possible elements of $\mathscr{Q}$.

2.1 Primitive substitutions

Definition 2.4. A substitution $\sigma\colon \mathcal{A} \to \mathcal{A}^*$ is called *primitive* if for all $a, b \in \mathcal{A}$ exists an $k \in \mathbb{N}$ such that b is a letter of $\sigma^k(a)$. It is of *constant length*, if it exists a $q \in \mathbb{N}_+$ for all $a \in \mathcal{A}$ such that $|\sigma(a)| = q$.

Lemma 2.5. *Any $\sigma \in \mathscr{Q}$ is a primitive substitution of constant length.*

Proof. Let $\sigma \in \mathscr{Q}$. Consider $\sigma(0) = \tilde{\sigma}(\tau_{\mathrm{TM}}(0)) = \tilde{\sigma}(0)\tilde{\sigma}(1)$, where $\sigma = \tilde{\sigma} \circ \tau_{\mathrm{TM}}$. If j is even, than we use ρ^{a_j-1}, where $a_j \geq 2$ so we get from Example 1.4

$$\tilde{\sigma}'(\rho^{a_j-1}(01)) = \tilde{\sigma}'(01^{a_j}) \Rightarrow 1 \in \sigma(0). \tag{1}$$

Where $\tilde{\sigma'}$ is defined by $\tilde{\sigma} = \tilde{\sigma'} \circ \rho^{a_j - 1}$. This implies the fact that $1 \in \sigma(0)$ because $\tilde{\sigma'}$ is just a concatenation of ρ and τ, wich always keep the 1 in their image. With the same arguments it follows that

$$\tilde{\sigma'}(\tau^{a_j-1}(01)) = \tilde{\sigma'}(010^{a_j-1}) \Rightarrow 1 \in \sigma(0). \tag{2}$$

if j is odd. We get the same result for $\sigma(1)$, if we use Example 1.4 with (1) and (2) we get $0 \in \sigma(1)$.
This also gives us the constant length of σ as $|\sigma(1)| = |\sigma'(10)| = |\sigma'(01)| = |\sigma(0)|$. □

Exercise 2.6. Proof Lemma 2.5.

Lemma 2.7. *Let σ be a substitution of constant length q with a $u = u_0u_1u_2 \ldots \in \{0,1\}^{\mathbb{N}}$ such that $\sigma(u) = u$. Additionally there exists a letter a and a $k \leq q-1$ such that for all $b \in \mathcal{A}$, $\sigma(b)_k = a$. That is, there exists a column of a's in the substitution.*

With that each $u_{q^{m-1}k+l}$ denotes an ultimate periodicity of $q^m, m \in \mathbb{N}_+$, where $l \in \{\sum_{i=0}^{m-2} a_i q^i : a_i \in \{0,\ldots,q-1\}\backslash\{k\}\}$ has base q expansion without k's.

Note that if the set is empty, then $l := 0$.

Proof. As $\sigma(b)_k = a$ for any $b \in \mathcal{A}$ and $\sigma(u) = u$, we have $u_{qn+k} = a$ for any $n \in \mathbb{N}$. This implies that for all $m \in \mathbb{N}$ the words $\sigma^m(u_{qn+k})$ are the same. As σ is of constant length q, their distance to each other is q^{m+1} and their first known occurence is at $q^m k$. Hence each letter $u_{q^m k+l}$ for $l \in \{\sum_{i=0}^{m-1} a_i q^i : a_i \in \{0,\ldots,q-1\}\}$ has period q^{m+1}. But these also include all letters with smaller periods we know of. As these are originated from a at position k, we can exclude them by forbidding any occurence of k in the base q expansion of l. □

Corollary 2.8. *Let σ be a substitution of constant length q with a $u = u_0u_1u_2 \ldots \in \{0,1\}^{\mathbb{N}}$ such that $\sigma(u) = u$. Additionally there exists a letter a , a divisor p of q and a $h \in \{0,\ldots,\frac{q}{p}-1\}$ such that for all $b \in \mathcal{A}$, $\sigma(b)_{\frac{nq}{p}-(h+1)} = a$ for all $n \in \{1,\ldots,p\}$.*

With that each $u_{q^{m-1}\left(\frac{q}{p}-(h+1)\right)+l}$ denotes an ultimate periodicity of $\frac{q^m}{p}, m \in \mathbb{N}_+$, where $l \in \{\sum_{i=0}^{m-2} a_i q^i : a_i \in \{0,\ldots,q-1\}\backslash\{\frac{q}{p}-(h+1), \frac{2q}{p}-(h+1), \ldots, q-(h+1)\}\}$

Proof. As for all $b \in \mathcal{A}$,$n \in \{1,\ldots,p\}$: $\sigma(b)_{\frac{nq}{p}-(h+1)} = a$ and $\sigma(u) = u$, we have $u_{\frac{q}{p}(n+1)-(h+1)} = a$ for any $n \in \mathbb{N}$. This implies that for all $m \in \mathbb{N}$ the words $\sigma^m(u_{\frac{q}{p}(n+1)-(h+1)}))$ are the same for each $n \in \mathbb{N}$. As σ is of constant length q, their distance to each other is $\frac{q^{m+1}}{p}$ and their first known occurence is at $\left(\frac{q}{p}-(h+1)\right)q^m$. Hence each letter $u_{\left(\frac{q}{p}-(h+1)\right)q^m+l}$ for $l \in \{\sum_{i=0}^{m-1} a_i q^i : a_i \in \{0,\ldots,q-1\}\}$ has period $\frac{q^{m+1}}{p}$. But these also include all letters with smaller periods we know of. We can exclude them by forbidding any occurence of $\frac{nq}{p}-(h+1), n \in \{1,\ldots,p\}$ in the base q expansion of l. □

Exercise 2.9. Write down the prefix of length 100 of $u = \lim_{n\to\infty} \sigma^n(0)$ of $\sigma = \tau\tau_{\mathrm{TM}}$ and verify 2.7.

Processing of the Exercise 2.9. The following listing shows the prefix of length 100 generated by 16. In this case we have to generate a prefix of length 100 and have to begin with zero. As mapping we have $\sigma = \tau\tau_{TM}$, which implies we have to use [2] as sequence in generator3, because j is odd and we have $\tau^{2-1}\tau_{TM} = \tau\tau_{TM}$. This generates the following string 1.
From Lemma 2.7 we know, that there should be for each $u_{q^{m-1}k+l}$ an ultimate periodicity of $q^m, m \in \mathbb{N}_+$, where $l \in \{\sum_{i=0}^{m-2} a_i q^i : a_i \in \{0,\ldots,q-1\}\backslash\{k\}\}$ has base q expansion

without k's. As the map σ is of constant length 3 with column at σ_2 every third element of this string is a red marked zero, which denotes the periodicity of three. The periodicity of nine is in contrast to the periodicity of three a block 01 starting at σ_6. This block you see with a black underline.

Listing 1: Prefix of length hundred

```
>> Generator3(100,'0',[2])

ans =

    '0101001010100010010010100101'
    '0100010010010100101001001010'
    '0010010100101010001001010010'
    '1000101010001001001010001010'
```

To check the periodicity of 27 it is better to generate a string of length 104. With Lemma 2.7 we see, that we have two blocks of two elements 01 and 10 starting at $\sigma_{18,19}$ and $\sigma_{21,22}$. This time we want to show this periodicity with a matlab function. You can see in listing 20. We consider the block $\sigma_{18,19,20,21,22}$ as the missing σ_{20} is already known to have the ultimate periodicity of 3.

Listing 2: Periodicity of length 27

```
>> periodicity(19,23)
ans =     0     1     0     1     0

>> periodicity(46,50)
ans =     0     1     0     1     0

>> periodicity(73,77)
ans =     0     1     0     1     0

>> periodicity(100,104)
ans =     0     1     0     1     0
```

Listing 3: Periodicity of length 27

```
>> Generator3(104,'0',[2])

ans =

    '0101000101000100101010100'
    '1010001001001010001010101'
    '0010010100010100010010101'
    '0001010001001001010001001010'
```

Corollary 2.10. *For every $l \in \mathbb{N}$, which does not have any k in its base q expansion, that is also $l \in \{\sum_{i=0}^{\infty} a_i q^i : a_i \in \{0,\dots,q-1\}\backslash\{k\}\}$, we have that u_l is not equal to any of the letters described by Lemma 2.7. In a prefix of u of length q^m there are $(q-1)^m$ many of these, where $m \in \mathbb{N}_+$.*

Remark 2.11. Note that it is not that easy to show that there exists no periodicity. Take for example $\tau_{\mathrm{TM}}(a) = ab, \tau_{\mathrm{TM}}(b) = ba$, which has no periodicity. While $\sigma(a) := aba$, $\sigma(b) := bab$ generates the sequence $abababababababababab\ldots$, which is indeed 2-periodic.

Exercise 2.12. The convolution with respect to a Borel-measure μ is a well defined abelian bilinear operator from $L^1_\mu(G)\times L^1_\mu(G) \to L^1_\mu(G)$ given by $(f*g)(y) = \int f(x)g(y-x)d\mu(x)$.

For $(G,\mu) \in \{(\mathbb{R},\lambda),(\mathbb{Z},\delta_{\mathbb{Z}})\}$, the averaged convolution with respect to μ is defined for bounded Borel-measurable functions $f,g\colon G\to\mathbb{C}$, whenever for all $x\in G$ the limit

$$\lim_{N\to\infty}\frac{1}{\mu([-N,N])}(f\mathbb{1}_{[-N,N]})*(g\mathbb{1}_{[-N,N]})\,(x),$$

exists. In this case we also write $f\circledast g\,(x)$.

i) Proof that on the set $X := \{(f,g) : (f\circledast g)\ \textit{exists}\}$ the averaged convolution shares the properties of convolution. I.e. for $(f,g),(h,g)\in X$ and a constant a we have $(af+h)\circledast g$ exists and $f\circledast g = g\circledast f$.

ii) Show for a bounded Borel-measurable function h and $f\in L^1_\mu(G)$ that $h\circledast f = 0$.

iii) For $G=\mathbb{Z}$ show $\mathbb{1}_{\mathbb{Z}}\circledast\mathbb{1}_{\mathbb{Z}} = \mathbb{1}_{\mathbb{Z}}$.

