will be considered in this chapter. This restriction allows a simplification
because of the dominating practical importance of binary arithmetic
that the decoder can be considered effective for practical purposes.
reasons much more clearly than the arithmetic unit which it monitors. So
that it is possible to build the "decoder" which corrects or detects erroneous
This chapter is intended to summarize the most important results which

1. INTRODUCTION

OSCAR N. GARCIA

and

JAMES L. MAKESY

COMPUTER ARITHMETIC
ERROR-CORRECTING CODES

(Ed.) J. T. Tou
v 1. 4
ADVANCES IN INFORMATION SYSTEMS SCIENCE
Reported from
where each coefficient $a_i$ is either 0 or 1. When $a_i = 0$, $f_{i}$ is the stair-case function of the type

$$f_{i} = \begin{cases} 1 & \text{if } i \leq \frac{n}{2} \\
0 & \text{if } i > \frac{n}{2} \end{cases}$$

The right-hand side of the equality is the expression for $f_i$ of the type

$$f = f_1 + f_2 + \ldots + f_n$$

where

$$f = \begin{cases} 1 & \text{if } i \leq \frac{n}{2} \\
0 & \text{if } i > \frac{n}{2} \end{cases}$$

An important step in the design of any code system is the determination of the generator. This is the process of finding the generating function of the code. The generator function is a polynomial that represents the code. The coefficients of the polynomial are the elements of the code.

1.2 Arithmetic Weight

The arithmetic weight of a code is the number of 1's in the binary representation of a code word. The arithmetic weight of a code is important in error detection and correction. A code with low arithmetic weight is more likely to be detected and corrected. A code with high arithmetic weight is more likely to be undetected and uncorrected.

For the sake of completeness, we will show in this section how to determine the generator polynomial of a code. The generator polynomial is a polynomial that represents the code. The coefficients of the polynomial are the elements of the code.

The generator polynomial is obtained by finding the greatest common divisor (GCD) of the code word polynomials. The GCD is the polynomial that is the greatest common divisor of all the code word polynomials.

The GCD of the code word polynomials is found by using the Euclidean algorithm. The Euclidean algorithm is a method for finding the GCD of two polynomials. The algorithm is based on the fact that the GCD of two polynomials is the same as the GCD of the remainder of the division of one polynomial by the other.

The Euclidean algorithm is as follows:

1. Divide the first polynomial by the second polynomial.
2. Use the remainder of the division as the new dividend.
3. Repeat the process until the remainder is zero.
4. The last non-zero remainder is the GCD of the two polynomials.

The generator polynomial is the GCD of the code word polynomials. The generator polynomial is a polynomial that represents the code. The coefficients of the polynomial are the elements of the code.

The generator polynomial is used to determine the code word polynomials. The code word polynomials are the polynomials that represent the code words.

The code word polynomials are determined by multiplying the generator polynomial by the code word polynomials.

The code word polynomials are used to determine the code word. The code word is the polynomial that represents the code word. The code word is the polynomial that represents the information that is to be transmitted.

The code word is the polynomial that represents the information that is to be transmitted.

The code word is the polynomial that represents the information that is to be transmitted.
Form of $F$ is defined as follows and the number of places required for the radix-2 form of $F$ is denoted by $\nu F$. The number of places required for the NAF of $F$ is $\nu F$. The number of places required for the NAF of $\overline{F}$ is $\nu \overline{F}$.

**Theorem 1.1.** If $F$ and $\overline{F}$ are both even or both odd, then

$$\nu F = \nu \overline{F}$$

Proof. We note that $F$ and $\overline{F}$ are both even or both odd, so that

$$\nu F = \nu \overline{F}$$

are the coefficients in the NAF for $F$.

**Example 1.** $F = 187_{10} = \overline{F}$

The following theorem due to Crandall and Ruengcharoen (1994) shows that the number of additions required for the NAF of $F$ can be obtained by a single table lookup. The NAF of $F$ can be obtained by the following table lookup. The output of the table is $\nu F$.

**Theorem 1.1.** Let $F = [a_n a_{n-1} \ldots a_1 a_0]$ be the NAF of $F$. Then $2^{\nu F} \equiv F \pmod{2^n}$.

**Proof.** Let $F$ and $\overline{F}$ denote the respective coefficients in two non-negative integers.

**Theorem 1.1.** Every integer $F$ has a most compact form.

**Definition 1.** The arithmetic weight of the integer $F$ is denoted $W(F)$.
\[ (7) \quad (1 + a) = (1 + a) \]

**Definition 1.2.** The arithmetic distance between the integers \( a \) and \( b \) is the smallest non-negative integer \( d \) such that \( a = b + d \) or \( a = b - d \).

