Recursive and Periodic Sequences

How can we decide whether a sequence in a set is generated by a recursion formula of any depth? This text presents some quite elementary general results for this question and an algorithm that finds the shortest feedback shift register that generates a given finite sequence.

1 Periods of Infinite and Finite Sequences

Infinite Sequences

Let M be a set. An infinite sequence $x = (x_0, x_1, \ldots) = (x_i)_{i\in\mathbb{N}} \in M^{\mathbb{N}}$ is called **periodic** if there is an index m and an integer $n \geq 1$ such that

(1) $$x_{i+n} = x_i \quad \text{for all } i \geq m.$$

The smallest index m for which there is an n such that (1) holds is called the **preperiod** of the sequence x. Denote it by μ. Then the smallest index n such that (1) holds with $m = \mu$ is called the **period** ν of x.

Lemma 1 *Let $x \in M^{\mathbb{N}}$ be a periodic sequence with preperiod μ and period ν. Let m and $n \geq 1$ be indices such that* (1) *holds. Then $m \geq \mu$ and $\nu \mid n$.*

Proof. $m \geq \mu$ by the definition of the preperiod. Let $n = p\nu + q$ with $p \geq 0$ and $1 \leq q \leq \nu$. Thus

$$x_j = x_{j+n} = x_{j+p\nu+q} = x_{j+q} \quad \text{for all } j \geq m$$

because ν is the period and $m \geq \mu$. Let $i \geq \mu$ arbitrary, $k \in \mathbb{N}$ with $i + k\nu \geq m$. Then

$$x_i = x_{i+k\nu} = x_{i+k\nu+q} = x_{i+q} \quad \text{for all } i \geq \mu.$$

Therefore $q \geq \nu$ due to the minimality of ν, hence $q = \nu$ and $\nu \mid n = p\nu + \nu$. $\diamond$

Finite Sequences

Now let $x = (x_0, \ldots, x_{r-1}) \in M^r$ be a finite sequence of length r. It is called **periodic** if there is an index m and an integer $n \geq 1$, $m + n \leq r - 1$, such that

$$(2) \qquad x_{i+n} = x_i \quad \text{for all } i \text{ with } m \leq i \leq r - 1 - n.$$

If x is periodic, then the smallest index m for which there is an n such that (2) holds is called the **preperiod** of the sequence x. Denote it by μ. The smallest index n such that (2) holds with $m = \mu$ is called the **period** ν of the sequence x.

Bear in mind that Lemma 1 has no analogue for finite sequences. Of course, if (2) holds for a pair (m, n), then $m \geq \mu$ by definition of μ. However the divisibility of n by ν may break, as the following example shows:

Example

The sequence $x = (0, 1, 2, 1, 1) \in \mathbb{N}^5$ is periodic with $\mu = 1$ and $\nu = 3$. But (2) also holds for $m = 3$ and $n = 1$.

2 Recursive Sequences

A (finite or infinite sequence $x = (x_0, x_1, \ldots) \in M^r$ (with $r = 0$ or $r = \infty$[1]) is called **recursive** if there is a map $g\colon M \longrightarrow M$ such that

$$(3) \qquad x_i = g(x_{i-1}) \quad \text{for all } i = 1, 2, \ldots$$

An immediate consequence of this property is:

Lemma 2 *Let $x = (x_0, x_1, \ldots) \in M^r$ be a recursive sequence. Then:*

$$x_i = x_j \Longrightarrow x_{i+1} = x_{j+1} \quad \textit{for all index pairs } i, j \textit{ with } 0 \leq i < j < r - 1.$$

If there is a repetition $x_i = x_j$, then the sequence is periodic. In this case the preperiod μ is the smallest index such that the element x_μ reappears somewhere in the sequence, and $\mu + \nu$ is the index where the first repetition occurs.

In particular the values $x_0, \ldots, x_{\mu+\nu-1}$ are all distinct, and the values $x_0, \ldots, x_{\mu-1}$ never reappear in the sequence. See Figure 1.

[1] We use M^∞ as another notation for $M^{\mathbb{N}}$.

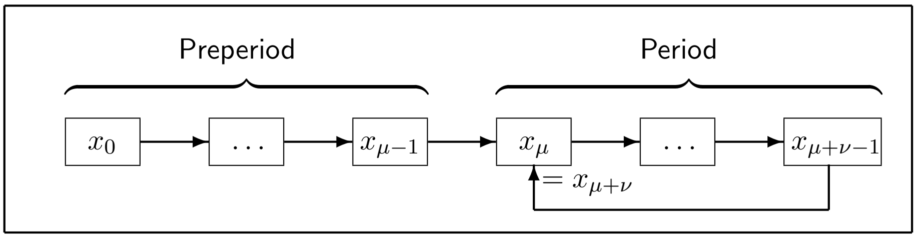

Figure 1: Period and preperiod

Proposition 1 *Let M be a set and $x = (x_i)_{i\in\mathbb{N}}$ be an infinite sequence in M. Then the following statements are equivalent:*

(i) *x is recursive.*

(ii) *x is periodic with preperiod μ and period ν, and the values $x_0, \ldots, x_{\mu+\nu-1}$ are all distinct, or all values x_i are distinct.*

(iii) *$x_i = x_j \Longrightarrow x_{i+1} = x_{j+1}$ for all index pairs i, j with $0 \le i < j$.*

Proof. For "(i) $\Longrightarrow$ (ii)" see the preceding paragraph.

"(ii) $\Longrightarrow$ (iii)" is trivial if all values are distinct. If there is a period see Figure 1.

For "(iii) $\Longrightarrow$ (i)" we may take the "state transition" map

$$g(x) = \begin{cases} x_i & \text{if there is an } i \text{ with } x = x_{i-1}, \\ \text{any value} & \text{otherwise.} \end{cases}$$

This map is well-defined by (iii). The remaining values don't matter (but make the solution ambiguous). $\diamond$

3 Sequences of Unknown Origin

Consider the situation where we are given a finite sequence $(x_0, \ldots, x_{r-1})$ of length r in a set M and know it is recursive, but the map g, that generates the sequence, is unknown. The first step is to find the preperiod and the period. The obvious algorithm proceeds by running through the given sequence, searching for a repetition. In addition one has to catch the case where the end of the given sequence is reached without detecting a repetition. Sage Example 1 implements this algorithm. Zero output values indicate that there is no repetition—a zero value in the second component of the return value suffices for this conclusion.