Processing of the Exercise 2.12. First we want to remark, that $L^1_\mu(G)$ is a vector space with $(L^1_\mu(G),+,\cdot)$ and a algebra with $(L^1_\mu(G),*)$.

i) For $(f,g),(h,g)\in X$ and a constant a we have

$$\begin{aligned}
a(f\circledast g)+(h\circledast g)(x) =& a\lim_{N\to\infty}\frac{1}{2N}(f\mathbb{1}_{[-N,N]})*(g\mathbb{1}_{[-N,N]})(x)\\
&+\lim_{N\to\infty}\frac{1}{2N}(h\mathbb{1}_{[-N,N]})*(g\mathbb{1}_{[-N,N]})(x)\\
=&\lim_{N\to\infty}\frac{1}{2N}(af\mathbb{1}_{[-N,N]})*(g\mathbb{1}_{[-N,N]})(x)\\
&+\lim_{N\to\infty}\frac{1}{2N}(h\mathbb{1}_{[-N,N]})*(g\mathbb{1}_{[-N,N]})(x)\\
=&\lim_{N\to\infty}\frac{1}{2N}((af\mathbb{1}_{[-N,N]})*(g\mathbb{1}_{[-N,N]})(x)\\
&+(h\mathbb{1}_{[-N,N]})*(g\mathbb{1}_{[-N,N]})(x))\\
=&\lim_{N\to\infty}\frac{1}{2N}(((af\mathbb{1}_{[-N,N]})+(h\mathbb{1}_{[-N,N]}))*(g\mathbb{1}_{[-N,N]})(x))\\
=&((af+h)\circledast g)(x).
\end{aligned}$$

Next on we check if $(e,f),(g,h)\in X$, their combined limit can be taken. For that fix an $x\in G$ and set

$$a_n := \frac{(e\mathbb{1}_{[-n,n]})*(f\mathbb{1}_{[-n,n]})\,(x)}{\mu([-n,n])},\qquad b_n := \frac{(g\mathbb{1}_{[-n,n]})*(h\mathbb{1}_{[-n,n]})\,(x)}{\mu([-n,n])},\qquad n\in\mathbb{N}_+$$

By choice $\lim_{n\to\infty}a_n =: a$ and $\lim_{n\to\infty}b_n =: b$ exists, therefore if we take any $\varepsilon>0$ it exists an $N\in\mathbb{N}$ such that $a_n-\varepsilon\le a\le a_n+\varepsilon$ and $b_n-\varepsilon\le b\le b_n+\varepsilon$ for all $n\ge N$. But then $a_n+b_n-2\varepsilon\le a+b\le a_n+b_n+2\varepsilon$, hence $\lim_{n\to\infty}a_n+b_n = a+b$ and we can use the properties of convolution for each $n\in\mathbb{N}$, while n approaches infinity.

ii) Let h be a bounded measurable function and $f \in L^1_\mu(G)$, then

$$\begin{aligned}(h \circledast f)(x) &= \lim_{N\to\infty} \frac{1}{2N}(h\mathbb{1}_{[-N,N]}) * (f\mathbb{1}_{[-N,N]})(x) \\ &\leq \lim_{N\to\infty} \frac{1}{2N}(\sup(h)\mathbb{1}_{[-N,N]}) * (f\mathbb{1}_{[-N,N]})(x) = 0.\end{aligned}$$

The last expression equals zero, because the convolution is bounded and the integral of f is finite, therewith the whole expression is finite and $\lim_{N\to\infty} \frac{1}{2N} = 0$.

iii) Let $G = \mathbb{Z}$.

$$\begin{aligned}\mathbb{1}_{\mathbb{Z}} \circledast \mathbb{1}_{\mathbb{Z}} &= \lim_{N\to\infty} \frac{1}{2N}(\mathbb{1}_{\mathbb{Z}}\mathbb{1}_{[-N,N]}) * (\mathbb{1}_{\mathbb{Z}}\mathbb{1}_{[-N,N]})(x) \\ &= \lim_{N\to\infty} \frac{1}{2N}(\mathbb{1}_{\mathbb{Z}\cap[-N,N]}) * (\mathbb{1}_{\mathbb{Z}\cap[-N,N]})(x) \\ &= \lim_{N\to\infty} \frac{1}{2N} \sum_{k\in\mathbb{Z}} \mathbb{1}_{\mathbb{Z}\cap[-N,N]}(k)\mathbb{1}_{\mathbb{Z}\cap[-N,N]}(x-k) \\ &= \lim_{N\to\infty} \frac{1}{2N} \sum_{k\in\mathbb{Z}\cap[-N,N]} \mathbb{1}_{\mathbb{Z}\cap[-N,N]}(x-k) = \mathbb{1}_{\mathbb{Z}}.\end{aligned}$$

The last two equations are true, because $\mathbb{1}_{\mathbb{Z}\cap[-N,N]}(k) = 1$ for $k \in \mathbb{Z} \cap [-N, N]$ and $\mathbb{1}_{\mathbb{Z}\cap[-N,N]}(x-k) = 1$, if $x - k \in \mathbb{Z} \cap [-N, N]$ and if $x - k \in \mathbb{Z}\backslash[-N, N]$ then $\mathbb{1}_{\mathbb{Z}\cap[-N,N]}(x-k) = 0$. For N to infinity $x - k$ is an element of $\mathbb{Z}$, so we have one as solution.

Lemma 2.13. *Let $g\colon \mathbb{N} \to \mathbb{N}$ be a bounded function and let $u\colon \mathbb{N} \to \mathbb{N}$, $l \mapsto 1$ if $l \in \left\{\sum_{i\in I} a_i q^i : (a_i)_{i\in I} \in (\{0,\dots,q-1\}\backslash\{k\})^*\right\}$ and 0 otherwise. Then*

$$|g| \circledast u = \lim_{N\to\infty} \frac{(|g|\mathbb{1}_{[-N,N]}) * (u\mathbb{1}_{[-N,N]})}{2N} = 0.$$

Proof. Let $z \in \mathbb{Z}$, then

$$\begin{aligned}(|g|\mathbb{1}_{[-N,N]}) * (u\mathbb{1}_{[-N,N]})(z) &= \sum_{x\in\mathbb{Z}} \mathbb{1}_{[-N,N]}(z-x)\mathbb{1}_{[-N,N]}(x)\,|g|(z-x)u(x) \\ &\leq \max_{\mathbb{Z}} |g| \sum_{x\in\mathbb{Z}} \mathbb{1}_{[-N,N]}(x)\,u(x) \\ &= (\max_{\mathbb{Z}} |g|)\,|\{x \in [-N,N] : u(x) \neq 0\}|\,.\end{aligned}$$

For prefixes of length $N = q^m$ we have $|\{x \in [-N,N] : u(x) \neq 0\}| = (q-1)^m$, which implies the result. Even if we consider subsequences, an upper bound for them is given by

$$\frac{(q-1)^m + (q-1)^{m+1}}{q^m + (q-1)^{m+1}} \leq q\frac{(q-1)^m}{q^m} = q\left(\frac{q-1}{q}\right)^m \to 0.$$

□

Remark 2.14. An interesting class of convolutions is given by $\{f * \widetilde{f} : f \in L^1_\mu\}$, where $\widetilde{f}(x) \coloneqq \overline{f(-x)}$, as every such convolution is positive definite. A function $g\colon G \to \mathbb{C}$ is called *posititve definite*, if for all $N \in \mathbb{N}$

$$\sum_{0\le i,j\le N} c_i \overline{c_j} g(x_i - x_j) \ge 0,$$

where $x_i \in G$, $c_i \in \mathbb{C}$ for all $0 \le i \le N$.

Corollary 2.15. *Let $u\colon \mathbb{N} \to \{0,1\}$ be such that $u \circledast \widetilde{u}$ exists. Fix a $q \ge 2, 0 \le k \le q-1$ and define $u_{\hat{k}}\colon \mathbb{N} \to \{0,1\}$ to be $l \mapsto u(l)$, if l has a base q expansion with respect to $(\{1,\dots,q-1\}\backslash\{k\})^*$ and 0 otherwise. Finally set $u_k \coloneqq u - u_{\hat{k}}$. Then*

$$u \circledast \widetilde{u} = u_k \circledast \widetilde{u_k}.$$

Proof. Notice that there is literally no difference in the proof of Lemma 2.13 for $\widetilde{u_{\hat{k}}}$. The remaining part follows from bilinearity

$$\begin{aligned} u \circledast \widetilde{u} =& (u_k + u_{\hat{k}}) \circledast (\widetilde{u_k} + \widetilde{u_{\hat{k}}}) \\ =& u_k \circledast \widetilde{u_k} + u_k \circledast \widetilde{u_{\hat{k}}} + u_{\hat{k}} \circledast \widetilde{u_k} + u_{\hat{k}} \circledast \widetilde{u_{\hat{k}}} \\ =& u_k \circledast \widetilde{u_k} + 0 + 0 + 0. \end{aligned}$$

□

Lemma 2.16. *Let $k, l, c, d \in \mathbb{Z}$. The following averaged convolution*

$$\mathbb{1}_{k\mathbb{Z}+c} \circledast \widetilde{\mathbb{1}_{l\mathbb{Z}+d}} = \frac{1}{\operatorname{lcm}(k,l)} \mathbb{1}_{\gcd(k,l)\mathbb{Z}+(c-d)} \tag{3}$$

and especially exists.

Proof. The proof is essentially an application of the chinese remainder theorem. Let $z \in \mathbb{Z}$, then

$$\begin{aligned} (\mathbb{1}_{k\mathbb{Z}+c}\mathbb{1}_{[-N,N]}) * (\widetilde{\mathbb{1}_{l\mathbb{Z}+d}\mathbb{1}_{[-N,N]}})(z) &= \sum_{x\in\mathbb{Z}} \mathbb{1}_{k\mathbb{Z}+c}(z-x)\mathbb{1}_{[-N,N]}(z-x)\, \widetilde{\mathbb{1}_{l\mathbb{Z}+d}}(x)\mathbb{1}_{[-N,N]}(x) \\ &= |(k\mathbb{Z} + (z-c)) \cap (l\mathbb{Z} - d) \cap ([-N,N] + z) \cap [-N,N])|\,. \end{aligned}$$

The first part is solved by any $x \in \mathbb{Z}$ s.t. $x = z-c \mod k$, $x = l-d \mod l$. The solution space is $\operatorname{lcm}(k,l)\mathbb{Z}+(z-c+d)$, if $(z-c+d) = 0 \mod \gcd(k,l)$ and $\emptyset$ otherwise. Another way to see this is by the equality $|(k\mathbb{Z} + (z-c)) \cap (l\mathbb{Z} - d)| = |(k\mathbb{Z} + (z-c+d)) \cap l\mathbb{Z}|$.

