1 The Algebraic Normal Form

Boolean maps can be expressed by polynomials—this is the algebraic normal form (ANF). The degree as a polynomial is a first obvious measure of nonlinearity—linear (or affine) maps have degree 1.

In this section we show how to determine the ANF and the degree of a Boolean map that is given by its value table.

1.1 Boolean functions and maps

We denote by $\mathbb{F}_2$ the Galois field with two elements. We use algebraic notation: $+$ is the addition in the field $\mathbb{F}_2$ and in vector spaces over $\mathbb{F}_2$. We reserve the character $\oplus$ for direct sums.

A **Boolean function** of n variables is a function

$$f\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2.$$

A **Boolean map** (or vector valued Boolean function) is a map

$$f\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^q.$$

Cryptologists like to call this an "S-box" or "substitution box".

Let $\mathcal{F}_n$ be the set of all Boolean functions on $\mathbb{F}_2^n$. We identify the set of all mappings $\mathbb{F}_2^n \longrightarrow \mathbb{F}_2^q$ with $\mathcal{F}_n^q$ in a natural way.

Usually a Boolean function is given by its **truth table**—that is by its value table or the graph of the function. This table is—in a canonical way—lexicographically ordered by the arguments $x \in \mathbb{F}_2^n$. In other words, by the natural order of the numbers $a = 0, \ldots, 2^n - 1$, represented in base 2 as

$$a = x_1 \cdot 2^{n-1} + \cdots + x_{n-1} \cdot 2 + x_n$$

and identified with the vectors $(x_1, \ldots, x_n) \in \mathbb{F}_2^n$.

The logical negation of the function $f \in \mathcal{F}_n$ is the function $\bar{f} = f + 1$.

Let $\mathcal{L}_n$ be the set of all linear forms, that is the dual space of $\mathbb{F}_2^n$. Let $\{e_1, \ldots, e_n\}$ be the canonical basis of $\mathbb{F}_2^n$ and $\cdot$ the canonical dot product. The identification of the linear form $x \mapsto u \cdot x$ with the vector $u \in \mathbb{F}_2^n$ gives the (basis dependent) isomorphism $\mathbb{F}_2^n \cong \mathcal{L}_n$ of vector spaces.

Let $\mathcal{A}_n$ be the set of affine functions $\mathbb{F}_2^n \longrightarrow \mathbb{F}_2$. There are 2^{n+1} of them—the linear functions and their negations, that is the functions

$$f(x) = \alpha(x) + c \qquad \text{where } \alpha \in \mathcal{L}_n \text{ and } c \in \mathbb{F}_2.$$

Let $\chi : \mathbb{F}_2 \longrightarrow \mathbb{C}^\times$ be the only nontrivial group homomorphism ("character"): $\chi(0) = 1$, $\chi(1) = -1$, or $\chi(a) = (-1)^a = 1 - 2a$, the last equation "par abus de notation" (identifying $1 \in \mathbb{F}_2$ with $1 \in \mathbb{R}$). In particular χ

is real valued. With each Boolean function $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ we associate its **character form** as $\chi_f := \chi \circ f : \mathbb{F}_2^n \longrightarrow \mathbb{R}^\times \subseteq \mathbb{C}^\times$, in short

$$\chi_f(x) = (-1)^{f(x)}.$$

Obviously $\chi_{f+g} = \chi_f \chi_g$. The formula for the product of two Boolean function is slightly more complicated. From the table

a	b	$a+b$	ab	$\chi(a)$	$\chi(b)$	$\chi(a+b)$	$\chi(ab)$
0	0	0	0	1	1	1	1
0	1	1	0	1	−1	−1	1
1	0	1	0	−1	1	−1	1
1	1	0	1	−1	−1	1	−1

we get the formula

$$\chi(a+b) + 2\chi(ab) = 1 + \chi(a) + \chi(b) \quad \text{for all } a, b \in \mathbb{F}_2.$$

Therefore for $f, g \in \mathcal{F}_n$ we have the product formula

$$2\chi_{fg} = 1 + \chi_f + \chi_g - \chi_f \chi_g.$$

Definition 1 For two Boolean functions $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$ the **Hamming distance** is the number of arguments where the functions differ:

$$d(f,g) := \#\{x \in \mathbb{F}_2^n \mid f(x) \neq g(x)\};$$

in other words: the number of ones in the truth table of $f + g$.

