Feedback with Carry Shift Registers and Bent Sequences

MIDN 1/C Charles Celerier

March 2, 2012

Abstract

A stream cipher makes use of pseudorandom sequences to create a nearly perfect one-time pad. The feedback with carry shift register (FCSR) is a finite state machine that generates pseudorandom sequences for us in stream ciphers. This paper will investigate how bent functions can be used to produce pseudorandom sequences that possess non-linearity properites which will resist register synthesis attacks. The paper also considers the 2-adic span of "bent sequences" which are related to the size of the FCSR necessary to produce the sequence.

1 Introduction

Bytes of data, or short sequences of 1s and 0s, are exchanged between computer systems each day on public channels. Because gentlemen do read each other's mail, it is necessary to secure the communication of private data sent over public channels. The solution to this problem is solved by *cryptography*, the designing of systems to secure data exchanges over public channels.

The following example is the same basic communication scenario presented by Trappe in [19]. Let there be two parties, Alice and Bob, who wish to communicate with one another. A third party, Eve, is a potential eavesdropper. Alice wants to send a message, known as the *plaintext*, to Bob. To accomplish this without Eve knowing what the message is before it is received by Bob, Alice must *encrypt* her message by some prearranged method, usually involving an *encryption key*, to generate a related message called *ciphertext*. The idea is that the sent ciphertext, even if it is intercepted by Eve, will be too difficult to interpret and will conceal the plaintext message. Upon receipt of the ciphertext, Bob will *decrypt* the message, usually involving a *decryption key*, similar to the encryption of the message, and obtain the plaintext message. A visual of this scenario is presented in [**?**].

This scenario is the standard example found in many different introductory cryptography references. Cryptographers have created numerous encryption and decryption schemes, or *cryptosystems*, to secure the messages sent between Alice and Bob. Many of these systems have been broken because of the amount

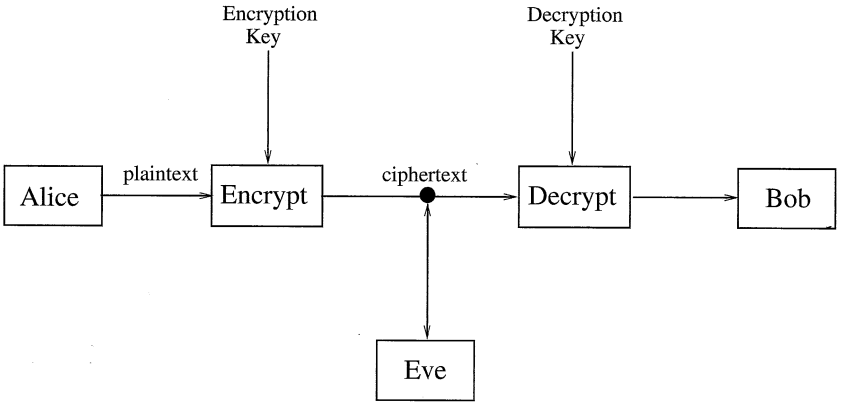

Figure 1: The Basic Comunication Scenario for Cryptography [19]

of work that goes into the study of breaking cryptosystems, called *cryptanalysis*. A constant battle exists between the designers and breakers of cryptosystems, strengthening designs and attacks every day. In fact, designing a strong cryptosystem typically requires knowledge of cryptography and cryptanalysis.

Before diving into the topic of this paper, there is an important assumption that must be presented. When designing a cryptosystem, every cryptographer assumes *Kerckhoffs' principle* [19]: "In assessing the security of a cryptosystem, one should always assume the enemy knows the method being used." The security of a cryptosystem cannot be based on the concealment of the encryption and decryption algorithms. In practice, the enemy can obtain the algorithms in many ways, including the defection or capture of people. The security must be based on the key and not the algorithm.

Imagine that Alice's message to Bob is a sequence of 1s and 0s, as it would be in the real world. Consider a cryptosystem which encrypts each bit in the sequence separately. This would be done by what is called a *stream cipher*, which, according to Rueppel, divides bit sequence into individual bits and enciphers each bit with a time-varying function whose time-dependency is governed by the internal state of the stream cipher [?]. The stream cipher can also be thought of in terms of a *keystream* which is a sequence of 1s and 0s the same length of the message that is added to the message using addition in $\mathbb{F}_2$ (also known as XOR). If the keystream was perfectly random, then the cryptosystem would be unbreakable, or *perfectly secret*, as discovered by Claude Shannon in his famous paper "Communication Theory of Secrecy Systems," written in 1945 and published in 1949. This cryptosystem is known as the *one-time pad*. Though it is perfectly secret, it can be difficult to implement because of the inability to produce perfectly random keystreams. Constructing a method to produce perfectly random sequences is a contradiction in itself.

Though it is impossible to create perfectly random sequences, it is possible

to get close. This type of sequence is called a *pseudorandom sequence*. These sequences have extremely long periods but are statistically indistinguishable from random sequences. This paper will discuss the construction of pseudorandom sequences using a particular finite state machine called the *feedback with carry shift register*. It will also investigate the possibility of incorporating *bent functions*, or perfectly non-linear functions, into the construction of psuedorandom sequences in an attempt to resist synthesizing algorithms.

2 Boolean Functions

This section will establish the definition of a *Boolean function* and introduce a few different tools used to measure properties of these functions. First, the finite field with two elements, denoted $\mathbb{F}_2$, is defined. The definitions and notations will follow those found in [3].

The two element field $(\mathbb{F}_2, \oplus, \cdot)$ is the set $\{0, 1\}$ with defined binary operations $\oplus$ and $\cdot$, also commonly referred to as the *XOR* and *AND* operators, respectively.

XOR	AND
$0 \oplus 0 := 0$	$0 \cdot 0 := 0$
$0 \oplus 1 := 1$	$0 \cdot 1 := 0$
$1 \oplus 0 := 1$	$1 \cdot 0 := 0$
$1 \oplus 1 := 0$	$1 \cdot 1 := 1$

Figure 2: Binary Operations for $\mathbb{F}_2$

It should be clear that $(\mathbb{F}_2, \oplus, \cdot)$ is a commutative ring with an identity. Additionally, the only non-zero element 1 is it's own inverse. Therefore $(\mathbb{F}_2, \oplus, \cdot)$ is a finite field. The n-dimensional vector space over $\mathbb{F}_2$ will be denoted by $\mathbb{F}_2^n$, with the usual inner product. Components of these vectors, the individual 1s and 0s, will be known as *bits*. For two vectors $x, y \in \mathbb{F}_2^n$ where $x = (x_0, \cdots, x_{n-1})$ and $y = (y_0, \cdots, y_{n-1})$, the scalar product in $\mathbb{F}_2^n$ will be defined as $x \cdot y \equiv x_0 \cdot y_0 \oplus \cdots \oplus x_{n-1} \cdot y_{n-1}$.