If the sequence is defined by a recursion formula, and is long enough, then the output values (if nonzero) are the length of the preperiod and the length of the period.

Sage Example 1 Searching for the first repetition in a sequence

```
def repetition(inlist):
  nn=len(inlist)
  for j in range(1,nn):
    for i in range(0,j):
      if (inlist[i] == inlist[j]):
        return [i,j-i]      # repetition detected
  return [0,0]              # no repetition detected
```

It is convenient to have an explicit notation for this situation: Given a finite sequence $x = (x_0, \ldots, x_{r-1}) \in M^r$ we consider the indices

- $\rho = \min\{j \mid 1 \leq j \leq r-1 \text{ and } x_i = x_j \text{ for some } i \leq j-1\}$,
- $\mu = \min\{i \mid 0 \leq i \leq \rho - 1 \text{ and } x_i = x_\rho\}$.

Then setting $\nu = \rho - \mu$ we call the pair (μ, ν) of integers **the first repetition** of x.

If x has a repetition, then $\rho = \mu + \nu \leq r - 1$. If x has no repetition we set $\rho = \mu = \nu = \infty$ (corresponding to the output value pair $(0, 0)$ in Sage Example 1).

Remark 1 By definition the elements $x_0, \ldots, x_{\mu+\nu-1}$ are pairwise different.

Remark 2 If the sequence x is periodic, then the first repetition (μ, ν) doesn't necessarily indicate the preperiod and period.

Example $x = (0, 1, 1, 0, 1, 0, 1)$ is periodic with preperiod 2 and period 2, but the first repetition is $\mu = 1$, $\nu = 1$, since $x_2 = x_1$. Or see the example in Section 1.

Now suppose that the given sequence $x = (x_0, \ldots, x_{r-1})$ is of unknown origin: We don't know how it is generated, and want to know whether it is recursive and, if yes, to learn as much on the generating ("state transition") map g as possible.

If there is no repetition in the given sequence, then the solutions are exactly the maps $g : M \longrightarrow M$ with

$$g(x) = \begin{cases} x_i & \text{if there is an } i \text{ with } 1 \leq i \leq r-1 \text{ and } x = x_{i-1}, \\ \text{any value} & \text{otherwise.} \end{cases}$$

In this case the sequence is recursive in a trivial way. This can occur only if $\#M \geq r$.

A similar trivial case is $\rho = \mu + \nu = r - 1$, the case where the last element x_{r-1} of the sequence is the only one that is a repetition. Then the sequence is also recursive, and the map g is given by the same formula. This can occur only if $\#M \geq r-1$.

If there is a repetition in the given sequence, then the algorithm above will find it, and provide two candidate numbers μ and ν for the preperiod and the period. Then we have to check the **consistency condition**

$$x_{i+\nu} = x_i \quad \text{for } \mu \leq i < r-\nu. \tag{4}$$

We call the sequence x **consistent** if its first repetition (μ, ν) satisfies the consistency condition (4).

Proposition 2 *Let M be a set and $x = (x_0, \ldots, x_{r-1}) \in M^r$ be a finite sequence containing a repetition. The following statements are equivalent:*

(i) *x is recursive.*

(ii) *x is consistent.*

(iii) *$x_i = x_j \Longrightarrow x_{i+1} = x_{j+1}$ for all $0 \leq i < j < r-1$.*

If these statements hold, then the generating map $g : M \longrightarrow M$ with $x_k = g(x_{k-1})$ for all $k = 1, \ldots, r-1$ has the values

$$g(x) = \begin{cases} x_k & \textit{if there is a } k \textit{ with } 1 \leq k \leq r-1 \\ & \quad \textit{and } x = x_{k-1}, \\ \text{any value} & \textit{otherwise,} \end{cases} \tag{5}$$

Proof. Since x contains a repetition we have $\mu, \nu < \infty$ for its first repetition (μ, ν), and the elements $x_0, \ldots, x_{\mu+\nu-1}$ are pairwise different.

"(iii) $\Longrightarrow$ (i)": By (iii) and (5) the map g is well-defined and yields a recursion formula for x.

"(i) $\Longrightarrow$ (ii)": Recursive implies periodic with preperiod μ and period ν. Thus (4) holds.

"(ii) $\Longrightarrow$ (iii)": For $0 \leq i < j < \mu+\nu$ always $x_i \neq x_j$, so there is nothing to prove.

Now assume $j \geq \mu + \nu$ and $x_i = x_j$. Let $j - \mu = q\nu + r$ be the integer division with remainder $0 \leq r < \nu$. Then $k := j - q\nu = \mu + r$ satisfies $\mu \leq k < \mu + \nu$ and, by (4), $x_k = x_j$.

If $i < \mu$, this yields $x_i = x_k$, contradicting the minimality of $\mu+\nu$. Hence we may assume $i \geq \mu$. Again let $i - \mu = p\nu + s$ be the integer division with remainder $0 \leq s < \nu$. Then $t := i - p\nu = \mu + s$ satisfies $\mu \leq t < \mu + \nu$ and

$$x_k = x_j = x_i = x_t.$$

The minimality of $\mu+\nu$ forces $k = t$. Since $k+1 = t+1 \geq \mu$ the consistency condition (4) implies

$$x_{k+1} = x_{k+1+q\nu} = x_{j+1} \quad \text{and} \quad x_{t+1} = x_{t+1+p\nu} = x_{i+1},$$

hence $x_{i+1} = x_{j+1}$. $\diamond$

Consistency is checked by Sage Example 2. The return value `crash` is the index where the first inconsistency is found, and 0 in case of success. If the consistency condition is violated, then the sequence is definitely not recursive (or part of recursive sequence).

Sage Example 2 Consistency check whether a conjectured period q with preperiod p persists for the remainder of a sequence.

```
def checkPeriod(inlist,p,q):
  crash = 0
  nn = len(inlist)
  for i in range(p,nn-q):
    if (inlist[i+q] != inlist[i]):
      crash = i
      break
  return crash
```

4 Multistep Recursion

Now assume Σ is a set and $f\colon \Sigma^l \longrightarrow \Sigma$ is a function. Each l-tuple of initial values $(u_0, \ldots, u_{l-1})$ gives rise to an infinite sequence in Σ by the recursion formula of depth l:

$$(6) \qquad u_i = f(u_{i-1}, \ldots, u_{i-l}) \quad \text{for } i = l,\, l+1, \ldots$$

Note that this scenario describes a feedback shift register (FSR) of length l over Σ, see Figure 2. Therefore we call f the **feedback function** of the sequence $u = (u_i)_{i \in \mathbb{N}}$.