Remarks

1. d is a metric on $\mathcal{F}_n$. The transitivity of d for $f, g, h \in \mathcal{F}_n$ is shown as follows: If $f(x) \neq h(x)$, then $f(x) \neq g(x)$ or $g(x) \neq h(x)$; therefore

$$\begin{aligned} d(f,g) + d(g,h) &= \#\{x \mid f(x) \neq g(x)\} + \#\{x \mid g(x) \neq h(x)\} \\ &\geq \#\{x \mid f(x) \neq h(x)\} = d(f,h). \end{aligned}$$

2. If $\bar{g} = g + 1$ is the negation of g, then $d(f, \bar{g}) = 2^n - d(f,g) =$ the number of arguments, where f and g coincide.

1.2 Boolean linear forms

For $u, x \in \mathbb{F}_2^n$ we can write the canonical dot product as

$$u \cdot x = \sum_{i=1}^{n} u_i x_i = \sum_{u_i=1} x_i = \sum_{i \in \mathrm{Supp}(u)} x_i$$

with the "support" of u,

$$\mathrm{Supp}(u) = \{i = 1, \ldots, n \mid u_i \neq 0\} = \{i = 1, \ldots, n \mid u_i = 1\}.$$

This means that the dot product with a fixed vector u is the partial sum over the coordinates of x in the support $I \subseteq \{1, \ldots, n\}$ of u or the **parity** of x over I. Since every linear form on a finite dimensional vector space can be written as a dot product with a fixed vector, we have shown:

Proposition 1 *The linear forms on $\mathbb{F}_2^n$ are the parity functions over the subsets $I \subseteq \{1, \ldots, n\}$.*

In other words every linear form can be written as

$$\alpha_I(x) = \sum_{i \in I} x_i \qquad \text{for all } x = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$$

with a subset $I \subseteq \{1, \ldots, n\}$. Thereby we have a natural bijection between the 2^n element set $\mathcal{L}_n$ and the power set $\mathfrak{P}(\{1, \ldots, n\})$.

Other common expressions are—for $I = \{i_1, \ldots, i_r\}$—:

$$\alpha_I(x) = x[I] = x[i_1, \ldots, i_r] = x_{i_1} + \cdots + x_{i_r}.$$

1.3 Functions and polynomials

Let $T = (T_1, \ldots, T_n)$ be an n-tuple of indeterminates. Every polynomial $p \in \mathbb{F}_2[T]$ defines a function $\Psi(p) \in \mathcal{F}_n$ by substitution:

$$\Psi(p)(x_1, \ldots, x_n) := p(x_1, \ldots, x_n).$$

The **substitution homomorphism**

$$\Psi : \mathbb{F}_2[T] \longrightarrow \mathcal{F}_n,$$

is a homomorphism of $\mathbb{F}_2$-algebras.

Lemma 1 *Ψ is surjective.*

Proof. (By induction over n) The induction begin $n = 0$ is trivial—the two constant polynomials correspond to the two constant functions. Now let $n \geq 1$. Let $x' = (x_1, \ldots, x_{n-1}) \in \mathbb{F}_2^{n-1}$ be the first $n-1$ components of $x = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$. Consider a function $f \in \mathcal{F}_n$. By induction we have for $b = 0, 1$

$$f(x', b) = p_b(x') \qquad \text{for all } x' \in \mathbb{F}_2^{n-1}$$

where $p_0, p_1 \in \mathbb{F}_2[T_1, \ldots, T_{n-1}]$; in the case $n = 1$ these are constants. Then

$$f(x', x_n) = (1 + x_n)p_0(x') + x_n p_1(x') \qquad \text{for all } x = \in \mathbb{F}_2^n.$$

Therefore $f = \Psi(p)$ with $p = p_0 + (p_0 + p_1)T_n$. $\diamond$

Note. An analogous statement holds over any finite field. The proof in the general case is slightly more complicated and uses interpolation; this is useful in cryptology too, since—over the Galois field $\mathbb{F}_{2^n}$—it is the basis for interpolation attacks on block ciphers. The following proposition 2 generalizes in the same way.