Example 1. Let $a, b \in \mathbb{F}_2^3$ such that $a = (1, 0, 1)$ and $b = (0, 1, 1)$ then

$$a + b = (1 \oplus 0, 0 \oplus 1, 1 \oplus 1) = (1, 1, 0)$$
$$a \cdot b = 1 \cdot 0 \oplus 0 \cdot 1 \oplus 1 \cdot 1 = 1$$

Each vector in $\mathbb{F}_2^n$ can be uniquely represented by an integer between 0 and 2^{n-1}. The binary representation function B is one-to-one.

$$B : \mathbb{F}_2^n \to \mathbb{N} \cup \{0\} \text{ such that } B(u) \equiv \sum_{i=0}^{n-1} u_i \cdot 2^i. \tag{1}$$

This definition of B has created the convention where the *low bit* or *least significant bit* appears on the left and the *high bit* or *most significant bit* appears

on the right. There is are two important functions which will be used extensively in this section. One counts the number of 1s in each vector and the other locates where those ones occur.

Definition 2.1. Let $u, v \in \mathbb{F}_2^n$. Then $wt : \mathbb{F}_2^n \to \mathbb{N} \cup \{0\}$ is defined by

$$wt(u) := \sum_{i=0}^{n-1} u_i$$

and $d : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{N} \cup \{0\}$ is defined by

$$d(u, v) := w(u + v).$$

Then $wt(u)$ is the *Hamming weight* of u and $d(u, v)$ is the *Hamming distance* between u and v.

Remark 2.1. The Hamming weight of a vector $u \in \mathbb{F}_2^n$ is the number of 1s in the vector, and the Hamming distance between two vectors $u, v \in \mathbb{F}_2^n$ is the number of bit differences between the two vectors.

Definition 2.2. Let $u \in \mathbb{F}_2^n$. Then $supp : \mathbb{F}_2^n \to \mathcal{P}(\mathbb{N} \cup \{0\})$ is defined by

$$supp(u) := \{i \in \mathbb{N} \cup \{0\} : u_i \neq 0\}$$

Example 2. Let $a, b, c \in \mathbb{F}_2^5$ such that

$$a = (0, 1, 1, 0, 1),\ b = (1, 1, 1, 0, 0),\ \text{and } c = (0, 0, 1, 1, 0).$$

Then,

$$\begin{array}{lll} wt(a) = 3 & supp(a) = \{1, 2, 4\} & d(a, b) = 2 \\ wt(b) = 3 & supp(b) = \{0, 1, 2\} & d(a, c) = 3 \\ wt(c) = 2 & supp(c) = \{2, 3\} & d(b, c) = 3. \end{array}$$

If we define the vectors $v_i \in \mathbb{F}_2^n$ such that $v_i = B^{-1}(i)$, then the sequence $(v_0, v_1, \cdots, v_{2^n-1})$ is said to be in *lexicographical order*. Using the convention of lexicographical ordering and the inner product in $\mathbb{F}_2^n$, a $2^n \times 2^n$ matrix can be written which captures all of the possible inner products of two vectors in $\mathbb{F}_2^n$. This is a matrix where $u_i \cdot v_j$ appears in the ith row and jth column. This particular matrix turns out to be a *Hadamard matrix* which will be studied later.

There is an interesting orthogonality property in the vectorspace $\mathbb{F}_2^n$ known as the *orthogonality principle* that every non-zero vector in $\mathbb{F}_2^n$ is orthogonal to exactly half of the vectors in the vectorspace.

Theorem 2.1. *Let $u \in \mathbb{F}_2^n$. Then*

$$\sum_{v \in \mathbb{F}_2^n} (-1)^{u \cdot v} = 2^n \text{ for } u = 0$$

$$= 0 \text{ otherwise.}$$

Proof. Let $u = 0 \in \mathbb{F}_2^n$. Then $\forall\ v \in \mathbb{F}_2^n$, $u \cdot v = 0$, so $(-1)^{u \cdot v} = 1$. Therefore $\sum_{v \in \mathbb{F}_2^n} (-1)^{u \cdot v} = |\mathbb{F}_2^n| = 2^n$.

Let $u \in \mathbb{F}_2^n$ where $u \neq 0$. Assume the ith bit of u is non-zero and define $e_i \in \mathbb{F}_2^n$ as a vector with all zero bits except for the ith bit which is 1. Then

$$\begin{aligned}\sum_{v \in \mathbb{F}_2^n} (-1)^{u \cdot v} &= \sum_{v \in \mathbb{F}_2^n} (-1)^{u \cdot (v + e_i)} \\ &= \sum_{v \in \mathbb{F}_2^n} (-1)^{u \cdot v} (-1)^{u \cdot e_i} \\ &= -\sum_{v \in \mathbb{F}_2^n} (-1)^{u \cdot v}.\end{aligned}$$

Therefore, $\sum_{v \in \mathbb{F}_2^n} (-1)^{u \cdot v} = -\sum_{v \in \mathbb{F}_2^n} (-1)^{u \cdot v}$, which implies $\sum_{v \in \mathbb{F}_2^n} (-1)^{u \cdot v} = 0$. □

When introduced to a new vector field, it is natural to begin looking at functions in that field. The particular function of interest here will be what is known as a *Boolean function*.

Definition 2.3. Any function f defined such that

$$f : \mathbb{F}_2^n \to \mathbb{F}_2$$

is a *Boolean function*.

The set of all Boolean function on n variables will be denoted by $\mathcal{BF}_n$

Boolean functions have been studied extensively, and there are various properties that are used to characterize them. The number of Boolean functions increases extremely rapidly as the number of variables increases.

$$|\{\mathcal{BF}_n : \mathbb{F}_2^n \to \mathbb{F}_2\}| = 2^{2^n} \tag{2}$$

As observed by Carlet, consider the set of all Boolean functions on 7 variables, and say that one nanosecond is spent at each function to identify the function and note some properties about it. If we visited every Boolean function this way, it would take 100 billions times the age of the universe to complete the search. For eight variables, there are more Boolean functions than there are atoms in the universe (10^{80}).

Here is an example of a Boolean function presented in a *truth table*.