We call a finite sequence $u = (u_0, u_1, \ldots, u_{r-1}) \in \Sigma^r$ **recursive of depth** l if $l < r$ and u satisfies a recursion formula of type (6) for $i = l, \ldots, r-1$. The minimal integer l for which u is recursive of depth l is called **the recursion depth**[2] of u, denoted by $\Lambda(u)$. If $\Lambda(u) = 1$, then u is recursive in the sense of Section 2.

Remark 1 If the sequence has no repetition, or if its last element u_{r-1} is the only one that is a repetition, then $\Lambda(u) = 1$ by the remarks in Section 3. Also $\Lambda(u) = 1$ if u is constant, in particular if $r = 1$.

Remark 2 In any case $\Lambda(u) \leq r - 1$ with a feedback function f that is completely arbitrary except for the value $f(u_{r-2}, \ldots, u_0) = u_{r-1}$.

[2]in [3] called maximum order complexity. This notation is adequate in the case where Σ is a finite field for then all functions are polynomials. In [1] it is called span.

Remark 3 Let $\Phi\colon \Sigma \longrightarrow \Sigma'$ be an injective map. Then $\Lambda(\Phi u) = \Lambda(u)$. In particular in the case $\Sigma = \mathbb{F}_2$, a two-element set, the recursion depth is unchanged if all sequence elements are flipped, that is $\Phi : 0 \leftrightarrow 1$.

The task is: For a given finite sequence $u = (u_0, u_1, \ldots, u_{r-1})$ of length r find its recursion depth $\Lambda(u)$, and reconstruct the corresponding feedback function f from u as far as possible.

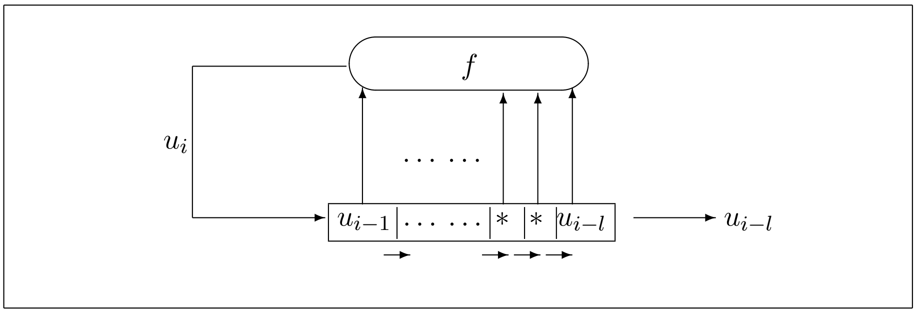

Figure 2: A feedback shift register (FSR)

To apply the previous results on recursions of depth 1 we consider the **state vectors** of dimension l,

$$u_{(i)} = \begin{pmatrix} u_i \\ \vdots \\ u_{i+l-1} \end{pmatrix} \in M := \Sigma^l \tag{7}$$

(for any $l \geq 1$ and, if r is finite, $l \leq r$), and, if u is recursive of depth l, the state transition map

$$g\colon M \longrightarrow M, \qquad \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_l \end{pmatrix} \mapsto \begin{pmatrix} x_2 \\ \vdots \\ x_l \\ f(x_l, \ldots, x_1) \end{pmatrix}.$$

The state vectors describe the content of the FSR for the steps $i = 0, 1, 2, \ldots$ and follow the recursion formula

$$u_{(i)} = \begin{pmatrix} u_i \\ \vdots \\ u_{i+l-2} \\ u_{i+l-1} \end{pmatrix} = \begin{pmatrix} u_i \\ \vdots \\ u_{i+l-2} \\ f(u_{i+l-2}, \ldots, u_{i-1}) \end{pmatrix} = g\begin{pmatrix} u_{i-1} \\ u_i \\ \vdots \\ u_{i+l-2} \end{pmatrix} = g(u_{(i-1)})$$

for $i = 1, \ldots, r-l$ (or for all i if $r = \infty$)—the sequence of state vectors is recursive in the sense of Section 2.

5 Periods and State Vectors

A sequence doesn't need to be recursive for the concept of state vectors to make sense. Given any (finite or infinite) sequence $u \in \Sigma^r$ and any integer $l \geq 1$ (and $l \leq r$ if r is finite) formula (7) allows the construction of the sequence of state vectors of dimension l. The code in Sage Example 3 implements this procedure for finite sequences.

Sage Example 3 Constructing the sequence of state vectors for depth l

```
def stateVectors(inlist,l):
  nn = len(inlist)
  outlist = []
  for i in range(0,nn-l+1):
    t = inlist[i:(i+l)]
    outlist.append(t)
  return outlist
```

First assume $r = \infty$ and assume the infinite sequence $u = (u_0, u_1, \ldots)$ has the period ν after a preperiod μ. Then

$$u_{(i+\nu)} = \begin{pmatrix} u_{i+\nu} \\ \vdots \\ u_{i+\nu+l-1} \end{pmatrix} = \begin{pmatrix} u_i \\ \vdots \\ u_{i+l-1} \end{pmatrix} = u_{(i)} \quad \text{for all } i \geq \mu. \tag{8}$$

Thus also the sequence of state vectors of dimension l is periodic with preperiod μ and period ν. Conversely equation (8) implies the periodicity of u:

Lemma 3 *Let $u \in \Sigma^\infty$ be an infinite sequence in the set Σ, let $l \geq 1$ be an integer, and let $(u_{(i)})_{i\in\mathbb{N}}$ be the sequence of state vectors of dimension l. The following statements are equivalent:*

(i) *u is periodic with preperiod μ und period ν.*

(ii) *$(u_{(i)})$ is periodic with preperiod μ und period ν.*

For a finite sequence $u = (u_0, \ldots, u_{r-1}) \in \Sigma^r$ and $1 \leq l \leq r$ the state vectors are $u_{(0)}, \ldots, u_{(r-l)} \in \Sigma^l$. If u is periodic with preperiod μ and period ν, then $u_{i+\nu} = u_i$ for all i with $\mu \leq i \leq r-1-\nu$. With this restriction of the indices equation (8) shows that $u_{(i+\nu)} = u_{(i)}$, where the largest occuring index, $i + \nu + l - 1$, is bounded by $r - 1$, hence $\mu \leq i \leq r - l - \nu$. Therefore this statement makes sense only if $\mu + \nu \leq r - l$.