What is the kernel of the homomorphism Ψ? Since $b^2 = b$ for all $b \in \mathbb{F}_2$, all the polynomials $T_1^2 - T_1, \ldots, T_n^2 - T_n$ are in the kernel, so is the ideal

$$\mathfrak{a} \trianglelefteq \mathbb{F}_2[T]$$

they generate. The induced homomorphism

$$\bar{\Psi} : \mathbb{F}_2[T]/\mathfrak{a} \longrightarrow \mathcal{F}_n$$

is surjective. Each element of the factor algebra $\mathbb{F}_2[T]/\mathfrak{a}$ can be written as a linear combination of the monomials that have a degree ≤ 1 in each T_i. There are 2^n of these, namely the products

$$T^I := T_{i_1} \cdots T_{i_r}$$

for arbitrary subsets

$$I = \{i_1, \ldots, i_r\} \subseteq \{1, \ldots, n\}.$$

So the pre-image of $\bar{\Psi}$ is a vector space of dimension $\leq 2^n$ over $\mathbb{F}_2$. Therefore its dimension must be $= 2^n$, and $\bar{\Psi}$ must be an isomorphism. We have shown:

Proposition 2 (Algebraic normal form, ANF) *Every Boolean function*

$$f \colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$$

can uniquely be written as a polynomial in n indeterminates that has degree ≤ 1 *in each single indeterminate:*

$$f(x_1, \ldots, x_n) = \sum_{I \subseteq \{1,\ldots,n\}} a_I x^I,$$

where the monomial x^I *is the product*

$$x^I = \prod_{i \in I} x_i,$$

and $a_I = 0$ *or* 1.

$k \in \mathbb{N}$	$u \in \mathbb{F}_2^3$	$I \subseteq \{1,2,3\}$	monomial
0	000	$\emptyset$	1
1	001	$\{3\}$	T_3
2	010	$\{2\}$	T_2
3	011	$\{2,3\}$	T_2T_3
4	100	$\{1\}$	T_1
5	101	$\{1,3\}$	T_1T_3
6	110	$\{1,2\}$	T_1T_2
7	111	$\{1,2,3\}$	$T_1T_2T_3$

Table 1: Various interpretations of a binary vector, example $n = 3$

The various interpretations of a binary vector $u \in \mathbb{F}_2^n$ are illustrated by a simple example in table 1, as well as the "canonical"' associations between them.

There is an alternative derivation of the algebraic normal form using normalization of Boolean expressions. This however doesn't generalize to other finite fields.

Corollary 1 *Every Boolean map $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^q$ is given by a q-tuple of polynomials $(p_1, \ldots, p_q) \in \mathbb{F}_2[T_1, \ldots, T_n]$ that have all partial degrees ≤ 1.*

(By "partial degree" we mean the degree in a single indeterminate T_i.)

Corollary 2 *Every Boolean map $f\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^q$ has a unique expression as*

$$f(x_1, \ldots, x_n) = \sum_{I \subseteq \{1,\ldots,n\}} x^I a_I$$

with coefficients $a_I \in \mathbb{F}_2^q$.

Definition 2 The degree of a Boolean map f as a polynomial,

$$\operatorname{Deg} f = \max\{\#I \mid a_I \neq 0\},$$

is called **(algebraic) degree** of f.

Remarks

1. In general $\operatorname{Deg} f \leq n$.
2. f is affine $\Leftrightarrow \operatorname{Deg} f \leq 1$.
3. The degree of a Boolean map f is the maximum of the degrees of its component polynomials $p_1, \ldots, p_q$.