The Hamming weight of f is the number of 1s that f has when evaluated at every point in the $\mathbb{F}_2^n$. Formally, this can be written

$$wt(f) = |\{u \in \mathbb{F}_2^n : f(u) = 1\}|.$$

A truth table is not a very efficient method to define a Boolean function. An algebraic definition of f is desirable. By writing f algebraically, more properties

x_0	x_1	x_2	x_3	$f(x_0, x_1, x_2, x_3)$
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	0	0
0	0	1	0	1
1	0	1	0	0
0	1	1	0	1
1	1	1	0	0
0	0	0	1	0
1	0	0	1	0
0	1	0	1	1
1	1	0	1	0
0	0	1	1	0
1	0	1	1	0
0	1	1	1	1
1	1	1	1	1

Table 1: Truth Table of f

of f will become apparent. As a first step toward defining f algebraically a function will be one-to-one and onto function will be defined which maps every f in $\mathcal{BF}_n$ to a vector in $\mathbb{F}_2^{2^n}$. This will be the function $V : \mathcal{BF}_n \to \mathbb{F}_2^{2^n}$ such that

$$V(f(u_0, \ldots, u_{n-1})) := (f(u_0), \ldots, f(u_{n-1})). \tag{3}$$

It is trivial to show that addition is homomorphic under V, in other words

$$V(f_1 \oplus f_2) = V(f_1) \oplus V(f_2).$$

Then a basis of $\mathbb{F}_2^{2n}$ must relate to an equivalent basis of $\mathcal{BF}_n$. Take the standard basis vectors of $\mathbb{F}_2^{2^n}$:

$$\begin{aligned} e_0 &= (1, 0, \ldots) \\ e_1 &= (0, 1, \ldots) \\ &\vdots \\ e_{2^n-1} &= (0, \ldots, 0, 1). \end{aligned}$$

Every vector of $\mathbb{F}_2^{2^n}$ can be written as a linear combination of these basis vectors. Equivalently, every Boolean function is a linear combination of the inverse maps of the basis vectors under V. Let $u \in \mathbb{F}_2^{2^n}$ and define $f = V^{-1}(u)$

and $f_i = V^{-1}(e_i)$. Then there exist a set of $c_i \in \mathbb{F}_2$ such that

$$\begin{aligned} u &= c_0 e_0 \oplus \cdots \oplus c_{2^n-1} e_{2^n-1} \\ \Rightarrow V^{-1}(u) &= c_0 V^{-1}(e_0) \oplus \cdots \oplus c_{2^n-1} V^{-1}(e_{2^n-1}) \\ \Rightarrow f &= c_0 f_0 \oplus \cdots \oplus c_{2^n-1} f_{2^n-1}. \end{aligned}$$

The Boolean fucntions $f_i = V^{-1}(e_i)$ are alled *atomic Boolean functions* because they are the building blocks of every Boolean function. Now the function f defined in Figure **??** can be written as a linear combination of the atomic Boolean functions in $\mathcal{BF}_4$. Since the coefficients in the linear combinations are either 0 or 1, it is true that every Boolean function can be written as the sum of $wt(f)$ atomic Boolean functions.

x_0	x_1	x_2	x_3	f	f_1	f_2	f_4	f_6	f_{10}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	0	0	0	0	0	0
0	1	0	0	1	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	1	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0
0	1	0	1	1	0	0	0	0	1	0	0
1	1	0	1	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	1	0
1	1	1	1	1	0	0	0	0	0	0	1

Table 2: f broken into atomic Boolean function in $\mathcal{BF}_4$

From Figure **??** the following equation should be clear:

$$f = f_1 + f_2 + f_4 + f_6 + f_{10} + f_{14} + f_{15}.$$

So far, there has been no discussion about the n-variable polynomical representation of a Boolean function. It however should be clear now that if we can write the atomic Boolean functions as polynomials, then the rest of the Boolean functions should follow. Though, it is not immediately obvious how to write the each atomic Boolean function as a polynomial in $\mathbb{F}_2[x_0, \cdots, x_{n-1}]/(x_0^2 \oplus x_1, \cdots, x_{n-1}^2 \oplus x_{n-1})$.

First, observe that $wt(f_i)$ always equals 1. Thus

$$\begin{aligned} f_i(u) &= 1 \quad u = B^{-1}(i) \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

It follows that

$$f_i = \Big(\prod_{j \in supp(f_i)} x_j\Big)\Big(\prod_{j \notin supp(f_i)} (1 \oplus x_j)\Big). \tag{4}$$

Equation 4 equals 1 at the vector $B^{-1}(i)$ and zero everywhere else. Continuing the example with the the function f will make this more apparent. The atomic functions which f is the sum of are written as polynomials below:

$$\begin{aligned}
f_1 &= (1 \oplus x_3)(1 \oplus x_2)(1 \oplus x_1)x_0 \\
&= x_0 \oplus x_1x_0 \oplus x_2x_0 \oplus x_2x_1x_0 \oplus x_3x_0 \oplus x_3x_1x_0 \oplus x_3x_2x_0 \oplus x_3x_2x_1x_0 \\
f_2 &= (1 \oplus x_3)(1 \oplus x_2)x_1(1 \oplus x_0) \\
f_4 &= (1 \oplus x_3)x_2(1 \oplus x_1)(1 \oplus x_0) \\
f_6 &= (1 \oplus x_3)x_2x_1(1 \oplus x_0) \\
f_{10} &= x_3(1 \oplus x_2)x_1(1 \oplus x_0) \\
f_{14} &= x_3x_2x_1(1 \oplus x_0) \\
f_{15} &= x_3x_2x_1x_0
\end{aligned}$$

Therefore,

$$f = x_0x_1x_2x_3 \oplus x_0x_1x_3 \oplus x_0x_3 \oplus x_0 \oplus x_1x_2x_3 \oplus x_1x_2 \oplus x_1 \oplus x_2x_3 \oplus x_2.$$

All of the n-variable atomic Boolean functions are linearly independent and act as a basis for $\mathcal{BF}_n$. It follows that each n-variable Boolean function is uniquely represeneted as a polynomial with coefficients in $\mathbb{F}_2$:

$$f(x) = \sum_{I \in \mathcal{P}(N)} a_I \Big(\prod_{i \in I} x_i\Big).$$

3 Bent Functions

Bent functions were originally defined in a paper by Rothaus in the Journal of Combinatorial Theory in 1976 [17]. These functions are useful for cryptographic applications because they are *perfectly nonlinear* which makes them resistant to differential attacks. The original defintion of the *bent function* is presented.

First, the discrete Fourier transform should be understood.

If $f(x) \in \mathcal{BF}_n$, so $(-1)^{f(x)}$ is well defined, then by the theory of discrete fourier transforms

$$(-1)^{f(x)} = \frac{1}{2^{n/2}} \sum_{\lambda \in \mathbb{F}_2^n} c(\lambda)(-1)^{\lambda \cdot x}$$

Where the $c(\lambda)$, the *Fourier coefficients* of $(-1)^{f(x)}$ are given by

$$c(\lambda) = \frac{1}{2^{n/2}} \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)}(-1)^{\lambda \cdot x}.$$

Definition 3.1. If all of the Fourier coefficients of $(-1)^{f(x)}$ are ± 1 then $f(x)$ is a *bent function*.

A simple bent function construction is accomplished by the Boolean functions in the *Maiorana-McFarland class*. This is the the set $\mathcal{M}$ which contains all Boolean function on $\mathbb{F}_2^n = \{(x, y) : x, y \in \mathbb{F}_2^n\}$, of the form:

$$f(x, y) = x \cdot \pi(y) \oplus g(y)$$

where π is any permutation on $\mathbb{F}_2^{n/2}$ and g any Boolean function on $\mathbb{F}_2^{n/2}$.

All function in the Maiorana-McFarland class of Boolean functions are bent.