Theorem 1.1. For any non-zero integers \( a \) and \( b \), there exists a unique integer \( d \) such that \( a = b + d \) or \( a = b - d \).

**Algorithm 1.**

1. **Initialization:** Let \( M = \mathbb{Z} \) be the set of integers.
2. **Input:** \( a, b \in M \).
3. **Output:** \( d = a - b \).
4. **Procedure:**
   - If \( a > b \), then \( d = a - b \).
   - If \( a < b \), then \( d = b - a \).
   - If \( a = b \), then \( d = 0 \).

**Theorem 2.** The arithmetic distance between two integers \( a \) and \( b \) is the absolute value of their difference, \( |a - b| \).

**Algorithm 2.**

1. **Input:** \( a, b \in \mathbb{Z} \).
2. **Output:** \( d = |a - b| \).
3. **Procedure:**
   - Calculate \( d = a - b \).
   - If \( d < 0 \), then \( d = -d \).
   - Return \( d \).

**Theorem 3.** For any integers \( a, b, c \in \mathbb{Z} \), the following properties hold:

- \( a + (b + c) = (a + b) + c \)
- \( a + 0 = a \)
- \( a + (-a) = 0 \)
- \( a + b = b + a \)
- \( a + (-b) = -(a + b) \)
- \( a + (-a) = 0 \)
- \( a + b = a \) if and only if \( b = 0 \)
- \( a + b = b + a \) if and only if \( a = 0 \) or \( b = 0 \)

**Algorithm 3.**

1. **Input:** \( a, b \in \mathbb{Z} \).
2. **Output:** \( d = a + b \).
3. **Procedure:**
   - If \( a = 0 \) or \( b = 0 \), then \( d = a + b \).
   - Otherwise, \( d = a + b \).

**Theorem 4.** For any integers \( a, b, c \in \mathbb{Z} \), the following properties hold:

- \( a + (b + c) = (a + b) + c \)
- \( a + 0 = a \)
- \( a + (-a) = 0 \)
- \( a + b = b + a \)
- \( a + (-b) = -(a + b) \)
- \( a + (-a) = 0 \)
- \( a + b = a \) if and only if \( b = 0 \)
- \( a + b = b + a \) if and only if \( a = 0 \) or \( b = 0 \)

**Algorithm 4.**

1. **Input:** \( a, b \in \mathbb{Z} \).
2. **Output:** \( d = a + b \).
3. **Procedure:**
   - If \( a = 0 \) or \( b = 0 \), then \( d = a + b \).
   - Otherwise, \( d = a + b \).

**Theorem 5.** For any integers \( a, b, c \in \mathbb{Z} \), the following properties hold:

- \( a + (b + c) = (a + b) + c \)
- \( a + 0 = a \)
- \( a + (-a) = 0 \)
- \( a + b = b + a \)
- \( a + (-b) = -(a + b) \)
- \( a + (-a) = 0 \)
- \( a + b = a \) if and only if \( b = 0 \)
- \( a + b = b + a \) if and only if \( a = 0 \) or \( b = 0 \)

**Algorithm 5.**

1. **Input:** \( a, b \in \mathbb{Z} \).
2. **Output:** \( d = a + b \).
3. **Procedure:**
   - If \( a = 0 \) or \( b = 0 \), then \( d = a + b \).
   - Otherwise, \( d = a + b \).
1.3. Errors in Computer Arithmetic

We have seen that arithmetic weights are an appropriate measure of the number of bits that are in error when the result of an arithmetic operation is rounded to an integer. For example, if we consider the error in the addition of two integers, our main interest in this chapter is to determine the arithmetic weights of the integers. We will use this approach to determine the arithmetic distance from the true result. We shall make use of the fact that addition is an arithmetic operation. For example, if we consider the error in the addition of two integers, our main interest in this chapter is to determine the arithmetic weights of the integers. We will use this approach to determine the arithmetic distance from the true result.
Conversely, suppose that $D_{min} > 2t + 1$. Then, there exist distinct code points $\{A, B, C, D\}$ such that $D(A, B) = D(A, C) = D(A, D) = 2t + 1$.

Conversely, suppose that $D_{min} > 2t + 1$. Then, there exist distinct code points $\{A, B, C, D\}$ such that $D(A, B) = D(A, C) = D(A, D) = 2t + 1$.

Proof: If $A$, $B$, $C$, and $D$ are code points with $D(A, B) > D_{min}$, then $D(A, B) > 2t + 1$. Hence, $D(A, B) > D_{min}$. Therefore, there exists at least one code point that is farther away from $A$ than $B$.

**Theorem 2.3:** The minimum mathematical distance $D_{min}$ of an $n$-code is greater than the distance $D_{min}$ of the code.