The algebraic degree is a first measure of nonlinearity of f. It is obviously invariant under affine transformations in preimage and image.

Exercise How many functions $\mathbb{F}_2^2 \longrightarrow \mathbb{F}_2$ are there? Enumerate them all.

1.4 Evaluation and interpolation

The algebraic normal form has the advantage that we can immediately read off the algebric degree. We also easily recognize the "structure" of a Boolean map, moreover we could comfortably classify orbits under the action of affine transforms and find "reduced" normal forms. On the other hand the truth table more clearly shows the "behaviour" of a map, and—as we shall see—leads to better detection of hidden linearity.

Therefore a method of switching between these two representations is hihgly needed. The transition from the algebraic normal form to the truth table is the evaluation of the component polynomials at all arguments. The inverse transformation is interpolation as in the proof of proposition 2. Here we show how to do this by an efficient algorithm.

The naive evaluation of a Boolean function $f \in \mathcal{F}_n$, or $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$, at all arguments $x \in \mathbb{F}_2^n$ costs 2^n evaluations $f(x)$ each with up to 2^n summands with up to $n-1$ multiplications. This makes an order of magnitude of $n \cdot 2^n \cdot 2^n$; since the size of the input is $N = 2^n$, the expense is essentially quadratic: $N^2 \cdot {}^2\log(N)$. A binary recursion or "divide and conquer strategy"—a recursive division in two partial tasks of half the input size—will lead to a significantly faster algorithm.

To begin with let's write the algebraic normal form in a slightly modifyed way:

$$f = \sum_{u \in \mathbb{F}_2^n} \alpha_f(u) T^{(u)} \qquad \text{with the monomial} \quad T^{(u)} = \prod_{i \in \mathrm{Supp}(u)} T_i.$$

The association between a binary vector u and a monomial $T^{(u)}$ is as in table 1. Then the $\alpha_f(u)$ themselves form a function $\alpha_f \in \mathcal{F}_n$ that we call the **coefficient representation** of f. On the other hand the truth table is represented by the family $(f(x))_{x \in \mathbb{F}_2^n}$, that means simply by $f \in \mathcal{F}_n$ itself. With this interpretation the evalution is the transformation

$$\Theta_n \colon \mathcal{F}_n \longrightarrow \mathcal{F}_n, \qquad \alpha_f \mapsto f.$$

The binary recursion starts with the unique decomposition

$$f = f_0 + T_1 f_1 \qquad \text{with} \quad f_0, f_1 \in \mathbb{F}_2[T_2, \ldots, T_n]$$

from the proof of lemma 1 (where the notation was different). Then for $y \in \mathbb{F}_2^{n-1}$ we have

$$\begin{aligned} f(0,y) &= f_0(y), \\ f(1,y) &= f_0(y) + f_1(y). \end{aligned}$$

In general let $0 \le i \le n$, $u \in \mathbb{F}_2^{n-i}$, and $f_u \in \mathbb{F}_2[T_{n-i+1}, \ldots, T_n]$ be defined by

$$f_u := \sum_{v \in \mathbb{F}_2^i} \alpha_f(u,v) T^{(v)}.$$

Then in the case $i = n$ and $u = 0 \in \mathbb{F}_2^0$

$$f_u = \sum_{v \in \mathbb{F}_2^n} \alpha_f(v) T^{(v)} = f.$$

On the other extreme, in the case $i = 0$ and $u \in \mathbb{F}_2^n$, we have

$$f_u = \alpha_f(u) \quad \text{constant.}$$

In between, for $1 \le i \le n$ and $u \in \mathbb{F}_2^{n-i}$, we have

$$f_u = f_{(u,0)} + T_{n-i+1} f_{(u,1)}.$$

Therefore the evaluation follows the recursion (for $y \in \mathbb{F}_2^{i-1}$)

$$\begin{aligned} f_u(0,y) &= f_{(u,0)}(y), \\ f_u(1,y) &= f_{(u,0)}(y) + f_{(u,1)}(y). \end{aligned}$$