Now suppose $(z-c+d) = 0 \mod \gcd(k,l)$, as otherwise the first part is empty. With that

$$\begin{aligned} &\lim_{N\to\infty} \frac{1}{2N} (\mathbb{1}_{k\mathbb{Z}+c}\mathbb{1}_{[-N,N]}) * (\widetilde{\mathbb{1}_{l\mathbb{Z}+d}\mathbb{1}_{[-N,N]}})(z) \\ &\qquad = \lim_{N\to\infty} \frac{|(\operatorname{lcm}(k,l)\mathbb{Z} + (z-c+d)) \cap ([-N,N]+z) \cap [-N,N])|}{2N} \\ &\qquad = \lim_{N\to\infty} \frac{|(\operatorname{lcm}(k,l)\mathbb{Z}) \cap [-N,N]|}{2N} = \frac{1}{\operatorname{lcm}(k,l)}. \end{aligned}$$

□

Exercise 2.17. Include averaged convolution for two sequences into your program and adapt it to heuristically verify Lemmas 2.13 and 2.16 and Corollary 2.15. Most programming languages have already a predefined convolution that could be used. For Corollary 2.15 and Lemma 2.16 it is considered enough to do so for the constant 1-sequence and the ones given by Lemma 2.16. Judging from this exercise you may also consider $u = \sum_{i=0}^{n} \mathbb{1}_{k_i\mathbb{Z}+c_i}$ for $k_i, c_i \in \mathbb{N}$ for all $0 \leq i \leq n$ and check if $u \circledast \widetilde{u} = u_k \circledast \widetilde{u_k}$ for some $k \in \mathbb{N}$.

Processing of the Exercise 2.17. In the exercise above we had to include the average convolution and verify Lemmas 2.13 and 2.16 and Corollary 2.15. To verify Lemma 2.13 we implemented 21. This needs the functions 22, 23, 26 and 24 explained in the attachment and returns the maximum value of the average convolution from u and g to show that this converges. In 4 below we calculated our results for $q = 4$ and $k = 2$. We used $g \equiv 1$ as the worst case of a bounded function.

In the listing 4 you can see that the average convolution converges to zero for length $x \to \infty$. Here the length was extended in increments of four to have a compareable value to the estimation used in the proof of Lemma 2.13. It occurs to be exact the expected value $\left(\frac{q-1}{q}\right)^m \cdot \frac{1}{2}$.

Listing 4: Verify Lemma 2.13

```
>> Lemma213Test(4^2,2,4)
ans =      0.2813

>> Lemma213Test(4^3,2,4)
ans =      0.2109

>> Lemma213Test(4^4,2,4)
ans =      0.1582

>> Lemma213Test(4^5,2,4)
ans =      0.1187

>> Lemma213Test(4^6,2,4)
ans =      0.0890

>> Lemma213Test(4^7,2,4)
ans =      0.0667

>> Lemma213Test(4^8,2,4)
ans =      0.0501

>> Lemma213Test(4^9,2,4)
ans =      0.0375

>> Lemma213Test(4^10,2,4)
ans =      0.0282

>> Lemma213Test(4^11,2,4)
ans =      0.0211

```

```
>> Lemma213Test(4^12,2,4)
ans =      0.0158
```

In Corollary 2.15 we had to show, that $u \circledast \widetilde{u} = u_k \circledast \widetilde{u_k}$. Therefore we used listing 27. In this function we also need listing 28, which realize for a given u the image of $u_{\hat{k}}$, with listing 29 you can expand the image values of a function from $\mathbb{N}$ to $\mathbb{Z}$ and with listing 30 you can mirror the image values. The last part in this function is the average convolution of all functions of u with listing 31 and then compare both sides. As result we calculated the sum of the differences, the elementwise maximum difference as well as the average difference.

Listing 5: Verify Corollary 2.15

```
>> [sum max avg]=Corollar215Test(4^3,Generator3(4^3,'1'
    ,[2 2]),3,4)
sum =     1.2891 max =     0.0625 avg =     0.0201

>> [sum max avg]=Corollar215Test(4^3,Generator3(4^4,'1'
    ,[2 2]),3,4)
sum =     1.2129 max =     0.0586 avg =     0.0190

>> [sum max avg]=Corollar215Test(4^3,Generator3(4^5,'1'
    ,[2 2]),3,4)
sum =     0.9624 max =     0.0483 avg =     0.0150

>> [sum max avg]=Corollar215Test(4^3,Generator3(4^6,'1'
    ,[2 2]),3,4)
sum =     0.7269 max =     0.0369 avg =     0.0114

>> [sum max avg]=Corollar215Test(4^3,Generator3(4^7,'1'
    ,[2 2]),3,4)
sum =     0.5477 max =     0.0277 avg =     0.0086

>> [sum max avg]=Corollar215Test(4^3,Generator3(4^8,'1'
    ,[2 2]),3,4)
sum =     0.4109 max =     0.0208 avg =     0.0064

>> [sum max avg]=Corollar215Test(4^3,Generator3(4^9,'1'
    ,[2 2]),3,4)
sum =     0.3074 max =     0.0157 avg =     0.0048

>> [sum max avg]=Corollar215Test(4^3,Generator3(4^10,'1'
    ,[2 2]),3,4)
sum =     0.2306 max =     0.0118 avg =     0.0036
```

Here we build a function with the generator class, which length also was extended in increments of four. We started with 1 and the map $\tau^2\rho\tau_{\mathrm{TM}}$. The meaning of $k = 3$ and $q = 4$ is the same we already mentioned in listing 4.

In Lemma 2.16 we want to show the equation (3). As already shown in the listing before we want to consider the difference between both convolutions, this is done by listing 32. In this function we have six input parameters, the first parameter represents

the length of the functions and the second is the length on that the difference between both functions shall be evaluated. With all other parameters we build the one functions $\mathbb{1}_{k\mathbb{Z}+c}$ and $\mathbb{1}_{l\mathbb{Z}+d}$ from Lemma 2.16, which you can see in listing 33. As already in the Corollary 2.15 before we use listing 31 as average convolution on $\mathbb{Z}$ and with the functions $\mathbb{1}_{4\mathbb{Z}+3}$ and $\mathbb{1}_{6\mathbb{Z}+4}$ we get

Listing 6: Verify Lemma 2.16

```
>> [sum max avg]=Lemma216Test(100,100,4,3,6,4)
sum =     0.5483  max =     0.0233  avg =     0.0055

>> [sum max avg]=Lemma216Test(1000,100,4,3,6,4)
sum =     0.0548  max =     0.0023  avg =     5.4833e-04

>> [sum max avg]=Lemma216Test(10000,100,4,3,6,4)
sum =     0.0055  max =     2.3333e-04
avg =     5.4833e-05

>> [sum max avg]=Lemma216Test(100000,100,4,3,6,4)
sum =     5.4833e-04  max =     2.3333e-05
avg =     5.4833e-06

>> [sum max avg]=Lemma216Test(1000000,100,4,3,6,4)
sum =     5.4833e-05  max =     2.3333e-06
avg =     5.4833e-07
```

Definition 2.18. Define $A_n := q^{n-1}k + \{\sum_{i=0}^{n-2} a_i q^i : a_i \in \{0,\dots,q-1\}\backslash\{k\}\}$ to be the set of q^n-periodic points . $|A_n| = (q-1)^{n-1}$

Further set $A_{n,1} := \{x \in A_n : u(x) = 1\}$ and define $A_{n,0}$ respectively. Hence $A_n = A_{n,1} \uplus A_{n,0}$

With that we can define

$$u_{k,N} := \sum_{n=1}^{N} \sum_{x\in A_{n,1}} \mathbb{1}_{q^n\mathbb{Z}+x} = u_{k,N-1} + \sum_{x\in A_{N,1}} \mathbb{1}_{q^N\mathbb{Z}+x}.$$

Remark 2.19. Notice that $u_{k,N} \to u_k$ pointwise for $N \to \infty$. We will now check convergence for the convolution.

Lemma 2.20.

$$u_{k,N} \circledast \widetilde{u_{k,N}} \to u_k \circledast \widetilde{u_k} = u \circledast \widetilde{u},$$

uniformly in N.

Proof. Pointwise convergence is already clear. To check uniform convergence, we first obtain for any $x \in A_{N,1}$

$$u_{k,N} \circledast \widetilde{\mathbb{1}_{q^N\mathbb{Z}+x}} = \sum_{n=1}^{N} \sum_{x_a\in A_{n,1}} \left(\mathbb{1}_{q^n\mathbb{Z}+x_a} \circledast \widetilde{\mathbb{1}_{q^N\mathbb{Z}+x}}\right) = \sum_{n=1}^{N} \sum_{x_a\in A_{n,1}} q^{-N}\mathbb{1}_{q^n\mathbb{Z}+(x_a-x)}.$$

The next step will be to calculate an upper bound for this function. As by definition of A_n the family of sets $(q^n\mathbb{Z} + x_a)_{x_a \in A_n}$ is disjoint, there can be at most N non-zero-evaluations of characteristic function at the same time. Another way to describe this is by $\max\left\{|\{x_a \in \mathbb{Z} : \exists 1 \leq n \leq N, x_a \in A_n : x_a = x_0 + x \mod q^n\}| : 0 \leq x_0 \leq q^N - 1\right\} \leq N$, as $A_n \hookrightarrow \mathbb{Z}_{q^n}$ for all $n \in \mathbb{N}$.