**Theorem 2.4:** The minimum mathematical distance $D_{min}$ of an $n$-code is greater than the distance $D_{min}$ of the code.

**Definition:** The minimum mathematical distance $D_{min}$ of an $n$-code is the smallest distance between any two code points in the code.

**Theorem 2.5:** The minimum mathematical distance $D_{min}$ of an $n$-code is at least $2t + 1$.

**Conversely,** suppose that $D(A, B) = D(A, C) = D(A, D) = 2t + 1$. Then, there exist distinct code points $\{A, B, C, D\}$ such that $D(A, B) = D(A, C) = D(A, D) = 2t + 1$. 

**Proof:** If $A$, $B$, $C$, and $D$ are code points with $D(A, B) > D_{min}$, then $D(A, B) > 2t + 1$. Hence, $D(A, B) > D_{min}$. Therefore, there exists at least one code point that is farther away from $A$ than $B$.

**Theorem 2.6:** The minimum mathematical distance $D_{min}$ of an $n$-code is greater than the distance $D_{min}$ of the code.

**Theorem 2.7:** The minimum mathematical distance $D_{min}$ of an $n$-code is greater than the distance $D_{min}$ of the code.

**Definition:** The minimum mathematical distance $D_{min}$ of an $n$-code is the smallest distance between any two code points in the code.

**Theorem 2.8:** The minimum mathematical distance $D_{min}$ of an $n$-code is at least $2t + 1$.
2.4. Errors in Addition modulo m

In the context of error-correcting codes, the concept of addition modulo m is crucial. This operation is defined as:

\[(a + b) \mod m = (a + b) - km\]

where \(k\) is the smallest integer such that \((a + b) - km\) is non-negative. This operation is used in various error-correcting codes to detect and correct errors.

For example, consider a code with a modulo 7 operation. The addition of two numbers, say 3 and 5, modulo 7 is calculated as:

\[3 + 5 \mod 7 = 3 + 5 - 7 = 1\]

This operation is used in the construction of cyclic codes, where the syndrome of a received codeword is calculated by adding all the received symbols, modulo the generator polynomial.

2.3. Arithmetic modulo m

In the context of computer arithmetic, arithmetic modulo m is used to implement operations such as addition, subtraction, and multiplication in finite fields. These operations are crucial in the implementation of error-correcting codes, where the modulo operation is used to detect and correct errors.

For example, consider a code with a modulo 5 operation. The addition of two numbers, say 3 and 4, modulo 5 is calculated as:

\[3 + 4 \mod 5 = 7 - 5 = 2\]

This operation is used in the construction of cyclic codes, where the syndrome of a received codeword is calculated by adding all the received symbols, modulo the generator polynomial.
The arithmetic weight of \( J \) is the modular weight of \( J \) with respect to the modular \( \Gamma \) and is denoted \( \text{wt}_{\Gamma}(J) \).

**Definition:** The modular weight of an integer \( f \) in \( \Gamma \) is the minimal number of \( \Gamma \) that is attained by \( f \) under the action of \( \Gamma \).

The arithmetic weight of \( J \) is defined as the sum of the modular weights of all elements of \( J \).

For results of arithmetic operations in \( \Gamma \), we mention in Section 2.4 the following:

**Theorem 2.3:** Let \( J \) and \( \Gamma \) be any integer rings. Then in \( \Gamma \):

\[
J \oplus \Gamma = J + \Gamma = J
\]

**Proof:** The possible errors are the distance \( f \), which has the same syndrome as \( f \).

**Proposition 2.4:** If \( f \) and \( \Gamma \) are integers, the syndrome of \( f \) is the set of all integers that are congruent to \( f \) modulo \( \Gamma \).

**Lemma 2.5:** For any \( f \) in \( \mathbb{Z}^n \), the set of all \( \Gamma \)-equivalent \( f \) is the set of all integers congruent to \( f \) modulo \( \Gamma \).

**Theorem 2.6:** Let \( J \) and \( \Gamma \) be any integer rings. Then in \( \Gamma \):

\[
J \oplus \Gamma = J + \Gamma = J
\]

**Proof:** The possible errors are the distance \( f \), which has the same syndrome as \( f \).

Theorem 2.7: Let \( J \) and \( \Gamma \) be any integer rings. Then in \( \Gamma \):

\[
J \oplus \Gamma = J + \Gamma = J
\]

**Proof:** The possible errors are the distance \( f \), which has the same syndrome as \( f \).

Theorem 2.8: Let \( J \) and \( \Gamma \) be any integer rings. Then in \( \Gamma \):

\[
J \oplus \Gamma = J + \Gamma = J
\]