Conversely assume that $\mu + \nu \leq r - l$ and $(u_{(i)})$ is periodic with preperiod μ and period ν, in particular $u_{(i+\nu)} = u_{(i)}$ for all indices i with $\mu \leq i \leq r - l - \nu$. Then obviously $u_{i+\nu} = u_i$ for all i with $\mu \leq i \leq r - 1 - \nu$. We have shown:

Proposition 3 *Let $u \in \Sigma^r$ be a finite sequence in the set Σ, let $l \geq 1$, $l < r$, be an integer, and let $(u_{(i)})_{0 \leq i \leq r-l}$ be the sequence of state vectors of dimension l. Then the following statements are equivalent:*

(i) *u is periodic with preperiod μ and period ν, and $\mu + \nu \leq r - l$.*

(ii) *$(u_{(i)})$ is periodic with preperiod μ und period ν.*

Now if, while considering the sequence $u \in \Sigma^r$, we detect a first repetition (μ, ν) in dimension l, hence $u_{(\mu+\nu)} = u_{(\mu)}$, then applying Proposition 2 and the consistency condition (4) we can decide whether the sequence of state vectors is generated by a recursion $u_{(i)} = g(u_{(i-1)})$ and hence the sequence u itself is generated by a recursion formula of depth l. If so we can reconstruct the generating map $g : \Sigma^l \longrightarrow \Sigma^l$ for the state vectors by the (trivially modified) Formula (5):

$$g(x) = \begin{cases} u_{(k)} & \text{if there is a } k \text{ with } 1 \leq k \leq \mu + \nu - 1 \\ & \quad \text{and } x = u_{(k-1)}, \\ \text{any value} & \text{otherwise,} \end{cases}$$

as well as the generating function $f \colon \Sigma^l \longrightarrow \Sigma$ by the formula

$$f(x) = \begin{cases} u_{k+l-1} & \text{if there is a } k \text{ with } 1 \leq k \leq \mu + \nu - 1 \\ & \quad \text{and } (x_1, \ldots, x_l) = (u_{k+l-2}, \ldots, u_{k-1}), \\ \text{any value} & \text{otherwise,} \end{cases}$$

since u_{k+l-1} is the l-th coordinate of $u_{(k)}$.

This procedure presupposes a known recursion depth l.

6 Unknown Recursion Depth

Now assume we are given a sequence $u = (u_0, u_1, \ldots, u_{r-1}) \in \Sigma^r$. We want to

- find the recursion depth $\Lambda(u)$
- and construct the corresponding feedback function. (In other words: Construct a minimal FSR over Σ that generates u.)

The previous considerations show that, after having found the recursion depth $l = \Lambda(u)$ and the corresponding period for the l-dimensional state vectors, the construction of the feedback function f is trivial. We can't expect a unique solution for f, but that doesn't matter.

Since we have no a priori clue about the recursion depth l we try $l = 1, 2, \ldots, r-1$ in order until we find a consistent first repetition in the state vectors of dimension l. In the worst case the algorithm terminates with the trivial (hardly useful) solution $l = r - 1$, and g and f completely arbitrary except at $(u_0, \ldots, u_{r-2})$.

For each dimension l we build the sequence of state vectors $u_{(0)}, \ldots, u_{(r-l)}$. If we find a repetition in dimension $l < r$, then obviously there also was a repetition in each dimension k with $1 \leq k < l$:

$$u_{(j)} = \begin{pmatrix} u_j \\ \vdots \\ u_{j+l-1} \end{pmatrix} = \begin{pmatrix} u_i \\ \vdots \\ u_{i+l-1} \end{pmatrix} = u_{(i)} \Longrightarrow \begin{pmatrix} u_j \\ \vdots \\ u_{j+k-1} \end{pmatrix} = \begin{pmatrix} u_i \\ \vdots \\ u_{i+k-1} \end{pmatrix}.$$

An immediate conclusion is:

Lemma 4 *Let $u = (u_0, u_1, \ldots, u_{r-1}) \in \Sigma^r$, and consider the corresponding sequence of state vectors of dimension l. Assume this sequence has no repetition. Then $\Lambda(u) \leq l$, and also the sequences of state vectors of larger dimensions have no repetition.*

In particular if the sequence u itself has no repeated elements we may skip the search for any depth $l \geq 2$, and output the solution $\Lambda(u) = 1$.

For each l and the corresponding sequence $(u_{(i)})_{0 \leq i \leq r-l}$ of state vectors we define:

- (μ_l, ν_l), the first repetition (maybe $\mu_l = \nu_l = \infty$), and $\rho_l = \mu_l + \nu_l$.
- $\chi_l = \min\{c \geq \mu_l + 1 \mid u_{(c+\nu_l)} \neq u_{(c)}\}$ (the first inconsistency). If $\rho_l = \infty$ (there is no repetition) or the consistency condition (4) holds for $\mu_l \leq i \leq r - l - \nu_l - 1$ (ν_l is a period for the remainder of the sequence), then $\chi_l = \infty$.

With these notations the previous considerations yield:

Lemma 5 (i) *$\rho_l = \infty \Longrightarrow \rho_k = \infty$ for all $k \geq l$, $k \leq r$.*

(ii) *For $1 \leq l \leq k$ we have $\rho_l \leq \rho_k$.*

(iii) *The status vectors of dimension l have a repetition if and only if $\rho_l \leq r - l$. If this is the case, then $0 \leq \mu_l < r-l$ and $1 \leq \nu_l \leq r-l-\mu_l$.*

(iv) *If the status vectors of dimension l have no repetitions (in other words $\rho_l = \infty$), then $\Lambda(u) \leq l$.*

(v) *If the status vectors of dimension l have repetitions, and the first one is consistent (in other words $\chi_l = \infty$), then $\Lambda(u) \leq l$. This applies in particular if the last vector is the first one that is a repetition (in other words $\rho_l = r - l$).*

(vi) *If the status vectors of dimension l have any inconsistent repetition, then $\Lambda(u) > l$.*

(vii) *If the status vectors of dimension l have an inconsistent first repetition (in other words $\chi_l < \infty$), then $\mu_l + 1 \leq \chi_l \leq r - l - \nu_l$, $\Lambda(u) > l$, and $\chi_1, \ldots, \chi_{l-1} < \infty$.*

(viii) *If $\chi_{l-1} < \infty$ and $\chi_l = \infty$, then $\Lambda(u) = l$.*

(ix) *If $\chi_l < \infty$, then $\Lambda(u) \geq l + \chi_l - \mu_l \geq l + 1$.*

(x) *If $\chi_l < \infty$ and $l + \chi_l - \mu_l \geq r - 1$, then $\Lambda(u) = r - 1$.*

Proof. (i)–(v) and (vii)–(viii) follow immediately from the definitions.