If we consider every $x \in A_N$, remember that $|A_n| = (q-1)^n$, this gives an upper bound by

$$\begin{aligned}
|u_{k,N}| \circledast |\widetilde{u_{k,N}}| =& \left(|u_{k,N-1}| + \sum_{x \in A_{N,1}} \mathbb{1}_{q^N\mathbb{Z}+x}\right) \circledast \left(|\widetilde{u_{k,N-1}}| + \sum_{x \in A_{N,1}} \widetilde{\mathbb{1}_{q^N\mathbb{Z}+x}}\right) \\
=& |u_{k,N-1}| \circledast |\widetilde{u_{k,N-1}}| + u_{k,N-1} \circledast \sum_{x \in A_{N,1}} \widetilde{\mathbb{1}_{q^N\mathbb{Z}+x}} \\
&+ \sum_{x \in A_{N,1}} \mathbb{1}_{q^N\mathbb{Z}+x} \circledast \widetilde{u_{k,N-1}} + \sum_{x \in A_{N,1}} \mathbb{1}_{q^N\mathbb{Z}+x} \circledast \sum_{x \in A_{N,1}} \widetilde{\mathbb{1}_{q^N\mathbb{Z}+x}} \\
\leq& |u_{k,N-1}| \circledast |\widetilde{u_{k,N-1}}| + N \sum_{x \in A_{N,1}} q^{-N} + N \sum_{x \in A_N} q^{-N} + \sum_{x \in A_N} q^{-N} \\
\leq& |u_{k,N-1}| \circledast |\widetilde{u_{k,N-1}}| + (2N+1)\left(\frac{q-1}{q}\right)^N.
\end{aligned}$$

Hence the convergence of $u_{k,N-1} \circledast \widetilde{u_{k,N-1}} \to u_k \circledast \widetilde{u_k}$ is uniformly. The last equality follows from Corollary 2.15 □

Exercise 2.21. Each $\sigma \in \mathscr{Q}$ is primitive with the property $\sigma(a)_0 = a$ for all $a \in \{0,1\}$ and therefore $u := \lim_{n\to\infty} \sigma^n(0)$ is well defined. Check, that this is indeed the case by showing that $\sigma^{n-1}(a)$ is a prefix of $\sigma^n(a)$ for all $n \in \mathbb{N}$. Make your progam able to handle $u_{k,N} \circledast \widetilde{u_{k,N}}$ and heuristically verify that for a substitution in $\mathscr{Q}$ with two different columns, hence k_1, k_2 and $k_1 \neq k_2$, of only one letter the limits are the same. Do not forget trivial checks like the constant one or constant zero sequences.

Processing of the Exercise 2.21.

Lemma 2.22. *For each $\sigma \in \mathscr{Q}$ with the properties of 2.21 $\sigma^{n-1}(a)$ is a prefix of $\sigma^n(a)$.*

Proof. This prove is done by induction. But first we want to remark, that

$$\begin{aligned}
\sigma(a) &= \sigma(a)_0\sigma(a)_1 \dots \sigma(a)_{|\sigma(a)|-1} \\
\sigma(a) &= a\tilde{\sigma}(a).
\end{aligned} \tag{4}$$

IA: For $n = 2$ we have $\sigma^1(a) = a\tilde{\sigma}(a)$ and this is a prefix of (4).
IV: $\sigma^{n-1}(a)$ is a prefix of $\sigma^n(a)$, so we have $\sigma^n(a) = \sigma^{n-1}(a)\tilde{\sigma}^{n-1}(a)$.
IS:

$$\begin{aligned}
\sigma^{n+1}(a) &= \sigma(\sigma^n(a)) \\
&\overset{\text{IV}}{=} \sigma(\sigma^{n-1}(a)\tilde{\sigma}^{n-1}(a)) \\
&= \sigma^n(a)\tilde{\sigma}^n(a).
\end{aligned}$$

□

To verify Exercise 2.21 we show that for a given map u $u_{k_1} \circledast u_{k_1} = u_{k_2} \circledast u_{k_2}$ for $k_1, k_2, k_1 \neq k_2$ two columns of the map u. To prove this for a given map u we implemented 34. As examples we calculated the stated mappings and analised the boxed columns.

$$\begin{array}{ll} \tau^2\tau_{TM}(0): & 0\ 1\ \boxed{0}\ \boxed{0} \\ \tau^2\tau_{TM}(1): & 1\ 0\ \boxed{0}\ \boxed{0} \end{array} \qquad \begin{array}{ll} \tau^2\rho\tau_{TM}(0): & 0\ 1\ \boxed{0}\ 0\ \boxed{1}\ 0\ 0 \\ \tau^2\rho\tau_{TM}(1): & 1\ 0\ \boxed{0}\ 0\ \boxed{1}\ 0\ 0 \end{array}$$

$$\begin{array}{ll} \tau^2\rho^2\tau\tau_{TM}(0): & 0\ 1\ 0\ 0\ \boxed{1}\ 0\ 0\ \boxed{1}\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0 \\ \tau^2\rho^2\tau\tau_{TM}(1): & 1\ 0\ 0\ 0\ \boxed{1}\ 0\ 0\ \boxed{1}\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0 \end{array}$$

Listing 7: Exercise 2.21 $\tau^2\tau_{TM}$

```
Comparing the convolutions on 64 elements.

>> [sum max avg]=Exercise221Test(Generator3(4^4,'0',[3])
    ,4^3,2,3,4)
sum =     0.1953 max =     0.0117 avg =    0.0031

>> [sum max avg]=Exercise221Test(Generator3(4^5,'0',[3])
    ,4^3,2,3,4)
sum =     0.1602 max =     0.0083 avg =    0.0025

>> [sum max avg]=Exercise221Test(Generator3(4^6,'0',[3])
    ,4^3,2,3,4)
sum =     0.1235 max =     0.0065 avg =    0.0019

>> [sum max avg]=Exercise221Test(Generator3(4^7,'0',[3])
    ,4^3,2,3,4)
sum =     0.0935 max =     0.0049 avg =    0.0015

>> [sum max avg]=Exercise221Test(Generator3(4^8,'0',[3])
    ,4^3,2,3,4)
sum =     0.0703 max =     0.0037 avg =    0.0011

>> [sum max avg]=Exercise221Test(Generator3(4^9,'0',[3])
    ,4^3,2,3,4)
sum =     0.0528 max =     0.0028 avg =    8.2517e-04

>> [sum max avg]=Exercise221Test(Generator3(4^10,'0',[3])
    ,4^3,2,3,4)
sum =     0.0396 max =     0.0021 avg =    6.1908e-04


Comparing the convolutions on 256 elements.

>> [sum max avg]=Exercise221Test(Generator3(4^4,'0',[3])
    ,4^4,2,3,4)
sum =     0.7266 max =     0.0156 avg =    0.0028
```

```
29
30 >> [sum max avg]=Exercise221Test(Generator3(4^5,'0',[3])
       ,4^4,2,3,4)
31 sum =     0.6934 max =     0.0122 avg =    0.0027
32
33 >> [sum max avg]=Exercise221Test(Generator3(4^6,'0',[3])
       ,4^4,2,3,4)
34 sum =     0.5684 max =     0.0092 avg =    0.0022
35
36 >> [sum max avg]=Exercise221Test(Generator3(4^7,'0',[3])
       ,4^4,2,3,4)
37 sum =     0.4384 max =     0.0069 avg =    0.0017
38
39 >> [sum max avg]=Exercise221Test(Generator3(4^8,'0',[3])
       ,4^4,2,3,4)
40 sum =     0.3318 max =     0.0052 avg =    0.0013
41
42 >> [sum max avg]=Exercise221Test(Generator3(4^9,'0',[3])
       ,4^4,2,3,4)
43 sum =     0.2496 max =     0.0039 avg =    9.7498e-04
44
45 >> [sum max avg]=Exercise221Test(Generator3(4^10,'0',[3])
       ,4^4,2,3,4)
46 sum =     0.1874 max =     0.0030 avg =    7.3197e-04
```

Listing 8: Exercise 2.21 $\tau^2\rho\tau_{TM}$

```
1  Comparing the convolutions on 49 elements.
2
3  >> [sum max avg]=Exercise221Test(Generator3(7^3,'0',[2
       2]),7^2,2,4,7)
4  sum =     0.5685 max =     0.0627 avg =    0.0116
5
6  >> [sum max avg]=Exercise221Test(Generator3(7^4,'0',[2
       2]),7^2,2,4,7)
7  sum =     0.5085 max =     0.0552 avg =    0.0104
8
9  >> [sum max avg]=Exercise221Test(Generator3(7^5,'0',[2
       2]),7^2,2,4,7)
10 sum =     0.4389 max =     0.0475 avg =    0.0090
11
12 >> [sum max avg]=Exercise221Test(Generator3(7^6,'0',[2
       2]),7^2,2,4,7)
13 sum =     0.3767 max =     0.0408 avg =    0.0077
14
15 >> [sum max avg]=Exercise221Test(Generator3(7^7,'0',[2
       2]),7^2,2,4,7)
16 sum =     0.3229 max =     0.0349 avg =    0.0066
17
18 Comparing the convolutions on 343 elements.
```

```
19
20 >> [sum max avg]=Exercise221Test(Generator3(7^3,'0',[2
       2]),7^3,3,5,7)
21 sum =     3.5000 max =     0.0627 avg =    0.0102
22
23 >> [sum max avg]=Exercise221Test(Generator3(7^4,'0',[2
       2]),7^3,3,5,7)
24 sum =     4.0556 max =     0.0575 avg =    0.0118
25
26 >> [sum max avg]=Exercise221Test(Generator3(7^5,'0',[2
       2]),7^3,3,5,7)
27 sum =     3.6270 max =     0.0510 avg =    0.0106
28
29 >> [sum max avg]=Exercise221Test(Generator3(7^6,'0',[2
       2]),7^3,3,5,7)
30 sum =     3.1304 max =     0.0442 avg =    0.0091
31
32 >> [sum max avg]=Exercise221Test(Generator3(7^7,'0',[2
       2]),7^3,3,5,7)
33 sum =     2.6863 max =     0.0379 avg =    0.0078
```

Listing 9: Exercise 2.21 $\tau^2\rho^2\tau\tau_{TM}$

```
1  Comparing the convolutions on 169 elements.
2
3  >> [sum max avg]=Exercise221Test(Generator3(17^2,'0',[2 2
       2]),17^2,4,7,17)
4  sum =     0.0588 max =     0.0017 avg =    2.0354e-04
5
6  >> [sum max avg]=Exercise221Test(Generator3(17^3,'0',[2 2
       2]),17^2,4,7,17)
7  sum =     0.0501 max =     0.0017 avg =    1.7326e-04
8
9  >> [sum max avg]=Exercise221Test(Generator3(17^4,'0',[2 2
       2]),17^2,4,7,17)
10 sum =     0.0506 max =     0.0016 avg =    1.7525e-04
11
12 >> [sum max avg]=Exercise221Test(Generator3(17^5,'0',[2 2
       2]),17^2,4,7,17)
13 sum =     0.0479 max =     0.0015 avg =    1.6565e-04
14
15 Comparing the convolutions on 2197 elements.
16
17 >> [sum max avg]=Exercise221Test(Generator3(17^3,'0',[2 2
       2]),17^3,4,7,17)
18 sum =     2.2595 max =     0.0026 avg =    4.5991e-04
19
20 >> [sum max avg]=Exercise221Test(Generator3(17^4,'0',[2 2
       2]),17^3,4,7,17)
21 sum =     3.0327 max =     0.0031 avg =    6.1729e-04
```

```
22
23 >> [sum max avg]=Exercise221Test(Generator3(17^5,'0',[2 2
       2]),17^3,4,7,17)
24 sum =     2.9365 max =     0.0031 avg =    5.9770e-04
```