4 Bent Sequences

Sequences generated using bent functions have nice cryptographic properties because of their perfect nonlinearity. These sequences can be generated multiple ways. Two easy examples are a filtering function on a shift register producing an m-sequence or a shift register which uses n different shift registers as input into a bent function. These two techniques are discussed by Carlet [2]. Both of constructions use input vectors from $gftwo^n$ in a psuedorandom order to generate the sequence. Before scrambling the input in this way, the sequences generated by lexicographic ordering of input vectors is considered.

Using the lexicographic ordering, the finite sequence generated will be the column of the outputs for the Boolean function read from the truth table of the Boolean fucntion. For example, the *finite Boolean sequence* using a lexicographic on the Boolean function f in 2 is

$$(0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1).$$

5 The Ring of N-adic Integers

The notation used in the definition of the N-adic numbers will follow the same notation used by Borevich and Shafarevich in Chapter 1 of **Number Theory**.

In this section, the set of N-adic integers is shown to be a commutative ring with an identity.

Definition 5.1. Let N be an integer. Then the infinite integer sequence $\{x_n\}$ determines an *N-adic integer* α, or $\{x_n\} \to \alpha$, if

$$x_{i+1} \equiv x_i \pmod{N^{i+1}} \quad \forall i \geq 0. \tag{5}$$

Two sequences $\{x_n\}$ and $\{x'_n\}$ determine the same N-adic integer if

$$x_i \equiv x'_i \pmod{N^{i+1}} \quad \forall i \geq 0. \tag{6}$$

The *set of all N-adic integers* will be denoted by $\mathbb{Z}_N$.

Each integer x is associated with a N-adic integer, determined by the sequence $\{x,\ x,\ \ldots,\ x,\ \ldots\}$. These integers will be called *rational integers* in the N-adic integers.

Example 3. Let $\{x_n\} \to \alpha \in \mathbb{Z}_3$. Then the first 5 terms of $\{x_n\}$ may look something like:

$$\begin{aligned}\{x_n\} &= \{1\ ,\ 1+2\cdot 3\ ,\ 1+2\cdot 3+1\cdot 3^2\ ,\\ &\qquad 1+2\cdot 3+1\cdot 3^2\ ,\ 1+2\cdot 3+1\cdot 3^2+1\cdot 3^4\ ,\ \ldots\}\\ &= \{1, 7, 16, 16, 97, \ldots\}\end{aligned}$$

Then equivalent sequences to $\{x_n\}$ could begin differently for the first few terms:

$$\begin{aligned}\{y_n\} &= \{4, 25, 16, 178, 583, \ldots\}\\ \{z_n\} &= \{-2, -47, 232, -308, 97, \ldots\}\end{aligned}$$

The sequences for $\{y_n\}$ and $\{z_n\}$ satisfy equation (5) for the first 5 terms, so they could be N-adic integers up to this point. Also, both are equivalent to $\{x_n\}$ according to the equivalence defined in equation (6).

$$\begin{aligned}1 &\equiv 4 \equiv 2 \pmod 3\\ 7 &\equiv 25 \equiv -47 \pmod{3^2}\\ 16 &\equiv 16 \equiv 232 \pmod{3^3}\\ 16 &\equiv 178 \equiv -308 \pmod{3^4}\\ 97 &\equiv 583 \equiv 97 \pmod{3^5}\end{aligned}$$

Therefore $\{x_n\}, \{y_n\}, \{z_n\} \to \alpha$.

Because there are infinitely many sequence representations for any N-adic integer, it is useful to define a canonical sequence to be used when writing N-adic integers as sequences.

Definition 5.2. For a given N-adic integer α, a given sequence $\{a_n\}$ with the properties:

i. $\{a_n\} \to \alpha$

ii. $\{a_n\} = \{a_0,\ a_0 + a_1\cdot N,\ \ldots,\ a_0 + \cdots + a_i\cdot N^i,\ \ldots\} : 0 \le a_i < N \quad \forall i \ge 0$

will be called *canonical*. The number $a_0a_1a_2\ldots a_i\ldots$ is the *digit representation* of α.

Example 4. In Example 3, the sequence $\{x_n\}$ was a canonical sequence that determined the N-adic integer α. A few more examples of canonical sequences determining 7-adic integers are given here:

$\beta = 3164\ldots$, then the canonical sequence $\{b_n\} \to \beta$ is

$$\begin{aligned}\{b_n\} &= \{3,\ 3+1\cdot 7,\ 3+1\cdot 7+6\cdot 7^2,\ 3+1\cdot 7+6\cdot 7^2+4\cdot 7^3,\ \ldots\}\\ &= \{3, 10, 304, 1676, \ldots\}\end{aligned}$$

$\gamma = 0164\ldots$, then the canonical sequence $\{c_n\} \to \gamma$ is

$$\begin{aligned}\{c_n\} &= \{0,\ 1\cdot 7,\ 1\cdot 7 + 6\cdot 7^2,\ 1\cdot 7 + 6\cdot 7^2 + 4\cdot 7^3,\ \ldots\} \\ &= \{0, 7, 301, 1673, \ldots\}\end{aligned}$$

$\delta = 5031\ldots$, then the canonical sequence $\{d_n\} \to \delta$ is

$$\begin{aligned}\{d_n\} &= \{5,\ 5, 5 + 3\cdot 7^2,\ 5 + 3\cdot 7^2 + 1\cdot 7^3,\ \ldots\} \\ &= \{5, 5, 152, 495, \ldots\}.\end{aligned}$$

Definition 5.3. Addition and multiplication in $\mathbb{Z}_N$ are done term by term. Let $\alpha, \beta \in \mathbb{Z}_N$ and $\{x_n\} \to \alpha, \{y_n\} \to \beta$. Then,

$$\begin{aligned}\{x_n\} + \{y_n\} &:= \{x_0 + y_0, x_1 + y_1, \ldots\} \to \alpha + \beta \\ \{x_n\} \cdot \{y_n\} &:= \{x_0 \cdot y_0, x_1 \cdot y_1, \ldots\} \to \alpha \cdot \beta\end{aligned}$$

Define $\{0, 0, 0, \ldots\} \to 0 \in \mathbb{Z}_N$ and $\{1, 1, 1, \ldots\} \to 1 \in \mathbb{Z}_N$

Lemma 5.1. *For $\alpha \in \mathbb{Z}_N$, $\alpha + 0 = 0 + \alpha = \alpha$ and $1 \cdot \alpha = \alpha \cdot 1 = \alpha$.*

Proof. Let $\{x_n\} \to \alpha \in \mathbb{Z}_N$.

$$\begin{aligned}\{x_n\} + \{0, 0, \ldots\} &= \{x_0 + 0, x_1 + 0, \ldots, x_i + 0, \ldots\} \\ &= \{x_0, \ldots, x_i, \ldots\}. \\ \{0, 0, \ldots\} + \{x_n\} &= \{0 + x_0, 0 + x_1, \ldots, 0 + x_i, \ldots\} \\ &= \{x_0, \ldots, x_i, \ldots\}.\end{aligned}$$