(vi) The status vectors cannot satisfy a recursion formula.

(ix) is a consequence of the following Lemma 7 (ii).

(x) follows immediately from (ix). ◇

Lemma 6 *Let $u = (u_0, u_1, \ldots, u_{r-1}) \in \Sigma^r$, and let $(u_{(i)})_{0 \leq i \leq r-l}$ be the corresponding sequence of state vectors of dimension l where $1 \leq l \leq r - 1$. Let*

$$M_l := \{(i,j) \mid 0 \leq i < j \leq r - l,\ u_{(i)} = u_{(j)},\ u_{i+l} \neq u_{j+l}\}.$$

Then $\Lambda(u) = \max\{l \mid M_l \neq \emptyset\}$.

Proof. If $M_l \neq \emptyset$, then the status vectors of dimension l have an inconsistent repetition, hence $\Lambda(u) > l$ by Lemma 5 (vi).

If $M_l = \emptyset$, then the sequence of status vectors satisfies (iii) of Proposition 1, hence is recursive. Thus $\Lambda(u) \leq l$. ◇

Lemma 7 *Let $u = (u_0, u_1, \ldots, u_{r-1}) \in \Sigma^r$, and let $(u_{(i)})_{0 \leq i \leq r-l}$ be the corresponding sequence of state vectors of dimension l where $1 \leq l \leq r - 1$. Let $(\bar{u}_{(i)})_{0 \leq i \leq r-k}$ be the sequence of state vectors of dimension $k = l + 1$.*

(i) *Assume there is a repetition $u_{(m+n)} = u_{(m)}$ that violates the consistency condition (4) at an index $s > m$, thus $u_{(s+n)} \neq u_{(s)}$. Then*

- *either $\bar{u}_{(m+n)} \neq \bar{u}_{(m)}$. This occurs if $s = m + 1$.*
- *or $\bar{u}_{(m+n)} = \bar{u}_{(m)}$, and this repetition violates the consistency condition at index $s - 1$. This occurs if $s \geq m + 2$.*

(ii) *If $\mu_l + 2 \leq \chi_l < \infty$, then $\mu_k = \mu_l$, $\nu_k = \nu_l$, and $\chi_k = \chi_l - 1$.*

Proof. First note that

$$\bar{u}_{(i)} = \begin{pmatrix} u_{(i)} \\ u_{i+l} \end{pmatrix} = \begin{pmatrix} u_i \\ u_{(i+1)} \end{pmatrix} \quad \text{for } i = 0, \ldots, r-k.$$

Hence $\bar{u}_{(i)} = \bar{u}_{(j)}$ if and only if $u_{(i)} = u_{(j)}$ and $u_{(i+1)} = u_{(j+1)}$ for $0 \le i \le j \le r-k$.

(i) If $s = m+1$ we have $u_{(m+n+1)} \neq u_{(m+1)}$, hence $\bar{u}_{(m+n)} \neq \bar{u}_{(m)}$. If $s \ge m+2$ we have

- $u_{(m+1)} = u_{(m+1+n)}$, thus $\bar{u}_{(m)} = \bar{u}_{(m+n)}$.
- $u_{(s-1)} = u_{(s-1+n)}$, but $u_{(s)} \neq u_{(s+n)}$. Hence $\bar{u}_{(s-1)} \neq \bar{u}_{(s-1+n)}$.

(ii) $\bar{u}_{(0)}, \ldots, \bar{u}_{(\mu_l+\nu_l-1)}$ are pairwise different. Since $u_{(\mu_l)} = u_{(\mu_l+\nu_l)}$ and $u_{(\mu_l+1)} = u_{(\mu_l+\nu_l+1)}$ also $\bar{u}_{(\mu_l)} = \bar{u}_{(\mu_l+\nu_l)}$. Thus $\mu_k = \mu_l$ and $\nu_k = \nu_l$. By (i) the first inconsistency occurs at $\chi_l - 1$. ◇

7 Finding the Recursion Depth

Example

Before stating the general algorithm let's go through a simple example. Let Σ be a two-element set, say $\Sigma = \mathbb{F}_2$. For $r = 10$ consider the sequence $u = (1,0,1,0,1,1,0,1,0,1)$.

Trying to find a recursion of depth $l = 1$ we find:

- $u_0 = 1 = u_2$, hence $\rho_1 = 2$, $\mu_1 = 0$, $\nu_1 = 2$.
- $u_1 = 0 = u_3$ and $u_2 = 1 = u_4$ (consistent).
- $u_3 = 0 \neq u_5$, hence an inconsistency at $\chi_1 = 3$.

Assuming $l = 2$ we likewise find

$$u_{(0)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = u_{(2)}, \quad u_{(1)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = u_{(3)}, \quad u_{(2)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \neq \begin{pmatrix} 1 \\ 1 \end{pmatrix} = u_{(4)}.$$

Hence $\rho_2 = 2$, $\mu_2 = 0$, $\nu_2 = 2$, and an inconsistency at $\chi_2 = 2$, as predicted by Lemma 7 (ii).

Trying $l = 3$ we find:

$$u_{(0)} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = u_{(2)}, \quad u_{(1)} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \neq \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = u_{(3)},$$

resulting in $\rho_3 = 2$, $\mu_3 = 0$, $\nu_3 = 2$, $\chi_3 = 1 = m_3 + 1$. Thus we encounter the first alternative of Lemma 7 (i) and expect a changing situation for $l = 4$. And indeed:

$$u_{(0)} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = u_{(5)}, \quad u_{(1)} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = u_{(6)},$$

the consistency condition is satisfied, and we get a recursion of depth $l = 4$ with a preperiod $\mu = \mu_4 = 0$ and a period $\nu = \nu_4 = 5$.