2.2 Analysis of $\mathscr{Q}$

The previous analysis just gives a very rough description of $\mathscr{Q}$, which would also hold for a much more general class of substitutions. Therefore the first aim is to look at finite compositions of $\sigma \in \mathscr{Q}$. For example one can take two substitutions $\tau\tau_{\mathrm{TM}}$, $\tau\rho\tau_{\mathrm{TM}}$ and see in which way $\tau\tau_{\mathrm{TM}}\tau\rho\tau_{\mathrm{TM}}$ differs from $\tau\rho\tau_{\mathrm{TM}}\tau\tau_{\mathrm{TM}}$. Tough they are not in $\mathscr{Q}$ it might also be fruitful to consider τ_{TM} and $\rho\tau_{\mathrm{TM}}$, especially the first substitution as it partakes in every element of $\mathscr{Q}$.

Exercise 2.23. Can one check heuristically that for periodic applications of substitutions generated from $\mathscr{Q}$ Remark 2.11 does not apply. Namely that for such substitutions only one letter colums generate periodicities.

Processing of the Exercise 2.23. To verify this we implemented the function 37 to generate all possible Elements of $\mathscr{Q}$ so the partial sum of $(a_n)_{n\in\mathbb{N}}$ adds up to n. Furthermore we implemented the function 38 that checks if a given map generates new periods thats aren't given by Lemma 2.7 and used this in 36 where we investigated each possible concatenation of elements of $\mathscr{Q}$ on periodicites. With this we found out about some more periodicities that we classified in Corollary 2.8. These could be seen as the first counterexample of the given task but as those periodicities can be seen as easy as the ones of Lemma 2.7 we decided to also implement these in 38 as known periodicities that are no longer of interest. Ignoring these periodicities we actually found counterexamples that met our interests. One of these is given by the concatenation

$$\begin{aligned}\tau^2\rho\tau_{TM}\circ\tau^2\tau_{TM}(0): &\quad 0100100100010010010010010100\\ \tau^2\rho\tau_{TM}\circ\tau^2\tau_{TM}(1): &\quad 1000100010010001001000100100\end{aligned}$$

This map with a constant length $q = 28$ generates for $u_{21,22}$ the ultimate periodicity 49.

Listing 10: Test if u_{21} has period 49

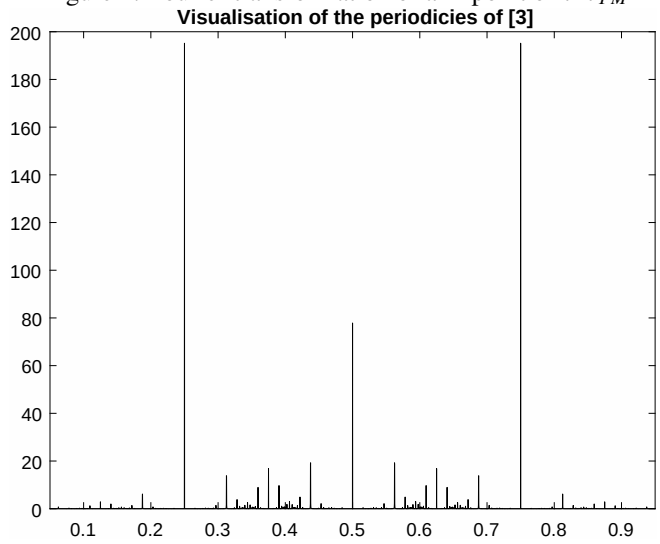
```
1
2 periodic(Generator3(100000000,'1',[3 0;2 2]),22,49)
3
4 ans =     1
```

The responsible blocks of columns are given by choosing $p = 4$ and $h \in \{0, 2\}$ in Corollary 2.8(Namely the columns at position 7,14,21 and 28 for $h = 0$ as well as 5,12,19 and 26 for $h = 2$) as well as the columns at position 15,16,22,23. Therefore this marks the first map from $\mathscr{Q}$ we found generating a periodicity generated by different blocks of columns that assort well.

Listing 11: Visualisation of the periodicity of $u_{21,22}$

```
1          '0100100100010001001000100100'
2          '1000100010010001001000100100'
```

Figure 1: Fourier transformation of a fixpoint of $\tau^2\tau_{TM}$




```
'0100100100010001001000100100'
'0100100100010001001000100100'
'1000100010010001001000100100'
'0100100100010001001000100100'
'0100100100010001001000100100'
'1000100010010001001000100100'
'0100100100010001001000100100'
'0100100100010001001000100100'
'0100100100010001001000100100'
'1000100010010001001000100100'
'0100100100010001001000100100'
'0100100100010001001000100100'
```

Exercise 2.24. Play around with $\mathscr{Q}$ and formulate at least one conjecture which is heuristically supported by your program or made into a theorem by being proven.

Processing of the Exercise 2.24. To conclude this worksheet we visualized the fourier transform $R_N(t) = \frac{1}{N}\left|\sum_{n\leq N} u_n e^{2\pi i n t}\right|^2$(See [3] Chapter 4.3) of fixpoints of the maps $\tau^2\tau_{TM}, \tau^2\rho\tau_{TM}$ and $\tau^2\rho^2\tau\tau_{TM}$ we already analized in Exercise 2.21. These can be seen in 1, 2 and 3. This gives an indicator about all periodicies of u as there will be peaks for every period p at positions $\frac{k}{p}, 0 \leq k < p$ varying in size depending on the size of the period p.

Given this we also used $R_N(t)$ to visualize our approximation of $u \circledast \widetilde{u}$ and $u_k \circledast \widetilde{u_k}$ in 4, 5 and 6. Here $u \circledast \widetilde{u}$ is represented by the orange function whilst $u_k \circledast \widetilde{u_k}$ is given by the blue function.

Figure 2: Fourier transformation of a fixpoint of $\tau^2\rho\tau_{TM}$

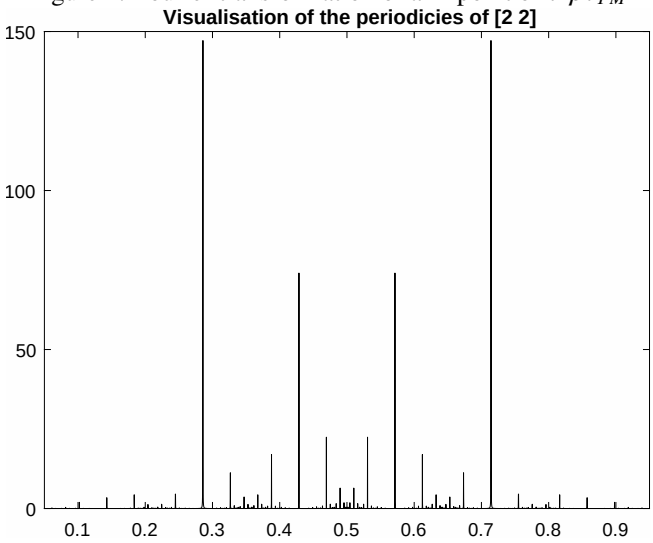


Figure 3: Fourier transformation of a fixpoint of $\tau^2\rho^2\tau\tau_{TM}$

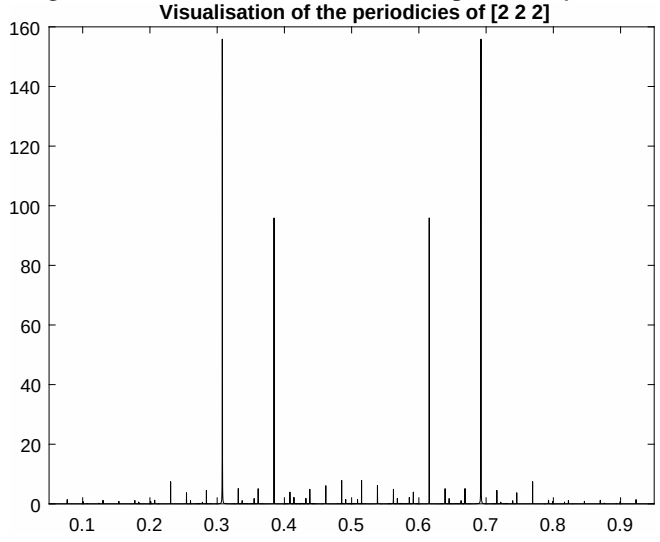


Figure 4: Autokorrelation of $u \circledast \widetilde{u}$ and $u_k \circledast \widetilde{u_k}$, length=$2 * 4^3$

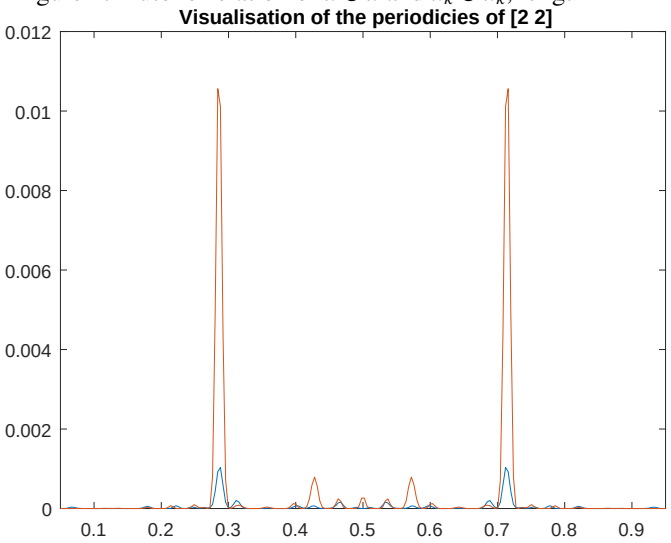


Figure 5: Autokorrelation of $u \circledast \widetilde{u}$ and $u_k \circledast \widetilde{u_k}$, length=$2 * 4^4$