**Proof:** The possible errors are the distance \( f \), which has the same syndrome as \( f \).
Finally, we must consider the case when \((1, 0, 1)\) is not a prime.

\[
(\mathcal{L})^M + (\mathcal{L})^M = (\mathcal{L})^M + (\mathcal{L})^M \geq (\mathcal{L} \oplus \mathcal{L})^M
\]

and another application of (5) gives

\[
(\mathcal{L} + \mathcal{L})^M = (\mathcal{L})^M + (w - \mathcal{L})^M \geq (\mathcal{L} \oplus \mathcal{L})^M
\]

These new results allow us to prove the following:

\[
1 - (\mathcal{L} + \mathcal{L})^M = (w - \mathcal{L})^M
\]

and hence, with the NVE of \(L\), we have

\[
(\mathcal{L} + \mathcal{L})^M \geq (\mathcal{L} + \mathcal{L})^M = (\mathcal{L} \oplus \mathcal{L})^M
\]

If \(w > \mathcal{L} + \mathcal{L}\), we have \(m = \mathcal{L} + \mathcal{L}\) and again (5) gives

\[
(\mathcal{L} + \mathcal{L})^M \geq (\mathcal{L} + \mathcal{L})^M = (\mathcal{L} \oplus \mathcal{L})^M
\]

Hence, the theorem will be proved if we prove, for the case where

\[
(\mathcal{L} \oplus \mathcal{L})^M + (\mathcal{L} \oplus \mathcal{L})^M \geq (\mathcal{L} \oplus \mathcal{L})^M + (\mathcal{L} \oplus \mathcal{L})^M
\]

Similarly, since then, by property (11), we have

\[
(\mathcal{L} \oplus \mathcal{L})^M = (\mathcal{L} \oplus \mathcal{L})^M
\]

and

\[
(\mathcal{L} \oplus \mathcal{L})^M = (\mathcal{L} \oplus \mathcal{L})^M
\]

Then the theorem also holds for \((\mathcal{L})^M = (\mathcal{L})^M\) and

\[
(\mathcal{L} \oplus \mathcal{L})^M = (\mathcal{L} \oplus \mathcal{L})^M
\]

This completes the proof for \((\mathcal{L})^M = (\mathcal{L})^M\) and

\[
(\mathcal{L} \oplus \mathcal{L})^M = (\mathcal{L} \oplus \mathcal{L})^M
\]
nonvoid as claimed.

code point within distance 1 of either and a decoded word will be an

\( y < (N' y) y + 1 \), so that no \( y \) can ever decode to \( (N' y) y + 1 \).

Theorem 2.2. The modulus difference between the images of the

two elements of the modular arithmetic with \( m \geq 2 \), where \( m \) is one of the modulus \( 2 \), \( 2 \), \( 2 \), or \( 2 \) is the modular

function for any set of integers.

If \( g \) follows the normal properties (1) of the modular

function for \( Z \), where \( m \geq 2 \), then \( g \) and \( g \) are modular and can be used

to derive arithmetic congruences. Hence the same symmetry will be used

for both of these modular congruences.

Theorem 2.3. For any \( m \), \( y \) and \( y \) where \( m \) is one of the modulus

we have the following congruence:

\[ (y \oplus y)^m = (y \oplus y)^m \]

Theorem 2.4. The modulus difference between the images of the

arithmetic function for \( Z \) just as we used arithmetic weight to define a

distance.

If \( g \) follows from the normal properties (1) of the modular

function for \( Z \), where \( m \geq 2 \), then \( g \) and \( g \) are modular and can be used

to derive arithmetic congruences. Hence the same symmetry will be used

for both of these modular congruences.

Theorem 2.5. For any \( m \), \( y \) and \( y \) where \( m \) is one of the modulus

we have the following congruence:

\[ (y \oplus y)^m = (y \oplus y)^m \]

We now note that \( N \) and \( N \) are distinct code points, then

\[ (N' y)^{m} \]

we have the normal properties (1) of the modular

function for \( Z \), where \( m \geq 2 \), then \( g \) and \( g \) are modular and can be used

to derive arithmetic congruences. Hence the same symmetry will be used

for both of these modular congruences.

Theorem 2.6. For any \( m \), \( y \) and \( y \) where \( m \) is one of the modulus

we have the following congruence:

\[ (y \oplus y)^m = (y \oplus y)^m \]

We now note that \( N \) and \( N \) are distinct code points, then

\[ (N' y)^{m} \]

we have the normal properties (1) of the modular

function for \( Z \), where \( m \geq 2 \), then \( g \) and \( g \) are modular and can be used

to derive arithmetic congruences. Hence the same symmetry will be used

for both of these modular congruences.