Algorithm

Lemmas 4–7 translate to the algorithm:

1. Start with $l = 1$.
2. For the current value of l build the sequence of state vectors $u_{(0)}, \ldots, u_{(n-l)}$ of dimension l and search for its first repetition (μ_l, ν_l).
   (a) If there is no repetition STOP and output $(l, 0, 0)$.
   (b) If a repetition is found check the first one, (μ_l, ν_l), for consistency.
      - If it is consistent STOP and output the current values for $\Lambda(u) = l$, preperiod $\mu = \mu_l$, and period $\nu = \nu_l$.
      - Otherwise there is no recursion formula of depth l, the first inconsistency occurs at $\chi_l \leq r - 1 - \nu_l$. Increment l by $\chi_l - \mu_l$. [*Comment:* For $l \leq s < l + \chi_l$ we have, by Lemma 7, $\mu_s = \mu_l$ with an inconsistency at $\chi_s = \chi_l - (s - l)$.]
3. If $l \geq r - 1$ STOP and output $(r - 1, 0, 0)$.

Sage Example 4 contains an implementation of this algorithm.

For the reconstruction of the feedback function f use the formula

$$f(x) = \begin{cases} u_i & \text{if there is an } i \text{ with } l \leq i \leq \mu + \nu - 1 \\ & \quad \text{and } x = u_{(i-l)}, \\ \text{any value} & \text{otherwise.} \end{cases} \tag{9}$$

Sage Example 4 Find the shortest FSR that generates a given sequence.

```
def getFSR(seq):
  nn=len(seq)
  l = 1
  while l < nn-1:
    vList=stateVectors(seq,l)
    pp=repetition(vList)
    mu=pp[0]          # supposed preperiod
    nu=pp[1]          # supposed period
    if nu == 0:       # no repetition found
      return [l,0,0]
    else:
      cr=checkPeriod(vList,mu,nu) # consistent with remainder
                                  # of sequence?
      if cr == 0:             # consistent ==> recursion found
        result = [l,mu,nu]
        return result
      else:                   # cr > mu
        l += cr - mu          # inconsistent ==> increment l
  return [nn-1,0,0]
```

The loop for the current value l of the dimension (or tentative recursion depth) has three possible outcomes:

1. There is no repetition in the state vectors. Then the algorithm terminates with output $(l, 0, 0)$.
2. The first repetition (μ_l, ν_l) in the sequence of state vectors is consistent. Then the algorithm terminates with output (l, μ_l, ν_l).
3. The first repetition is inconsistent. Then the algorithm continues with an incremented value of l.

Proposition 4 *Let* $u = (u_0, u_1, \ldots, u_{r-1}) \in \Sigma^r$ *be a sequence of length* $r \geq 3$*. Then* $\Lambda(u) = r - 1$ *if and only if* $u_0 = \ldots = u_{r-2} \neq u_{r-1}$*.*

Proof. If $u_0 = \ldots = u_{r-2} \neq u_{r-1}$, then $\rho_1 = 1$, $\mu_1 = 0$, $\nu_1 = 1$, $\chi_1 = r - 2$, and $\chi_1 - \mu_1 = r - 2$. Hence $\Lambda(u) \geq 1 + r - 2 = r - 1$, hence $\Lambda(u) = r - 1$.

Conversely assume that $\Lambda(u) = r - 1$. Then for each $l < r - 1$ the algorithm has outcome number 3: The state vectors of dimension l have an inconsistent first repetition. In particular for $l = r - 2$ the $r - l = 3$ state vectors are:

$$u_{(0)} = \begin{pmatrix} u_0 \\ \vdots \\ u_{r-3} \end{pmatrix}, \quad u_{(1)} = \begin{pmatrix} u_1 \\ \vdots \\ u_{r-2} \end{pmatrix}, \quad u_{(2)} = \begin{pmatrix} u_2 \\ \vdots \\ u_{r-1} \end{pmatrix}.$$

The only possibility for an inconsistent repetition is $u_{(0)} = u_{(1)} \neq u_{(2)}$. Hence $u_0 = \ldots = u_{r-2} \neq u_{r-1}$. $\diamond$

For $r = 2$ we get $\Lambda(u) = 1$ no matter which $u = (u_0, u_1) \in \Sigma^2$ we take. (And for $r = 1$ the algorithm outputs $(0, 0, 0)$, answering a question that nobody would ask.)

Remark

For an application in cryptanalysis one would like to continue the given sequence $u \in \Sigma^r$ beyond its end, or in other words to predict further elements of the sequence. If the algorithm outputs (l, μ, ν) with nonzero ν, then it suggests a continuation based upon the detected periodicity. This may or may not lead to a success, but at least it provides a clue.

The situation is by far worse if the algorithm outputs the trivial result $(l, 0, 0)$. Then the shortest FSR that generates the given input sequence has a feedback function f that is completely arbitrary beyond the end of u, see Formula (9) with $\mu + \nu$ replaced by r. In particular the algorithm provides absolutely no clue for predicting further elements of the sequence.

8 Examples

1. $r = 4$, $u = 0110$ (where the string represents a sequence)

   - For $l = 1$ we have $\rho_1 = 2$, $\mu_1 = 1$, $\nu_1 = 1$, $u_1 = u_2 \neq u_3$, thus $\chi_1 = 2$. Hence we increment l by $\chi_1 - \mu_1 = 1$.

   - For $l = 2$ the state vectors

   $$\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

     have no repetition, hence the algorithm stops with output $(2, 0, 0)$.

2. $r = 6$, $u = 001010$

   - For $l = 1$ we have $\rho_1 = 1$, $\mu_1 = 0$, $\nu_1 = 1$, $u_0 = u_1 \neq u_2$, $\chi_1 = 1$. Hence we increment l by $\chi_1 - \mu_1 = 1$.

- For $l = 2$ the state vectors

$$\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}$$

  have $\rho_2 = 3$, $\mu_2 = 1$, $\nu_2 = 2$. Since $u_{(2)} = u_{(4)}$ this repetition is consistent, and the algorithm stops with output $(2, 1, 2)$.

3. $r = 7$, $u = 0110110$

- For $l = 1$ we have $\rho_1 = 2$, $\mu_1 = 0$, $\nu_1 = 1$, $u_1 = u_2 \neq u_3$, $\chi_1 = 2$. Hence we increment l by $\chi_1 - \mu_1 = 1$.

- For $l = 2$ the state vectors

$$\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix} \begin{pmatrix}1\\0\end{pmatrix}$$

  have $\rho_2 = 3$, $\mu_2 = 0$, $\nu_2 = 3$. Since $u_{(1)} = u_{(4)}$ and $u_{(2)} = u_{(5)}$ this repetition is consistent, and the algorithm stops with output $(2, 0, 3)$.