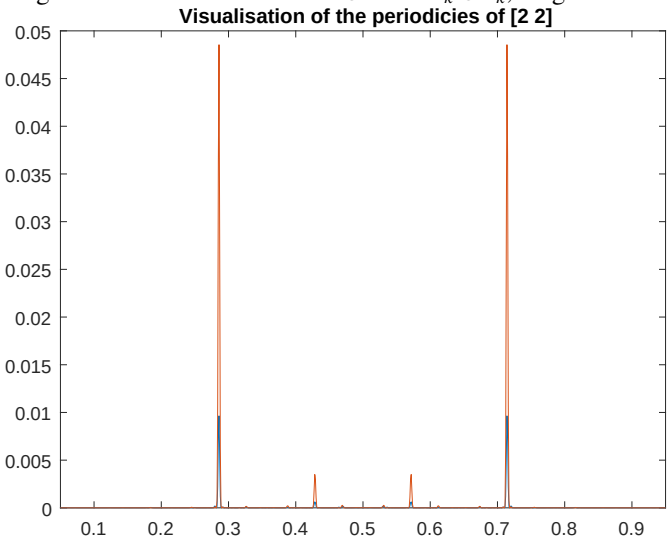
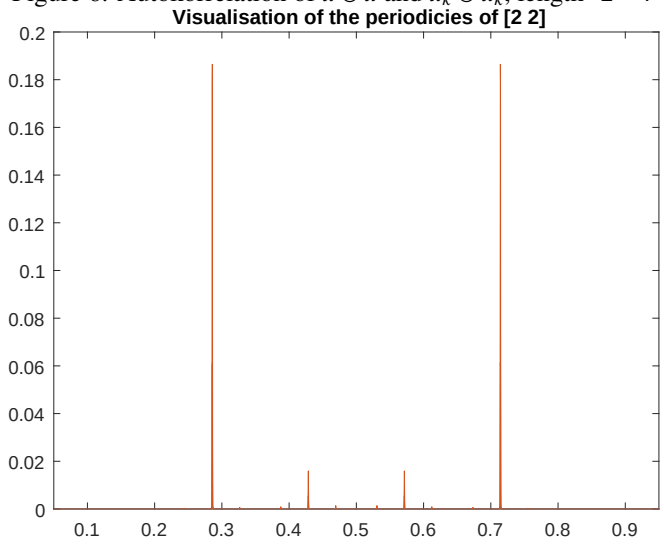


Figure 6: Autokorrelation of $u \circledast \widetilde{u}$ and $u_k \circledast \widetilde{u_k}$, length=$2 * 4^5$



References

[1] M. Baake and U. Grimm. *Aperiodic Order*. Cambridge University Press. CPI Group Ltd, Croydon, CR0 4YY, 2013.

[2] N. Pytheas Fogg. *Substitutions in Dynamics, Arithmetics and Combinatorics*. Lecture Notes in Mathematics. Springer-Verlag Berlin Heidelberg, 2002.

[3] M. Queffélec. *Substitution Dynamical Systems - Spectral Analysis*. Lecture Notes in Mathematics. Springer-Verlag Berlin Heidelberg, 2010.

Attachment

Listing 12: τ

```
% Realises the map tau.
function s = tau2(x)
s=strrep(x,'1','10');
end
```

Listing 13: ρ

```
% Realises the map rho.
function s = rho2(x)
s=strrep(x,'0','01');
end
```

Listing 14: θ

```
% Realises the map theta.
function s = theta2(x)
s=strrep(x,'0','a');
s=strrep(s,'1','0');
s=strrep(s,'a','1');
end
```

Listing 15: τ_{TM}

```
% Realises the map ttm.
function s = ttm2(x)
s=strrep(x,'0','a');
s=strrep(s,'1','b');
s=strrep(s,'a','01');
s=strrep(s,'b','10');
end
```

Listing 16: Generate prefix of a given mapping of $\mathscr{Q}$

```
%%Generator3(n,x,A)
%%Generates the prefix of an element of \mathscr{Q}.
%@return the prefix of a given element from \mathscr{Q}
    with the length n.
%@n the length of the requested prefix.
%@x the inserted string to generate the prefix.
%@A the finite sequel (a_n) that represent the element of
     \mathscr{Q}.
function prefix = Generator3(n,x,A)
B=size(A);
if ischar(x)~= true
     throw (MException('Component:InputError','Incorrect
         input format x char needed'))
end
```

```
12 if isempty(x)
13     throw (MException('Component:InputError','Incorrect
           input format x is empty'))
14 end
15 if isfloat(n)~= true
16     throw (MException('Component:InputError','Incorrect
           input format n float needed'))
17 end
18 if isfloat(A)~= true
19     throw (MException('Component:InputError','Incorrect
           input format A matrix of float needed'))
20 end
21 for i=1:B(1)
22     if A(i,1)< 2 || A(i,find(A(i,:),1,'last'))<2
23         throw (MException('Component:LanguageError','
               Incorrect input. first or last a_i not 2 or
               bigger'))
24     end
25 end
26 pref=x;
27 while length(pref)<n
28     pref=omega2(pref,A);
29 end
30 prefix=pref(1:n);
31 end
```

Listing 17: Generate prefix of a random mapping of $\mathscr{Q}$

```
1  %%Generator2(n,x,amount,long,max)
2  %%Generates the prefix of an random element of \mathscr{Q
       }.
3  %@return the prefix of a given element from \mathscr{Q}
       with the length n.
4  %@n the length of the requested prefix.
5  %@x the inserted string to generate the prefix.
6  %@amount the amount of random elements of \mathscr{Q}
       that shall be
7  %concatenated.
8  %@long the maximal possible length of the sequence (a_n)
       for each mapping.
9  %@max an upper bound for each element of (a_n).
10 %See also func(amount,long,max). It is advised to choose
       each of these
11 %entries \leq 5 as the calculation time is of exponential
        growth
12 %and will take a while above these number.
13 function prefix =Generator2(n,x,amount,long,max)
14 A=Func(amount,long,max);
15 B=size(A);
16 if ischar(x)~= true
```

```
    throw (MException('Component:InputError','Incorrect
        input format x char needed'))
end
if isfloat(n)~= true
    throw (MException('Component:InputError','Incorrect
        input format n float needed'))
end
if isfloat(amount)~= true || isfloat(long)~= true ||
    isfloat(max)~= true
    throw (MException('Component:InputError','Incorrect
        input format a,b,c integer needed'))
end
for i=1:B(1)
    if A(i,1)< 2 || A(i,find(A(i,:),1,'last'))<2
        throw (MException('Component:LanguageError','
            Incorrect input. first or last a_i not 2 or
            bigger'))
    end
end
pref=x;
while length(pref)<n
    pref=omega2(pref,A);
end
prefix=pref(1:n);
end
```

Listing 18: Realization of the Elements of $\mathscr{Q}$

```
%%omega(x,A)
%%Realizes the map of Q given by A upon x.
function o = omega(x,A)
B=size(A);
o=x;
for j=1:B(1)
    o=ttm2(o);
    if (-1)^length(A(j,:))==1
        for i=1:(A(j,end)-1)
            o=rho2(o);
        end
    elseif (-1)^length(A(j,:))==-1
        for i=1:(A(j,end)-1)
            o=tau2(o);
        end
    end
    for i=1:(length(A(j,:))-1)
        if (-1)^(length(A(j,:))-i)==1
            for k=1:A(j,length(A(j,:))-i)
                o=rho2(o);
            end
        elseif (-1)^(length(A(j,:))-i)==-1
```

```
                for k=1:A(j,length(A(j,:))-i)
                    o=tau2(o);
                end
            end

        end
    end
end
```

Listing 19: Generates a random matrix to represent a map of $\mathscr{Q}$

```
%%Func( amount,length,max )
%%Generates a random matrix to represent a family of
    mappings of
%%\mathscr{Q} that shall be concatenated
%@return matrix of the size (amount x length)
% @amount the amount of rows the matrix shall have.
% @length the amount of colums the matrix shall have.
% @max the maximal possible value for each entry of the
    matrix A.
function [ A ] = Func( amount,length,max )
A=zeros(amount,length);
for i=1:amount
    A(i,1)=randi([2,max]);
    r=randi([1,length]);
    for j=2:r-1
        A(i,j)=randi(max);
    end
    if r~=1
        A(i,r)=randi([2,max]);
    end

end

end
```

Listing 20: periodicity test

```
%@a,b the block you want to consider of the given
%function
function [ block ] = periodicity(a,b)
blockarray=str2num(replace(Generator3(104,'0',[2]),'',' '
    ));
block=blockarray(a:b);
end
```

Listing 21: Testfunction for Lemma 2.13

```
%%Lemma212Test(n,k,q)
%%A function used to verify Lemma 2.13. The function g is
    represented by the
```

```
%%worst case the constant one-function.
%@return returns the highest value of the convolution of
    u
%@n the towards infinity diverging length of the
    convolution.
%@q the base q to realise the q expansion.
%@k the element 0\leq k\leq q-1 that is not allowed in
    the base q
%expansion.
function [ out ] = Lemma213Test(n,k,q)
out=max(AConvAufN(n,DarstUAufN(n,k,q),one(n)));
end
```

Listing 22: Average convolution on $\mathbb{N}$

```
%%AConvAufN(n,fix1,fix2)
%%Calculates the average convolution of two given
    functions.
%@return returns the average colvolution values of two
    given functions of
%the size 2*n.
%@n the towards infinity diverging length of the
    convolution.
%@fix1 @fix2 the maps of the two functions that shall be
    convoluted.
function [ c ] = AConvAufN(n,fix1,fix2)
fix11=replace(fix1(1:n),'',' ');
fix21=replace(fix2(1:n),'',' ');
c=1/(2*n).*conv(str2num(fix11),str2num(fix21));
end
```