$\{x_n\} = \{x_n\} + \{0, 0, \ldots\} = \{0, 0, \ldots\} + \{x_n\}$ implies $\alpha = \alpha + 0 = 0 + \alpha$. Therefore, the additive identity in $\mathbb{Z}_N$ is 0.

$$\begin{aligned}\{x_n\} \cdot \{1, 1, \ldots\} &= \{x_0 \cdot 1, x_1 \cdot 1, \ldots, x_i \cdot 1, \ldots\} \\ &= \{x_0, \ldots, x_i, \ldots\}. \\ \{1, 1, \ldots\} \cdot \{x_n\} &= \{1 \cdot x_0, 1 \cdot x_1, \ldots, 1 \cdot x_i, \ldots\} \\ &= \{x_0, \ldots, x_i, \ldots\}.\end{aligned}$$

$\{x_n\} = \{x_n\} \cdot \{1, 1, \ldots\} = \{1, 1, \ldots\} \cdot \{x_n\}$ implies $\alpha = \alpha \cdot 1 = 1 \cdot \alpha$. Therefore, the multiplicative identity in $\mathbb{Z}_N$ is 1. □

Finally, this paper defines a *ring* accroding to Fine, Gaglione, and Roesenberger and shows that $\mathbb{Z}_N$ is a *commutative ring with an identity.*

Definition 5.4. A *ring* is a set R with two binary operations defined on it. These are usually called addition denoted by $+$, and multiplication denoted by $\cdot$ or juxtaposition, satisfying the following six axioms:

1. Addition is commutative: $a + b = b + a \ \ \forall a, b \in R$.
2. Addition is associative: $a + (b + c) = (a + b) + c \ \ \forall a, b, c \in R$.
3. There exists an additive identity, denoted by 0, such that $a+0 = a \ \ \forall a \in R$.
4. $\forall a \in R$ there exists an additive inverse, denoted by $-a$, such that $a + (-a) = 0$.
5. Multiplication is associative: $a(bc) = (ab)c \ \forall a, b, c \in R$
6. Multiplication is left and right distributive over addition:

$$a(b + c) = ab + ac$$
$$(b + c)a = ba + ca$$

If it is also true that

7. Multiplication is commutative: $ab = ba \ \ \forall a, b \in R$, then R is a *commutative ring*.

Further if

8. There exists a multiplicative identity denoted by 1 such that $a \cdot 1 = a$ and $1 \cdot a = a \ \ \forall a \in R$, then R is a *ring with an identity*.

If R satisfies all eight properties, then R is a *commutative ring with an identity*.

Theorem 5.1. *$\mathbb{Z}_N$ is a commutative ring with an identity.*

Proof. Let $\{x_n\}, \{y_n\}, \{z_n\}$ determine $\alpha, \beta, \gamma \in \mathbb{Z}_N$ respectively. Then

1. *Commutativity of Addition*

$$\begin{aligned}\{x_n\} + \{y_n\} &= \{x_0 + y_0, \dots, x_i + y_i, \dots\} \\ &= \{y_0 + x_0, \dots, y_i + x_i, \dots\} \\ &= \{y_n\} + \{x_n\}.\end{aligned}$$

   $\{x_n\} + \{y_n\} \to \alpha + \beta$ and $\{x_n\} + \{y_n\} = \{y_n\} + \{x_n\} \to \beta + \alpha$. Therefore, by Definition 5.1, $\alpha + \beta = \beta + \alpha$.

2. *Associativity of Addition*

$$\begin{aligned}\{x_n\} + (\{y_n\} + \{z_n\}) &= \{x_n\} + \{y_0 + z_0, \dots, y_i + z_i, \dots\} \\ &= \{x_0 + (y_0 + z_0), \dots, x_i + (y_i + z_i), \dots\} \\ &= \{(x_0 + y_0) + z_0, \dots, (x_i + y_i) + z_i, \dots\} \\ &= \{x_0 + y_0, \dots, x_i + y_i, \dots\} + \{z_n\} \\ &= (\{x_n\} + \{y_n\}) + \{z_n\}.\end{aligned}$$

   Therefore, $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$.

3. *Existence of the Additive Identity*

   By Lemma 5.1, 0 is the additive identity.

4. *Existence of Additive Inverses*

   Define $-\{x_n\} = \{p - x_0, p^2 - x_1, \ldots, p^{i+1} - x_i, \ldots\} \to -\alpha$. Then

$$\begin{aligned}\{x_n\} + (-\{x_n\}) &= \{x_0 + p - x_0, x_1 + p^2 - x_1, \ldots, x_i + p^{i+1} - x_i, \ldots\} \\ &= \{p, p^2, \ldots, p^{i+1}, \ldots\} \\ &\equiv \{0, 0, \ldots\} \\ &= 0.\end{aligned}$$

   Therefore, $\alpha + (-\alpha) = 0$.

5. *Associativity of Multiplication*

$$\begin{aligned}\{x_n\}(\{y_n\}\{z_n\}) &= \{x_n\}\{y_0 z_0, \ldots, y_i z_i, \ldots\} \\ &= \{x_0(y_0 z_0), \ldots, x_i(y_i z_i), \ldots\} \\ &= \{(x_0 y_0) z_0, \ldots, (x_i y_i) z_i, \ldots\} \\ &= \{x_0 y_0, \ldots, x_i y_i, \ldots\}\{z_n\} \\ &= (\{x_n\}\{y_n\})\{z_n\}.\end{aligned}$$

   Therefore, $\alpha(\beta\gamma) = (\alpha\beta)\gamma$.

7. *Commutativity of Multiplication*

$$\begin{aligned}\{x_n\}\{y_n\} &= \{x_0 y_0, \ldots, x_i y_i, \ldots\} \\ &= \{y_0 x_0, \ldots, y_i x_i, \ldots\} \\ &= \{y_n\}\{x_n\}.\end{aligned}$$

   Therefore, $\alpha\beta = \beta\alpha$.

6. *Left and right distributivity of multiplication over addition*

$$\begin{aligned}\{x_n\}(\{y_n\} + \{z_n\}) &= \{x_n\}\{y_0 + z_0, \ldots, y_i + z_i, \ldots\} \\ &= \{x_0(y_0 + z_0), \ldots, x_i(y_i + z_i), \ldots\} \\ &= \{x_0 y_0 + x_0 z_0, \ldots, x_i y_i + x_i z_i, \ldots\} \\ &= \{x_n\}\{y_n\} + \{x_n\}\{z_n\}.\end{aligned}$$

   By commutativity of multiplication,

$$\begin{aligned}(\{y_n\} + \{z_n\})\{x_n\} &= \{x_n\}(\{y_n\} + \{z_n\}) \\ &= \{x_n\}\{y_n\} + \{x_n\}\{z_n\} \\ &= \{y_n\}\{x_n\} + \{z_n\}\{x_n\}.\end{aligned}$$

   Therefore, $\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$ and $(\beta + \gamma)\alpha = \beta\alpha + \gamma\alpha$.