9 The distribution of the recursion depth

In this section we focus on binary sequences, that is we consider the two-element alphabet $\Sigma = \{0, 1\}$, abbreviated by $\mathbb{F}_2$, the field of two elements—however this algebraic structure is not relevant for the moment.

Sage Example 5 generates the list of the recursion depths of all binary sequences of a given length (together with the corresponding period data). It uses the conversion of an integer to the list of bits of its binary representation, see Sage Example 6 (that is taken from the module `Bitblock.sage` of the cryptology script). Sage Example 7 shows an exemplary application for length 6. The result:

```
0 |  [0, 0, 0, 0, 0, 0] |  [1, 0, 1]
1 |  [0, 0, 0, 0, 0, 1] |  [5, 0, 0]
2 |  [0, 0, 0, 0, 1, 0] |  [4, 0, 0]
3 |  [0, 0, 0, 0, 1, 1] |  [4, 0, 0]
4 |  [0, 0, 0, 1, 0, 0] |  [3, 0, 0]
5 |  [0, 0, 0, 1, 0, 1] |  [3, 0, 0]
6 |  [0, 0, 0, 1, 1, 0] |  [3, 0, 0]
7 |  [0, 0, 0, 1, 1, 1] |  [3, 0, 0]
8 |  [0, 0, 1, 0, 0, 0] |  [3, 0, 0]
9 |  [0, 0, 1, 0, 0, 1] |  [2, 0, 3]
10 |  [0, 0, 1, 0, 1, 0] |  [2, 1, 2]
11 |  [0, 0, 1, 0, 1, 1] |  [3, 0, 0]
12 |  [0, 0, 1, 1, 0, 0] |  [2, 0, 4]
```

```
13 |  [0, 0, 1, 1, 0, 1] |  [2, 1, 3]
14 |  [0, 0, 1, 1, 1, 0] |  [3, 0, 0]
15 |  [0, 0, 1, 1, 1, 1] |  [2, 2, 1]
16 |  [0, 1, 0, 0, 0, 0] |  [2, 2, 1]
17 |  [0, 1, 0, 0, 0, 1] |  [3, 0, 0]
18 |  [0, 1, 0, 0, 1, 0] |  [2, 0, 3]
19 |  [0, 1, 0, 0, 1, 1] |  [3, 0, 0]
20 |  [0, 1, 0, 1, 0, 0] |  [4, 0, 0]
21 |  [0, 1, 0, 1, 0, 1] |  [1, 0, 2]
22 |  [0, 1, 0, 1, 1, 0] |  [3, 0, 0]
23 |  [0, 1, 0, 1, 1, 1] |  [3, 0, 0]
24 |  [0, 1, 1, 0, 0, 0] |  [2, 3, 1]
25 |  [0, 1, 1, 0, 0, 1] |  [2, 0, 4]
26 |  [0, 1, 1, 0, 1, 0] |  [3, 0, 0]
27 |  [0, 1, 1, 0, 1, 1] |  [2, 0, 3]
28 |  [0, 1, 1, 1, 0, 0] |  [3, 0, 0]
29 |  [0, 1, 1, 1, 0, 1] |  [3, 0, 0]
30 |  [0, 1, 1, 1, 1, 0] |  [4, 0, 0]
31 |  [0, 1, 1, 1, 1, 1] |  [1, 1, 1]
32 |  [1, 0, 0, 0, 0, 0] |  [1, 1, 1]
33 |  [1, 0, 0, 0, 0, 1] |  [4, 0, 0]
34 |  [1, 0, 0, 0, 1, 0] |  [3, 0, 0]
35 |  [1, 0, 0, 0, 1, 1] |  [3, 0, 0]
36 |  [1, 0, 0, 1, 0, 0] |  [2, 0, 3]
37 |  [1, 0, 0, 1, 0, 1] |  [3, 0, 0]
38 |  [1, 0, 0, 1, 1, 0] |  [2, 0, 4]
39 |  [1, 0, 0, 1, 1, 1] |  [2, 3, 1]
40 |  [1, 0, 1, 0, 0, 0] |  [3, 0, 0]
41 |  [1, 0, 1, 0, 0, 1] |  [3, 0, 0]
42 |  [1, 0, 1, 0, 1, 0] |  [1, 0, 2]
43 |  [1, 0, 1, 0, 1, 1] |  [4, 0, 0]
44 |  [1, 0, 1, 1, 0, 0] |  [3, 0, 0]
45 |  [1, 0, 1, 1, 0, 1] |  [2, 0, 3]
46 |  [1, 0, 1, 1, 1, 0] |  [3, 0, 0]
47 |  [1, 0, 1, 1, 1, 1] |  [2, 2, 1]
48 |  [1, 1, 0, 0, 0, 0] |  [2, 2, 1]
49 |  [1, 1, 0, 0, 0, 1] |  [3, 0, 0]
50 |  [1, 1, 0, 0, 1, 0] |  [2, 1, 3]
51 |  [1, 1, 0, 0, 1, 1] |  [2, 0, 4]
52 |  [1, 1, 0, 1, 0, 0] |  [3, 0, 0]
53 |  [1, 1, 0, 1, 0, 1] |  [2, 1, 2]
54 |  [1, 1, 0, 1, 1, 0] |  [2, 0, 3]
55 |  [1, 1, 0, 1, 1, 1] |  [3, 0, 0]
56 |  [1, 1, 1, 0, 0, 0] |  [3, 0, 0]
```

```
57 |  [1, 1, 1, 0, 0, 1] |  [3, 0, 0]
58 |  [1, 1, 1, 0, 1, 0] |  [3, 0, 0]
59 |  [1, 1, 1, 0, 1, 1] |  [3, 0, 0]
60 |  [1, 1, 1, 1, 0, 0] |  [4, 0, 0]
61 |  [1, 1, 1, 1, 0, 1] |  [4, 0, 0]
62 |  [1, 1, 1, 1, 1, 0] |  [5, 0, 0]
63 |  [1, 1, 1, 1, 1, 1] |  [1, 0, 1]
```

Sage Example 5 List the recursion depths of all binary sequences of a given length.

```
def NLdistr(rr):
  Llist = []
  for t in range(2^rr):
    bb = int2bbl(t,rr)
    ll = getFSR(bb)
    Llist.append(ll)
  return Llist
```

Sage Example 6 Convert an integer to a bitblock of length dim via its base-2 representation.

```
def int2bbl(number,dim):
  n = number                          # catch input
  b = []                              # initialize output
  for i in range(0,dim):
    bit = n % 2                       # next base-2 bit
    b = [bit] + b                     # prepend
    n = (n - bit)//2
  return b
```

Sage Example 7 List the recursion depths of all binary sequences of length 6.

```
sage: r = 6
sage: results = NLdistr(r)
sage: for i in range(2^r):
    print(i, "| ", int2bbl(i,r), "| ", results[i])
```

To get an overview over the distribution of recursion depths we use Sage Example 8. The sage command `sage: distr = NLctr(r); distr` (still for

$r = 6$) yields the list [0, 6, 20, 28, 8, 2]. For a somewhat more illustrative example we take $r = 20$ and show the result as a histogram, see Figure 3.