Listing 23: Representation of map u from Lemma 2.13

```
%%DarstUAufN(n,k,q)
%Realization of the map u from Lemma 2.13 that checks a
    natural number
%of its q base expansion without k.
%@return returns tha mapping of u of the first n natural
    numbers.
%@n the amount of consecutive natural numbers that is of
    interest.
%(counting 0 as first natural number)
%@q the base q to realise the q expansion.
%@k the element 0\leq k\leq q-1 that is not allowed in
    the base q
%expansion.
function [ out ] =DarstUAufN(n,k,q)
out=one(n);
A=QadicBaseExpansion(n,q);
for i=1:size(A,1)
```

```
    for j=1:find(A(i,:),1,'last')
        if A(i,j)==k
            out(i)='0';
        end
    end
end
end
```

Listing 24: Realisation of the constant one-function

```
%%one(n)
%%Representation of the constant one-function.
%@n length of the function.
function [out] = one(n)
out=replace(num2str(ones([1 n])),' ','');
end
```

Listing 25: Realisation of the constant zero-function

```
%%zero(n)
%%Representation of the constant zero-function.
%@n length of the function.
function [ out ] = zero( n )
out=replace(num2str(zeros([1 n])),' ','');
end
```

Listing 26: Illustration of quadic base

```
%%QadicBaseExpansion(x,q)
%%Calculates the q-adic base expansion of the first x
    consecutive natural
%%numbers.
%@n the amount of consecutive natural numbers that is of
    interest.
%(counting 0 as first natural number)
%@q the base q to realise the q expansion.
function [A,m] = QadicBaseExpansion(n,q)
m=0;
x=0;
while q^m<n
    x=x+(q-1)*q^m;
    m=m+1;
end
if (x+2)>n
    m=m-1;
end
A=zeros(n,m+1);
for i=1:n
    s=i-1;
    for j=1:m+1
```

```
            A(i,m+2-j)=floor(s/q^(m+1-j));
            s=s-floor(s/q^(m+1-j))*q^(m+1-j);
        end
    end
end
```

Listing 27: Testfunction for corollar 2.15

```
%%Corollar215Test( int n(>0),char Array fixu(length(fixu)
    =n),int
%%k(0<=k<=q-1),int q(>=2))
%A function used to verify corollar 2.15
%@return returns (in this order) the sum, max and average
     difference between
%the convolutions.
%First parameter n represents the length of the functions
     that
%shall be compared.
%Second parameter realises the image of a given map u.
%Parameter k and q are needed to create the function u_\
    hat{k}.
function [ out1,out2, out3] = Corollar215Test(n,fixu,k,q)
fixucaret=DarstU_kCaret(fixu,k,q);
fixu=FuncExpandToZ(fixu);
fixucaret=FuncExpandToZ(fixucaret);
fixuk=replace(num2str(fixu-fixucaret),' ','');
fixusim=ComplexMirror(fixu);
fixuksim=ComplexMirror(fixuk);
convfixu=AConvAufZ(round(length(fixu)/2),fixu,round(
    length(fixu)/2),fixusim,round(length(fixu)/2));
convfixuk=AConvAufZ(round(length(fixu)/2),fixuk,round(
    length(fixu)/2),fixuksim,round(length(fixu)/2));
out1=sum(abs(convfixu(length(fixu)-floor(n/2):length(fixu
    )+round(n/2))-convfixuk(length(fixu)-floor(n/2):length
    (fixu)+round(n/2))));
out2=max(abs(convfixu(length(fixu)-floor(n/2):length(fixu
    )+round(n/2))-convfixuk(length(fixu)-floor(n/2):length
    (fixu)+round(n/2))));
out3=out1/n;
end
```

Listing 28: Realization of $u_{\hat{k}}$

```
%%DarstU_kCaret(fixu,k,q)
%%Realizes the map u_\hat{k} for a given map u.
%@fixu the mapping of the map u.
%@q the base q to realise the q expansion.
%@k the element 0\leq k\leq q-1 that is not allowed in
    the base q
%expansion.
```

```
function [ out] = DarstU_kCaret( fixu ,k,q )
mapu_k=DarstUAufN(length(fixu),k,q);
for i=1:length(mapu_k)
    if mapu_k(i)== '1'
        mapu_k(i)= fixu(i);
    end
end
out=mapu_k;
end
```

Listing 29: Expanding a function to $\mathbb{Z}$

```
%%FuncExpandToZ(func1)
%%Symmetrically expands a function defined on N  to Z
    mapping all negativ
%%numbers on 0.
%@func1 the mapping of a map that shall be expanded.
%@return  returns the expanded function.
function [ out ] = FuncExpandToZ(func1)
out=zero(length(func1)-1);
out=strcat(out,func1);
end
```

Listing 30: complexe mirror a function

```
%%ComplexeMirror(func1)
%%Maps each entry at position x onto -conj(x).As we only
    observe real
%%integers this function only flips a given map.
%@return returns \overset{\sim}{func1}
%@func1 the map to be complexe mirrored.
function [ out ] = ComplexMirror(func1)
out=fliplr(func1);
end
```

Listing 31: Average convolution on $\mathbb{Z}$

```
%AConvAufZ(n,fix1,start1,fix2,start2)
%%Calculates the average convolution of two given
    functions.
%@return returns the average colvolution values of two
    given functions of
%the size 4*n.
%@n the towards infinity diverging length of the
    convolution.
%@fix1 @fix2 the maps of the two functions that shall be
    convoluted.
%@start1 @start2 the position of the map of 0 in each of
    the given
%functions.
```

```
function [ c ] = AConvAufZ(n, fix1,start1,fix2,start2 )
while start1-n<1 || start1+n>length(fix1)
    fix1=strcat('0',fix1,'0');
    start1=start1+1;
end
while start2-n<1 || start2+n>length(fix2)
    fix2=strcat('0',fix2,'0');
    start2=start2+1;
end
fix11=fix1(start1-n:start1+n);
fix21=fix2(start2-n:start2+n);
fix12=replace(fix11,'',' ');
fix22=replace(fix21,'',' ');
c=1/(2*n).*conv(str2num(fix12),str2num(fix22));

end
```

Listing 32: Testfunction to verify Lemma 2.16

```
%%lemma216Test( int n(>0),int m(>0),int k(>0),int c,intl
    (>0),int d)
%%A function used to verify lemma 2.16
%@return return (in this order) the sum, max and average
%of elementwise differences between the convoluted
%functions and the expected function.
%@n First parameter n represents the length of the
    functions that
%shall be constructed.It is known as the faktor N
    diverging towards
%infinity to realise the averaged convolution.
%@m Second parameter m is the length on that the
    difference between the two
%functions of lemma 2.16 shall be evaluated.
%Since average convolution is a limit value process it is
     recommended to
%choose m much smaller than n.
%@k @c @l @d The parameters k,c,l,d represent the needed
    parameters for the
%given functions in lemma 2.16.
function [ out1, out2,out3 ] = Lemma216Test(n,m,k,c,l,d )
[kcind,kcindstart]=Builder(2*n,k,c);
[ldind,ldindstart]=Builder(2*n,l,d);
[klcdind,klcdindstart]=Builder(2*n,gcd(k,l),c-d);
klcdind=1/lcm(k,l).*str2num(replace(klcdind,'',' '));
con=AConvAufZ(n,kcind,kcindstart,ComplexMirror(ldind),
    ldindstart);
out1=sum(abs(klcdind(klcdindstart-floor(m/2):klcdindstart
    +round(m/2))-con(floor(length(con)/2)-floor(m/2):floor
    (length(con)/2)+round(m/2))));
out2=max(abs(klcdind(klcdindstart-floor(m/2):klcdindstart
```

```
    +round(m/2))-con(floor(length(con)/2)-floor(m/2):floor
    (length(con)/2)+round(m/2))));
out3=out1/m;
end
```

Listing 33: building a one-function on given modulus and remainder

```
%%Builder( n,modulo,rest )
%%Builds a one-functions defined on modulo*Z+rest
%@n the length of the function.It will be build
    symmetrically around the 0
%giving negative numbers the priority.
%@modulo the modulus of the function.
%@rest the remainder of the function.
function [out,start] = Builder( n,modulo,rest )
if rest<=0
    rest=rest+floor(abs(rest)/modulo)*modulo+modulo;
else
    rest=rest-floor(rest/modulo)*modulo;
end
a=zero(modulo);
a(rest+1)='1';
start=1;
out='';
while length(out)<n+2
    out=strcat(a,out,a);
    start=start+modulo;
end
if mod(n,2)==0
    out=out(start-n/2:start+n/2-1);
else
    out=out(start-floor(n/2):start+floor(n/2));
end
start=n/2+1;
end
```

Listing 34: Testfunction to check the difference between the convolution of two colums

```
%%Exercise221Test(fixu,m,k1,k2,q)
%%A function to test the idea of 2.21
%@return returns the sum of the elementwise difference of
    the
%two observed convolutions.
%@fixu the map of u.
%@m Second parameter m is the length on that the
    difference between the two
%convolutions shall be evaluated.
%Since average convolution is a limit value process it is
    recommended to
%choose m much smaller than length(fixu).
```

```
%@q the constant length of the given map u.
%@k1 @k2 the position of the columns.
function [ out1,out2, out3] = Exercise221Test(fixu,m,k1,
    k2,q)
U_k1local=FuncExpandToZ(U_kAufNGivenU(fixu,k1,q));
U_k2local=FuncExpandToZ(U_kAufNGivenU(fixu,k2,q));
a=AConvAufZ(length(fixu),U_k1local,length(fixu),
    ComplexMirror(U_k1local),length(fixu));
b=AConvAufZ(length(fixu),U_k2local,length(fixu),
    ComplexMirror(U_k2local),length(fixu));
out=a(round(length(a)/2)-floor(m/2):round(length(a)/2)+
    round(m/2))-b(round(length(b)/2)-floor(m/2):round(
    length(b)/2)+round(m/2));
out1=sum(abs(out));
out2=max(abs(out));
out3=out1/m;
end
```

Listing 35: A function that calculates u_k

```
%%U_kAufNGivenU(fixu,k,q)
%%A function that calculates u_k given a k and the map u.
%@fixu the map of u.
%@q the base q needed to calculate the map of u_kCaret.
%@k the coefficent not allowed in the base expansion in
    u_kCaret.
function [ out ] = U_kAufNGivenU(fixu,k,q)
out=replace(num2str(fixu-DarstU_kCaret(fixu,k,q)),' ','')
    ;

end
```