8. *Existence of a multiplicative identity*

   By Lemma 5.1, 1 is the multiplicative identity.

   Properties 1 through 8 from Definition 5.4 are satisfied, so $\mathbb{Z}_N$ is a commutative ring with an identity.

□

Theorem 5.2. *An N-adic integer α, which is determined by a sequence $\{x_n\}$, is a unit if and only if x_0 is relatively prime to N.*

Proof. Let α be a unit. Then there is an N-adic integer β such that $\alpha\beta = 1$. If β is determined by the sequence $\{y_n\}$, then

$$x_i y_i \equiv 1 \pmod{N^{i+1}} \quad \forall i \geq 0. \tag{7}$$

In particular, $x_0 y_0 \equiv 1 \pmod{N}$ and hence $x_0 \not\equiv 0 \bmod N$. Thus, x_0 is relatively prime to N. Conversely, let x_0 be relatively prime to N. Then $x_0 \not\equiv 0 \pmod{N}$. From (5)

$$\begin{aligned} x_1 &\equiv x_0 \pmod{N} \\ &\vdots \\ x_{i+1} &\equiv x_i \pmod{N^i}. \end{aligned}$$

Working backward, $x_{i+1} \equiv x_i \equiv \cdots \equiv x_1 \equiv x_0 \pmod{N}$. Thus, x_i is relatively prime to $p \ \forall i \geq 0$, which implies x_i is relatively prime to N^{i+1}. Consequently, $\forall i \geq 0 \ \exists y_i$ such that $x_i y_i \equiv 1 \pmod{N^{i+1}}$. Since $x_{i+1} \equiv x_i \pmod{p}^i$ and $x_{i+1} y_{i+1} \equiv x_i y_i \pmod{N^i}$. Then, $y_{i+1} \equiv y_i \pmod{N^i}$. Therefore the sequence $\{y_n\}$ determines some N-adic integer β. Because $x_i y_i \equiv 1 \pmod{N^{i+1}} \ \forall i \geq 0$, $\alpha\beta = 1$. This means α is a unit. □

From this theorem it follows that a rational integer $a \in \mathbb{Z}_N$ is a unit if and only if a is relatively prime to N. If this condition holds, then $a^{-1} \in \mathbb{Z}_N$. Then for any rational integer $b \in \mathbb{Z}_N$, $b/a = a^{-1}b \in \mathbb{Z}_N$.

6 Constructing Sequences Determing $\frac{b}{a}$ in $\mathbb{Z}_N$

For any rational number b/a, a relatively prime to N, there exists a sequence $\{x_n\} \to b/a \in \mathbb{Z}_N$. At this point, it is worth using the digit representation for integers in $\mathbb{Z}_N$. So $\{x_n\} = \{x_0, \ x_0 + x_1 N, \ \ldots, \ x_0 + \cdots + x_i N^i, \ \ldots\}$ and $b/a = x_0 x_1 \ldots x_i \ldots$. Rather than finding $a^{-1} \pmod{N}^{i+1}$ to determine each x_i, it is not too difficult for every i to find $\sum_{k=0}^{i} x_k N^k$ such that

$$b \equiv a \sum_{k=0}^{i} x_k N^k \pmod{N^{i+1}}. \tag{8}$$

Then,

$$x_i = \frac{\sum_{k=0}^{i} x_k N^k - \sum_{k=0}^{i-1} x_k N^k}{N^k}. \tag{9}$$

Nearly all of the digits for any rational number in $\mathbb{Z}_N$ can also be found using powers of N^{-1}, which is much simpler to analyze than the brute force search for the digits mentioned above.

Theorem 6.1. *Let $u_0, q, N \in \mathbb{Z}$, where q is relatively prime to N, $|u_0| < q$, and $q = -q_0 + \sum_{i=1}^{r} q_i N^i$ for $0 \leq q_i < N$. Define $\alpha = u_0/q \in \mathbb{Z}_N$ such that $\alpha = \sum_{i=0}^{\infty} a_i N^i$ for $0 \leq a_i < N$. Also, define $u_k \in \mathbb{Z}$ such that $u_k/q = \sum_{i=k}^{\infty} a_i N^{i-k} \in \mathbb{Z}_N$ and $\gamma \equiv N^{-1} \pmod{q}$. Then, there exist u_k for every $k \geq 0$ such that*

$$a_k \equiv q^{-1} u_k \pmod{N}. \tag{10}$$

If $-q < u_0 < 0$, then $u_k \in \{-q, \ldots, -1\}$ for $k \geq 0$. Otherwise, for $k > \lfloor \log_N(q) \rfloor = r$, $u_k \in \{-q, \ldots, -1\}$

Let $\omega \in \{-q, \ldots, -1\}$ such that $\omega \equiv \gamma^k u_0 \pmod{q}$. Then for $k > \lfloor \log_N(q) \rfloor = r$, or if $-q < u_0 < 0$, then $k \geq 0$,

$$a_k \equiv q^{-1} \omega \pmod{N}. \tag{11}$$

Proof. Write u_0/q in terms of u_k.

$$\begin{aligned} \frac{u_0}{q} &= a_0 + N\frac{u_1}{q} = a_0 + a_1 N + N^2 \frac{u_2}{q} = \ldots \\ &= \sum_{i=0}^{k-1} a_i N^i + N^k \frac{u_k}{q} \quad \forall k \geq 1. \end{aligned} \tag{12}$$

Rewrite (12) to be

$$p^k u_k = u_0 - q\left(\sum_{i=0}^{k-1} a_i p^i \right) \quad \forall k \geq 1 \tag{13}$$

Then $|u_0| < q$ and $0 \leq a_i < p$ from the assumptions and equation (13). These imply for all $k \geq 1$, $|u_0| = |q \sum_{i=0}^{k-1} a_i p^i + p^k u_k| < q$. Then,

$$\begin{aligned} -q - q\sum_{i=0}^{k-1} a_i p^i < p^k u_k < q - q\sum_{i=0}^{k-1} a_i p^i \\ \Rightarrow -q\left(\frac{1 + \sum_{i=0}^{k-1} a_i p^i}{p^k}\right) < u_k < q\left(\frac{1 - \sum_{i=0}^{k-1} a_i p^i}{p^k}\right). \end{aligned}$$

u_k may only greater than zero when $\frac{1-\sum_{i=0}^{k-1} a_i p^i}{p^k}$ greater than zero. This only occurs when the sequence $[a_0, \ldots, a_j] = [0, \ldots, 0]$ for $j \geq 0$. Such a sequence occurs if and only if $u_0 \geq 0$ and $u_0 \equiv 0 \pmod{p}^i$ for $0 \leq i \leq j$, $j \geq 0$. This is clear from the construction of p-adic sequences for rational numbers. Therefore u_k may only be greater than zero if $u_0 \geq 0$ and $u_0 \equiv 0 \pmod{p}^i$ for $0 \leq i \leq j$, $j \geq 0$. The lower bound is greater than $-q$. This clear because $\frac{1+\sum_{i=0}^{k-1} a_i p^i}{p^k} \leq 1$. Therefore,

$$-q < u_k < 0 \; for \; -q < u_0 < 0.$$

If $0 \geq u_0 < q$, then the upper bound remains unchanged.