We convert this recursion into an iterative procedure by defining a sequence of vectors $x^{(i)} = (x_u^{(i)})_{u \in \mathbb{F}_2^n}$ with coefficients in $\mathbb{F}_2$ as follows: Let

$$x^{(0)} := (\alpha_f(u))_{u \in \mathbb{F}_2^n},$$

be the initial vector. For $i = 1, \ldots, n$ let the n-bit index decompose into $u\xi v$ with $n - i$ bits u, one bit ξ, and $i - 1$ bits v, and let

$$\begin{aligned} x_{u0v}^{(i)} &:= x_{u0v}^{(i-1)}, \\ x_{u1v}^{(i)} &:= x_{u0v}^{(i-1)} + x_{u1v}^{(i-1)}. \end{aligned}$$

By induction we get:

Proposition 3 *Let $(x^{(i)})$ be the recursively defined sequence as above. Then*

$$x_{(u,y)}^{(i)} = f_u(y) \qquad \textit{for all} \quad u \in F_2^{n-i}, y \in \mathbb{F}_2^i;$$

in particular

$$x^{(n)} = (f(u))_{u \in \mathbb{F}_2^n}$$

is the truth table of f.

In the opposite direction the iteration has the same structure:

$$\begin{aligned} x_{u0v}^{(i-1)} &:= x_{u0v}^{(i)}, \\ x_{u1v}^{(i-1)} &:= x_{u0v}^{(i)} + x_{u1v}^{(i)}. \end{aligned}$$

Therefore the inverse transformation of Θ_n, the construction of the coefficient representation given the truth table, follows exactly the same procedure, so it must be identical to Θ_n:

Corollary 1 *The evaluation transformation Θ_n is an involution of $\mathcal{F}_n$.*

In particular *the inverse application of Θ_n also determines the algebraic degree of a Boolean function* that is given by its truth table.

To implement the evaluation procedure in a common programming language we interpret the indices as natural numbers $k = \sum k_{n-i} 2^i$ in the integer interval $[0 \dots 2^n - 1]$ as in table 1. Then in the iteration formula we have $u1v = u0v + 2^i$, and the equations become

$$x_k^{(i+1)} = \begin{cases} x_k^{(i)}, & \text{if } k_{n-i} = 0, \\ x_{k-2^i}^{(i)} + x_k^{(i)}, & \text{if } k_{n-i} = 1, \end{cases}$$

for $k = 0, \dots, 2^n - 1$. The bit k_{n-i} is extracted from k by the formula

$$k_{n-i} = \left\lfloor \frac{k}{2^i} \right\rfloor \bmod 2 = (k \gg i) \bmod 2,$$

where $k \gg i$ is the i bit shift to the right. Now the entire algorithm in "pidgin Pascal" looks as follows:

Prozedur [REV] (Recursive evaluation)

Input and output parameters: A vector x of length 2^n, $x[0], \dots, x[2^n - 1]$.

Local variables: A vector y of length 2^n, $y[0], \dots, y[2^n - 1]$.
Loop counters $i = 0, \dots, n-1$ and $k = 0, \dots, 2^n - 1$.

Instructions:

```
For i = 0,...,n-1:
    For k = 0,...,2^n - 1:
        If (((k >> i) mod 2) = 1) then y[k] := x[k - 2^i] XOR x[k]
            else y[k] := x[k]
    For k = 0,...,2^n - 1:
        x[k] := y[k]
```

Here x and y are vectors over $\mathbb{F}_2$, that means bit sequences, and the addition in $\mathbb{F}_2$ is represented by the Boolean operator XOR.

The expense is $n \cdot 2^n$ loop executions, each with one binary addition, one shift, and one single bit complement. In summary that makes $3n \cdot 2^n$ "elementary" operations. The procedure essentially needs $2 \cdot 2^n$ bits of memory. Expressed as a function if the input size $N = 2^n$, the expense is almost linear: $3N \cdot {}^2\log N$.

The corresponding C procedure in the sources is called `rev`.