Sage Example 8 Distribution of the recursion depths of all binary sequences of a given length

```
def NLctr(rr):
  ctrlist = [0]*rr
  for t in range(2^rr):
    bb = int2bbl(t,rr)
    ll = getFSR(bb)[0]
    ctrlist[ll] += 1
  return ctrlist
```

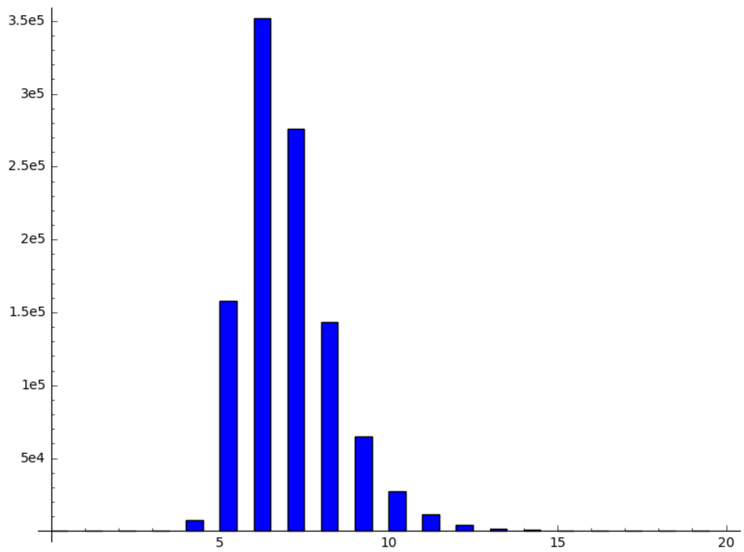

Figure 3: Distribution of recursion depths of binary sequences of length 20

10 To Do

Several questions seem worth of further investigation:

1. Explore the impact of the size $\#\Sigma$ of the “alphabet” Σ on the distribution of the recursion depth.
    - For example a sequence $u \in \Sigma^r$ must contain a repetition if $r < \#\Sigma$.

2. Table 1 contains the results of `NLctr` for binary sequences of lengths r with $3 \leq r \leq 20$. Explore the regularities of this scheme.

   - Denote the number of sequences of length r with recursion depth l by $A(r,l)$. Then $A(16,8) = A(17,9) = A(18,10) = \ldots = 4562$. Why?

3. The paper [3] indicates a proof that the mean value of $\Lambda(u)$ over $u \in \Sigma^r$ is $2 \cdot \log_s(r)$ where $s = \#\Sigma$. Flesh out this proof.

4. For $u \in \Sigma^r$ let the "recursive profile" be the sequence $(\Lambda(u^{(t)}))_{1 \leq t \leq r}$ where $u^{(t)}$ is the partial sequence $(u_0, \ldots, u_{t-1})$. Explore the recursive profile.

   - In particular study the transition $t \to t+1$. Compare also the considerations in [3] and the Berlekamp-Massey algorithm for the linearity profile.

5. For binary sequences compare the recursion depth with the linear complexity[3], and the recursive profile with the linearity profile.

6. Study the complexity of the algorithm in Section 7. Compare it with the algorithm given in [3].

7. For a binary sequence $u \in \mathbb{F}_2^r$ of recursion depth l the feedback function f is uniquely determined on the subset $\{u_{(0)}, \ldots, u_{(\mu_l+\nu_l-1)}\} \subseteq \mathbb{F}_2^l$, and completely arbitrary on the complementary subset of $2^l - \mu_l - \nu_l$ elements of $\mathbb{F}_2^r$. Which options are left for optimizing f with respect to diverse quantities such as the algebraic degree, or other nonlinearity measures?

8. Compare the recursion depth as a measure of the complexity of a sequence with the linear complexity and the Turing complexity.

[3] the analogous notion using *linear* feedback shift registers only, see the cryptology lecture notes

Table 1: Distribution of the recursion depth for lengths up to 20

```
[0,6, 2]
[0,6, 8,  2]
[0,6,16,  8,   2]
[0,6,20, 28,   8,     2]
[0,6,20, 64,  28,     8,     2]
[0,6,20,106,  86,    28,     8,     2]
[0,6,20,154, 208,    86,    28,     8,     2]
[0,6,20,194, 430,   250,    86,    28,     8,    2]
[0,6,20,210, 808,   630,   250,    86,    28,    8,    2]
[0,6,20,210,1366,  1440,   680,   250,    86,   28,    8,    2]
[0,6,20,210,2084,  3114,  1704,   680,   250,   86,   28,    8,   2]
[0,6,20,210,2938,  6344,  4020,  1792,   680,  250,   86,   28,   8,   2]
[0,6,20,210,3858, 12206,  9162,  4460,  1792,  680,  250,   86,  28,   8,  2]
[0,6,20,210,4814, 22152, 20242, 10684,  4562, 1792,  680,  250,  86,  28,  8,  2]
[0,6,20,210,5774, 38298, 43262, 24890, 11204, 4562, 1792,  680, 250,  86, 28,  8, 2]
[0,6,20,210,6686, 63524, 89570, 56660, 26716,11344, 4562, 1792, 680, 250, 86, 28, 8, 2]
[0,6,20,210,7390,101714,179978,126316, 62438,27464,11344, 4562,1792, 680,250, 86,28, 8,2]
[0,6,20,210,7646,157816,352060,275938,143442,65072,27614,11344,4562,1792,680,250,86,28,8,2]
```

Bibliography

[1] Agnes Hui Chan, Richard A. Games. On the quadratic spans of periodic sequences. CRYPTO '89, 82–89.

[2] Solomon W. Golomb. *Shift Register Sequences.* Revised Edition: Aegean Park Press, Laguna Hills 1982.

[3] Cees J. A. Jansen, Dick E. Boekee. The shortest feedback shift register that can generate a given sequence. CRYPTO '89, 90–99.