$$-q < u_k < q\left(\frac{1-\sum_{i=0}^{k-1} a_i p^i}{p^k}\right) \text{ for } 0 \leq u_0 < q$$

There is still work to be done on the upper bound.

$$\begin{aligned} 0 \leq \sum_{i=0}^{k-1} a_i p^i < p^k \; for \; k \geq 1 \\ \Rightarrow -q(\sum_{i=0}^{k-1} a_i p^i) \leq 0 \\ \Rightarrow u_0 - q(\sum_{i=0}^{k-1} a_i p^i) < q \\ \Rightarrow p^k u_k < q \\ \Rightarrow u_k < \frac{q}{p^k}. \end{aligned}$$

For $k > \lfloor \log_p(q) \rfloor = r$, $|q/p^k| < 1$. Therefore, $-q < u_k < 0$ for $0 \leq u_0 < q$ and $k > r$. Further lowering the upperbound, if $u_k = 0$, then $u_0/q = \sum_{i=0}^{k-1} a_i p^i + 0$. This implies u_0/q is a rational integer, which is not true. Noting finally that u_k must be an integer. If $|u_0| < q$ and $u_0 < 0$, or $|u_0| < q$, $u_0 \geq 0$, and $k > \lfloor \log_p(q) \rfloor = r$, then

$$u_k \in \{-q, \ldots, -1\}.$$

It has now been shown for certain restrictions u_k belongs to a specific set of representatives for the residue classes of $\mathbb{Z}/(q)$. Define $\gamma \equiv p^{-1} \pmod{q}$. Reducing (13) modulo q shows that

$$u_k \equiv \gamma u_{k-1} \pmod{q}. \tag{14}$$

Since this is true for all k greater than or equal to 1, it is clear that

$$u_k \equiv \gamma^k u_0 \pmod{q}. \tag{15}$$

Reducing (13) modulo p shows that

$$a_k \equiv q^{-1} u_k \pmod{p}. \tag{16}$$

Define $\omega \equiv \gamma^k u_0 \pmod{q}$, and $\rho \equiv q^{-1} \bmod p$. Finally, if $|u_0| < q$ and $u_0 < 0$, or $|u_0| < q$, $u_0 \geq 0$, and $k > \lfloor \log_p(q) \rfloor = r$, then

$$a_k \equiv \rho\omega \pmod{p}. \tag{17}$$

□

Corollary 6.1. *Let $0 \leq u_0 < q$. Define j to be the greatest integer such that $u_0 \equiv 0 \pmod{p}^j$. Then the following are true:*

i. $j \leq \lfloor \log_p q \rfloor = r$

ii. $[a_0, \ldots, a_{j-1}] = [0, \ldots, 0]$

iii. $u_k > 0$ *for* $k = j$

iv. $u_k \not\equiv 0 \pmod{p}$

The results of this corollary are straightforward.

Theorem 6.1 shows that for $-q < u_0 < 0$, there is a sequence of numerators $\{u_k\}$ directly related to the sequence of digits $\{a_k\}$ for $u_0/q \in \{z_n\}$. This is provides a more powerful tool for the analysis of the sequences generated by AFSRs. The results shown here fill in the gaps of the incorrect proof shown in Theorem 10 of Klapper and Xu's paper.

7 The p-adic Metric and Completeness

To prove completeness of the set of p-adic, we will begin by introducing a valuation function. From this point we will define the p-adic metric and concluding with a proof showing that $\mathbb{Z}_p$ satisfies the Cauchy criterion.

Definition 7.1. Let $\alpha = \{a_n\} \in \mathbb{Z}_p \setminus \{0\}$. If m is the smallest number in $\mathbb{N}$ such that $x_m \not\equiv 0 \pmod{p}^{m+1}$, then the *valuation* of α is m, or $v_p(\alpha) = m$.

If $\alpha = 0$, then $v_p(\alpha) = \infty$.

Definition 7.2. If $\alpha \in \mathbb{Z}_p$, then the *p-adic norm* of α is $\|\alpha\|_p = p^{-m}$ where $m = v_p(\alpha)$.

Definition 7.3. A sequence $\{\alpha_n\}$, where $\alpha_i \in \mathbb{Z}_p$, *converges* if $\exists \alpha \in \mathbb{Z}_p$ *s.t.* $\forall\, \epsilon > 0\ \exists N \in \mathbb{N}$ *s.t.* $\forall\, n \geq N \Rightarrow \|\alpha_n - \alpha\|_p < \epsilon$.

8 Finite State Machines

It is interesting to consider FCSRs in the context of general finite state machines. In 1967, Dr. Solomon W. Golomb published *Shift Register Sequences* establishing a definition of finite state machines and shift registers used in much of the literature today. This section will begin by defining the finite state machine according to Golomb and move toward the definition of generalized shift registers. In the next section, FCSRs will be defined with Golomb's style in mind.

Definition 8.1. A *finite state machine* consists of a finite collection of *states* K, sequentially accepts a sequence of *inputs* from a finite set A, and produces a sequence of *outputs* from a finite set B. Moreover, there is an *output function* μ which computes the present output as a fixed function of present input and present state, and a *next state function* δ which computes the next states as a fixed function of present input and present state. In a more mathematical manner, μ and δ are defined such that

$$\mu : K \times A \to B \qquad \mu(k_n, a_n) = b_n \tag{18}$$

$$\delta : K \times A \to K \qquad \delta(k_n, a_n) = k_{n+1} \tag{19}$$

Golomb presents two important theorems about these machines. Both deal with the periodicity of any finite state machine, which include FCSRs. The theorems and proofs are presented here.