Listing 36: Testfunction to check if the periodic application generates periodicity

```
%%Exercise223Test(n1,n2,m,x)
%%For a given integer n>=2 and a string x (for example
    '0' or '1')
%%calculates if periodic applications(for now only
    alternation of two elements)
%of elements of Q generate periodicies.
%@n1 the length on wich the periodicy shall be tested.
%@n2 the sum that the elements of the finite sequence (
    a_n) maximally
%add up to.(This is essential for the calculation time)
    minimum is 2.
%@m users definition of periodicity giving the amount of
    consecutive
%identical outputs needed to be called periodic.
%@x the string to start with.
function Exercise223Test(n1,n2,m,x)
```

```
pause on;
A=cell(1,n2-1);
for i=2:n2
    A(i-1)={PossibilityGenerator(i)};
end
for k=2:n2
    B=cell2mat(A(k-1));
    B1=size(B);
    for i=1:B1(1)

        for j=2:n2
            B1=size(B);
            C=cell2mat(A(j-1));
            C1=size(C);
            if B1(2)<C1(2)
                B=cat(2,B,zeros(B1(1),C1(2)-B1(2)));
            else
                C=cat(2,C,zeros(C1(1),B1(2)-C1(2)));
            end
            C1=size(C);
            for h=1:C1(1)
                B2=num2str(B(i,:));
                B3=[];
                for l=1:C1(2)
                    B3=strcat(B3,B2(3*l-2));
                end
                C2=num2str(C(h,:));
                C3=[];
                for l=1:C1(2)
                    C3=strcat(C3,C2(3*l-2));
                end
                D=PeriodicityCheck(n1,m,x,[B(i,:);C(h,:)
                    ]);
                if isempty(D)
                    fprintf('The Elements %s and %s
                        generate no new periodicies\n',B3,
                        C3)
                else
                    fprintf('The Elements %s and %s
                        generate the new periodicies\n',B3
                        ,C3)
                    %disp(D)
                end
            end

        end
    end
end
end
```

Listing 37: A function to generate all possible elements of $\mathscr{Q}$ that fit inputs

```
%%PossibilityGenerator(n,fields)
%%Generates all possible elements of \mathscr{Q} given an input n
%@return returns a matrix containing each possible vector of real positiv
%integers that elements add up to n. In addition the first and last element
%of each vector are limited to be 2 or bigger.
%@n the sum that the elements of the finite sequence (a_n) add up to.
%@fields(optional) optional parameter that limits the maximum length of each
%vector.
function out1= PossibilityGenerator(n,fields)
if n==2
    out1=2;
else
    if nargin<2
        if n==2
            fields=1;
        else
            fields=n-2;
        end
    end
    maxfields=n-2;
    out1=[];
    maxValue=n-fields-1;
    while(1)
        if (isempty(out1)==1)
            out2=zeros(1,maxfields);
            out2(1)=2;
            for i=1:fields-1
                out2(i+1)=1;
            end
            out2(fields)=out2(fields)+n-fields-1;
        else
            out2=out1(B(1),:);
            lasttouch=find(out2,1,'last');
            while   (out2(lasttouch) == 1) ||(lasttouch==find(out2,1,'last') && out2(lasttouch)<=2)
                lasttouch=lasttouch-1;
            end
            if lasttouch<2
                break;
            end
            if ((out2(lasttouch)==maxValue) && (maxValue>2))
```

```
                    out2(lasttouch)=out2(lasttouch)-1;
                    out2(find(out2,1,'last'))=out2(find(out2
                        ,1,'last'))+1;
                end
                out2(lasttouch)=out2(lasttouch)-1;
                out2(lasttouch-1)=out2(lasttouch-1)+1;
            end
            out1=cat(1,out1,out2);
            B=size(out1);
        end;
        if fields>1
            out1=cat(1,out1,PossibilityGenerator(n,fields-1))
                ;
        end
end
end
```

Listing 38: A function to test a given map for new periodicies

```
%@n the length of the fixpoint whose periodicity shall be
    determined
%@m users definition of periodicity giving the amount of
    consecutive
%identical outputs needed to be called periodic.
function [ out ] = PeriodicityCheck(n,m,x,A)
func=Generator3(n,x,A);
func0=omega2('0',A);
func1=omega2('1',A);
columns=[];
out=[];
A=[];
%------------------generating the already known
    periodicity cases-----------------
for i=1:numel(func0)
    if strcmp(func0(i),func1(i))
        columns=[columns i-1];
    end
end
BaseExpansion=QadicBaseExpansion(n,numel(func0));
%------------------generating the periodicity cases from
    Corollary 2.8-------------------
for i=setdiff(divisors(numel(func0)),1) %
    for h=0:(numel(func0)/i)-1
        if sum(ismember((numel(func0)/i-h):(numel(func0)/
            i):numel(func0),columns+1))==i && periodic(
            func0,numel(func0)/i-h,numel(func0)/i)
            for j=2:length(BaseExpansion)
                const2=0;
                for k=1:find(BaseExpansion(j,:),1,'
                    last')-1
```

```
25                          if mod((BaseExpansion(j,k)+1+h),(
                                numel(func0)/i))==0
26                              const2=1;
27                          end
28                      end
29                      if BaseExpansion(j,find(BaseExpansion(
                            j,:),1,'last'))==((numel(func0)/i)
                            -1-h) && ~const2
30                          A=[A; j numel(func0).^find(
                                BaseExpansion(j,:),1,'last')/i
                                ];
31                      end
32                  end
33              end
34          end
35      end
36      %------------------generating the periodicity cases from
            Lemma 2.7-----------------
37      for i=2:length(BaseExpansion)
38          const1=sum(BaseExpansion(i,find(BaseExpansion(i,:),1,
                'last'))== columns);
39          if isempty(find(BaseExpansion(i,find(BaseExpansion(i
                ,:),1,'last'))== columns,1,'last'))
40              const3=0;
41          else
42              const3=columns(find(BaseExpansion(i,find(
                    BaseExpansion(i,:),1,'last'))== columns,1,'
                    last'));
43          end
44          const2=(sum(const3== BaseExpansion(i,1:(find(
                BaseExpansion(i,:),1,'last')-1)))==0);
45          if const1 && const2
46              A=[A; i numel(func0).^find(BaseExpansion(i,:),1,'
                    last')];
47          end
48      end
49      A=reduce(sortrows(A));
50      for i=1:size(A,1)
51          j=A(i,1);
52          while j<n-(A(i,2)+1)
53              A=[A;j+A(i,2) A(i,2)];
54              j=j+A(i,2);
55          end
56      end
57      A=reduce(sortrows(A));
58      %------------------calculating all new periodicity
            -----------------
59      for i=setdiff(1:n,A(:,1))
60          k=floor((n-i)/m);
61          for j=1:k
```

```
62          if periodic(func,i,j)
63              out=[out; i-1 j];
64          end
65      end
66  end
67
68  for i=1:size(A,1)
69       k=floor((n-A(i,1))/m);
70       h=A(i,2);
71       if i~=1 && A(i-1,1)==A(i,1)
72       C=union(C,h:h:k);
73       else
74       C=h:h:k;
75       end
76       if i== size(A,1) || A(i+1,1)~=A(i,1)
77          for j=setdiff(1:k,C)
78
79              if periodic(func,A(i,1),j)
80                  out=[out; A(i,1)-1 j];
81              end
82          end
83       end
84  end
85  out=sortrows(out);
86  %------------------generall periodicity check
        -----------------
87  % for i=1:n
88  %      k=floor((n-i)/m);
89  %      for j=1:k
90  %              if periodic(func,i,j)
91  %                  out=[out; i-1 j];
92  %              end
93  %      end
94  % end
95  %------------------function to check a certain
        periodicity----------------
96      function out1 = periodic(func, start, period)
97          posloc1=func(start);
98          posloc2=start;
99          while func(start)== posloc1 && posloc2<length(
                func)
100             posloc1=func(posloc2);
101             posloc2=posloc2+period;
102         end
103         if posloc2+period>length(func)
104             out1=1;
105         else
106             out1=0;
107         end
108     end
```

```
109 %Since 2018 there is a built-in function divisors(n) in
    matlab.
110 %For users with outdated versions we recommend this
    function we got from
111 %https://de.mathworks.com/matlabcentral/answers/21542-
    find-divisors-for-a-given-number
112     %     function d = divisors( n )
113     %         %DIVISORS(N) Returns array of divisors of n
114     %         if ~isscalar(n)
115     %             error('n must be a scalar');
116     %         end
117     %         if n < 1
118     %             error('n must be positive integer');
119     %         end
120     %         if n == 1
121     %             d = 1;
122     %             return;
123     %         end
124     %         f = factor(n);
125     %         pf = unique(f);
126     %         for a = 1:length(pf)
127     %             o(a) = sum(f == pf(a));
128     %         end
129     %         mi = zeros(size(o));
130     %         d = zeros(1,prod(o+1));
131     %         a = 1;
132     %         carry = 0;
133     %         while ~carry
134     %             d(a) = prod(pf.^mi);
135     %             a = a + 1;
136     %             if mi(1) < o(1)
137     %                 mi(1) = mi(1) + 1;
138     %             else
139     %                 mi(1) = 0;
140     %                 carry = 1;
141     %             end
142     %             for b = 2:length(o)
143     %                 if carry
144     %                     if mi(b) < o(b)
145     %                         mi(b) = mi(b) + 1;
146     %                         carry = 0;
147     %                     else
148     %                         mi(b) = 0;
149     %                         carry = 1;
150     %                     end
151     %                 end
152     %             end
153     %         end
154     %         d = sort(d);
155     %     end
```

```
156 end
```

Listing 39: A function to reduce a given matrix

```
1  %%reduce(A)
2  %Given a row-sorted Matrix A. Deletes all unnecessary
       rows of A.
3  %These are for the given purpose those that have same
       first entrie and
4  %a multiple second entrie of another row.
5  function [out] = reduce(A)
6  out=[];
7       while ~isempty(A)
8           i=A(1,1);
9           out2=[];
10          while ~isempty(A) && A(1,1)==i
11              out2=[out2; A(1,:)];
12              A(1,:)=[];
13          end
14          for j=1:size(out2,1)
15              for k=j+1:size(out2,1)
16                  if out2(j,1)~=0 && out2(k,1)~=0 && mod(out2
                        (k,2),out2(j,2))==0
17                      out2(k,:)=zeros(1,size(out2(k,:),2));
18                  end
19              end
20          end
21          out2(all(~out2,2),:)=[];
22          out=[out;out2];
23      end
24 end
```