Theorem 8.1. *If the input to a finite state machine is eventually constant, then the output is eventually periodic.*

Proof. Let t be the time when the input becomes constant, so $a_t = a_{t+1} = \ldots$. Because K is a finite collection of states, there exists times $r > s > t$ such that $k_r = k_s$. Then, by induction, $\forall i > 0$,

$$k_{r+i+1} = \delta(k_{r+i}, a_{r+i}) = \delta(k_{s+i}, a_{s+i}) = k_{s+i+1}$$

Therefore,

$$b_{r+i+1} = \mu(k_{r+i+1}, a_{r+i+1}) = \delta(k_{s+i+1}, a_{s+i+1}) = b_{s+i+1}$$

Thus, the eventual period of this machine is $r - s$. □

Theorem 8.2. *If the input sequence to a finite state machine is eventually periodic, then the output sequence is eventually periodic.*

Proof. Let p be the period of the inputs once the machine becomes periodic at time t. Then, for $h > 0$ and $c > t$, $a_c = a_{c+hp}$. Similar to the proof of Theorem 8.1, using the fact that K is finite, there must be $r > s > t$ such that, for some $h > 0$,

$$k_{r+1} = \delta(k_r, a_r) = \delta(k_s, a_{r+hp}) = k_{s+1}.$$

It should also be clear that $a_{r+i} = a_{r+i+hp}$ for $h > 0$. So by induction, $\forall i > 0$

$$k_{r+i+1} = \delta(k_{r+i}, a_{r+i}) = \delta(k_{s+i}, a_{r+i+hp}) = k_{s+i+1}$$

Finally, this proves $b_{r+i+1} = b_{s+i+1}$. Thus, the eventual period of this machine is $r - s$. □

The next object defined is called an N-ary n-stage machine. It can be used to represent any finite state machine. It is also a natural generalization of shift registers, so thinking of finite state machines in the context of N-ary n-state machines will make the transition to talking about shift registers much smoother.

Definition 8.2. Choose $n, m, r \in \mathbb{N}$. An *N-ary n-stage machine* consists of the following:

1. $D = \{0, \ldots, N-1\}$. This set contains the *N-ary digits* of the machine.
2. $K = \{\sum_{i=0}^{n} x_i N^i : x_i \in D\}$. This set contains the *N-ary states* of the machine.
3. $A = \{\sum_{i=0}^{m} y_i N^i : y_i \in D\}$. This set contains the *N-ary inputs* of the machine.
4. $B = \{\sum_{i=0}^{r} z_i N^i : z_i \in D\}$. This set contains the *N-ary outputs* of the machine.
5. $F = \{f_i(x_0, \ldots, x_n, y_0, \ldots, y_m) : 0 \leq i < n\}$. This set contains the *N-ary next state functions* of the machine.
6. $G = \{g_i(x_0, \ldots, x_n, y_0, \ldots, y_m) : 0 \leq i < r\}$. This set contains the *N-ary output functions* of the machine.

The next state and output are determined from the current state and input by the following equations:

$$x_i^* = f_i(x_0, \ldots, x_n, y_0, \ldots, y_m) \quad 0 \leq i < n \tag{20}$$
$$z_i = g_i(x_0, \ldots, x_n, y_0, \ldots, y_m) \quad 0 \leq i < r \tag{21}$$

9 Feedback with Carry Shift Registers

A *feedback with carry shift register* is a feedback shift register which uses a linear combination each state toA description of N-ary feedback with carry shift registers is defined here. The definition of the register follows the one given in Andrew Klapper's book.

Definition 9.1. Let $q_0, q_1, \ldots, q_m \in \mathbb{Z}/(p)$ for $p \in \mathbb{Z}$ and assume that $q_0 \not\equiv 0 \pmod{p}$. An *algebraic feedback shift register* (or *AFSR*) over ($\mathbb{Z}$,p,S) of length

m with *multipliers* or *taps* $q_0, q_1, \ldots, q_m$ is a discrete state machine whose states are collections

$$(a_0, a_1, \ldots, a_{m-1}; z) \text{ where } a_i \in S \text{ and } z \in \mathbb{Z}$$

consisting of cell contents a_i and memory z. The state changes according to the following rules:

1. Compute
$$\sigma = \sum_{i=1}^{m} q_i a_{m-i} + z.$$
2. Find $a_m \in S$ such that $-q_0 a_m \equiv \sigma \pmod{p}$. That is $a_m \equiv -q_0^{-1}\sigma \pmod{p}$.
3. Replace $(a_0, \ldots, a_{m-1})$ by $(a_1, \ldots, a_m)$ and replace z by $\sigma(\text{div}p) = (\sigma + q_0 a_m)/p$.

10 Xu's Rational Approximation Algorithm

In Andrew Klapper's book, he roughly describes Xu's rational approximation algorithm for π-adic sequences in any ring R. A description of Xu's Algorithm is presented here in the context of the ring $\mathbb{Z}_p$, where p does not have to be prime.

The rational approximation problem is presented here: Given the eventually periodic digit representation of $\frac{a}{b} \in \mathbb{Z}_p$, find a and b.

If there are no constraints on a and b, then the entire sequence must be known to solve the problem. Specifically, a single algebraic feedback shift register can not produce all $\frac{a}{b} \in \mathbb{Z}_p$.

References

[1] Z. I. Borevich and I. R. Shafarevich, *Number theory*, Academic Press, 1966.

[2] C. Carlet, *Boolean functions for cryptography and error correcting codes*, Boolean Methods and Models (Y. Crama and P. L. Hammer, eds.), Cambridge University Press, 2006.

[3] T. W. Cusik and P. Stănică, *Cryptographic boolean functions and applications*, Elsevier, 2009.

[4] J. Đ. Golić, *Recent advances in stream cipher cryptanalysis*, Publications De L'Institut Mathématique **64** (1998), 183–204.

[5] S. W. Golomb, *Shift register sequences*, Aegean Park Press, 1982.

[6] M. Goresky and A. Klapper, *Algebraic shift register sequences*, Cambridge University Press, 2012.

[7] A. Klapper and M. Goresky, *Cryptoanalysis based on 2-adic rational approximation*, CRYPTO, 1995, pp. 262–273.

[8] ———, *Feedback shift registers, 2-adic span, and combiners with memory*, Journal of Cryptology **10** (1997), 111–147.

[9] ———, *Arithmetic correlations and walsh transforms*, IEEE Transactions on Information Theory **58** (2012), no. 1, 479–492.

[10] A. Klapper and J. Xu, *Algebraic feedback shift registers*, Theor. Comput. Sci. **226** (1999), no. 1-2, 61–92.

[11] ———, *Register synthesis for algebraic feedback shift registers based on non-primes*, Des. Codes Cryptography **31** (2004), no. 3, 227–250.

[12] N. Koblitz, *p-adic numbers, p-adic analysis, and zeta-functions*, Springer-Verlag, 1977.

[13] J. R. Munkres, *Topology: A first course*, Prentice Hall, 1975.

[14] T. Neumann, *Bent functions*, Ph.D. thesis, University of Kaiserslautern, May 2006.

[15] N. Nisan and M. Szegedy, *On the degree of boolean functions as real polynomials*, Computational Complexity **4** (1994), 301–313.

[16] R. A. Reuppel, *Analysis and design of stream ciphers*, Springer-Verlag, 1986.

[17] O. S. Rothaus, *On b̈entf̈unctions*, Journal of Combinatorial Theory **20** (1976), no. 3, 300–305.

[18] W. J. Townsend and M. A. Thornton, *Walsh spectrum computations using cayley graphs*, IEEE Midwest Symposium on Circuits and Systems, August 2001, pp. 110–113.

[19] W. Trappe and L. C. Washington, *Introduction to cryptography with coding theory*, 2nd ed., Pearson Education, 2006.